Conrad, P. T., Tang, A., Hundhausen, C., Buffardi, K., & Haque, S. (2026, July 20–22). *What Makes A Good Commit? A Delphi Study* [Paper presentation]. 38th International Conference on Software Engineering Education and Training (CSEE&T 2026), Florence, Italy. To appear in *Lecture Notes in Computer Science* (LNCS), Springer. https://cseet26.techconf.org/

Awarded Best Paper by the program committee out of 95 submissions (26 accepted papers): https://cseet26.techconf.org/track/award

What Makes A Good Commit? A Delphi Study

Phillip T. Conrad[1][0000−0003−3676−7775], Aileen Tang[2][0009−0006−4754−4138], Christopher Hundhausen[2][0000−0001−5128−905X], Kevin Buffardi[3][0000−0002−4205−888X], and Summit Haque[2][0000−0002−4658−4393]

[1] UC Santa Barbara, Santa Barbara CA 93117 USA
[2] Oregon State University, Corvallis, OR 97331 USA
[3] Chico State University, Chico CA 95929 USA

Abstract. To teach students software engineering skills, instructors must be able to explain what quality looks like and be able to assess it. Operationally, this means that we need comprehensive rubrics for assessing artifacts such as commits, issues, pull requests and code reviews. For software engineering education research, such rubrics are also needed for assessing the effectiveness of various interventions. Ideally, these rubrics would be rooted in some ground truth, rather than reflecting the subjective opinions of individual instructors; this is particularly important if rubrics are used in empirical studies. In this paper, we seek to establish a "ground truth" for the assessment of commits by using the Delphi process, an empirical method for establishing expert consensus around a complex question. We assembled a diverse panel of 16 experts in software engineering with voices from both industry and academia. By applying the Delphi process through four rounds of questions to our panel, we derived a set of rubric items for commits that represent the consensus view of the expert panel. Our contribution is both the rubric items themselves and a roadmap by which future researchers can arrive at similar rubrics for other software engineering artifacts.

Keywords: Software Engineering · Software Engineering Education · Empirical Software Engineering · Version Control · Commits · Delphi · Assessment

1 Introduction

To perform meaningful assessment and engage in continuous improvement, computer science instructors must be able to define the qualities of desirable software engineering artifacts, and measure whether students' work products demonstrate these qualities. However, the definition of "desirable" can vary greatly across instructors and courses, resulting in reduced instructional consistency and uneven evaluation practices.

This lack of consensus has consequences for both teaching and research. When instructors use personal or idiosyncratic criteria for assessing software engineering artifacts, students receive inconsistent signals about professional expectations and may struggle to develop transferable skills. Variability in assessment limits

our ability to share pedagogical materials and rubrics across institutions [28]. When student work is evaluated against unclear standards, feedback becomes less actionable and learning outcomes may suffer [13].

Developing standards of quality for version control artifacts such as commits is particularly important. Commits serve as atomic units of software development work and form an essential communication channel among developers [30]. Many students may graduate without understanding effective commit practices [8]. Instructors who wish to teach and assess commits must either develop personal rubrics, adopt others' rubrics that may not fit their context, or avoid formal assessment entirely.

This problem extends to software engineering education research. Researchers investigating version control practices, collaborative development, and software engineering processes need validated instruments to measure outcomes [23]. Without rubrics grounded in expert consensus, findings risk reflecting only individual investigators' views rather than broader community standards. The lack of standardized criteria also makes it difficult to compare results from different studies or build cumulative knowledge through replication studies [12].

To develop a more stable, expert-validated definition of high-quality commits, we applied the Delphi method with a panel of software engineering professionals from both industry and academia. Using a multi-round Delphi process, we established a rigorous empirical basis for the qualities of a desirable commit, along with a corresponding set of rubric items to use in evaluating commits.

2 Related Work

2.1 Delphi Process

The Delphi process was originally developed by researchers at the Rand Corporation tasked by the US military with making certain predictions about questions involving the Cold War [9]. The core principle is to draw on the collective knowledge of a group of experts, while avoiding various biases that can arise with direct interpersonal interactions, such as deferring to people who are more assertive, extroverted, or senior [1,6]. Delphi panels are traditionally anonymous to one another, and remain so after the conclusion of the research [1]. We maintain this approach, while disclosing demographic information about the panelists so that readers can have confidence that the panelists have the expertise to provide informed opinions, and represent a broad spectrum of voices (see Section 3).

2.2 Assessment in Software Engineering Education

Assessment in software engineering education must capture both foundational knowledge and practical competencies, reflecting the diversity of tasks students complete—from small guided exercises to larger, independent projects [11]. Authentic assessment has become central in higher education because it mirrors real-world professional contexts and better prepares students for industry expectations [35,25]. However, studies note considerable variation in how authenticity

is interpreted and implemented, with digital technologies unevenly integrated into assessment design and feedback processes [14].

Alongside authenticity, effective assessment frameworks emphasize feedback, reflection, and collaboration. In software engineering, structured peer assessment has demonstrated value by improving teamwork, accountability, and equitable contribution within student project groups [29]. Collectively, the literature suggests that while assessment practices are evolving toward more authentic and learner-centered models, further refinement and standardization are needed to support robust competency development in software engineering programs.

2.3 Commits

In version control, a *commit* is a unit of change in a code base, typically consisting of a list of changes to a set of files along with metadata about those changes. This concept exists in many version control systems and goes by different names. For example, what git calls a commit [5] is referred to as a *changeset* in Mercurial [22], and a *changelist* in Perforce [24].

Metadata for a commit includes the *commit message*, a brief comment in natural language describing the change. Some developers have adopted particular conventions and notations for commit messages. For example, the *Conventional Commits* standard specifies notations to distinguish among bug fixes, features, and several other change types, as well as to distinguish between breaking and non-breaking changes [7].

Commits can function as the unit of analysis for many kinds of investigations in software engineering. Alomar et al. [3] used commit messages to investigate refactoring in code bases. Li and Paxson [20] used commit messages as a basis for studying security patches. Reis et al. [27] argued that commit messages for security patches are often not descriptive enough, and proposed commit messages conventions to address this [26].

2.4 Assessment of Commit Quality

Several prior publications have examined what constitutes commit quality. Hundhausen et al. [16] used the "Goal-Question-Metric" approach from software engineering research to develop metrics for commit, issue and product quality for student work in software engineering courses. With the goal that "Students will write commit messages that are consistent with industry expectations for quality," Hundhausen et al. devised a rubric with the following criteria:

- **atomic**: Do the code changes in the commit deal with one and only one concern?
- **accurate**: Does the commit message describe all changes, and only those changes, made in the commit?
- **precise**: Does the commit message unambiguously describe the changes made, situating the commit in the context of the code base or project?

– **justified**: Does the commit message describe why the change was made from the perspective of the end user?

The work of Hundhausen et al. yielded some important contributions: (1) it demonstrated the value of applying a rubric to commits, (2) it showed that a group of academics can converge on high inter-rater reliability when classifying commits relative to a rubric; (3) it found that the baseline level of commit quality is low among undergraduates, indicating there is considerable room for improvement. However, a weakness of this work is that the authors' proposed rubric was not based on a rigorous analysis of empirical data, but rather on a subjective summary of "industry discussions of good practices" such as [17]. Our work described in this paper improves upon limitations to the preceding literature by adopting a more rigorous Delphi process.

In related work, Agarwal et al. [2] explored commit quality in five High Performance Computing (HPC) projects and three non-HPC projects. The authors presented strong rationales for three metrics: the fraction of unique commit comments, their size, and the number of files per commit) but cite no prior work or empirical basis for choosing these.

Tian et al. [32,31] sought to establish a standard for "good commit messages," noting that "a widely recognized standard of high-quality messages is lacking." They reviewed 46 academic papers for definitions of "good commit messages", reviewed the top 50 results from a Google search of the phrase "good commit message", and solicited opinions from "30 experienced OSS developers" which they defined as having contributed more than 10 commits to a group of open-source projects they were studying. From this analysis they came up with a simplified rubric for commit messages: they should summarize *what* (the changes in the commit) and *why* (the reasons for the changes). This simplified rubric served as a useful basis for their analysis, but it falls short of the goal we set in this paper. First, their rubric applied only to commit messages, while ours examines commits as a whole. Further, while the authors have made a good faith effort to document their process (for example, providing a dataset of the 46 papers they surveyed), their analysis lacks the kind of specifics that usually accompany a systematic literature survey of this kind (e.g. search terms, inclusion/exclusion criteria, etc.) [18]. This same lack of detail extends to the content analysis of experts and search results. Nonetheless, Tian et al. makes significant progress towards showing the extent to which there is room for improvement in commit messages, which further motivates our emphasis on evaluating this skill in our students.

In sum, prior work has examined commit quality from various perspectives, including quantitative metrics, individual researcher assessments, and domain-specific practices. However, no prior work has systematically sought expert consensus on the desirable qualities of commits for educational assessment and empirical research purposes. Our work addresses this gap by using the Delphi method to establish consensus-based quality criteria that can serve as a foundation for rubrics used in teaching practice, computing education research, and empirical software engineering research.

We also note that there is a considerable body of work on the *automated generation of commit messages* (see, for example, [34,19,15,33,21]). Tian et al. [32] provides an entry point for this work. Although this line of work is not directly relevant to our goal, it underscores its importance. Indeed, the emergence of AI-generated commit messages makes it increasingly necessary for students to be able to *assess* whether the generated messages are appropriate, and to be able to assess the entirety of a commit, not only the commit message.

3 Recruitment and Selection

Potential Delphi panel members were recruited through the authors' professional and academic networks, along with outreach efforts on LinkedIn. We took care to include individuals from underrepresented groups in our recruiting to ensure that diverse perspectives would be reflected in the panel responses and discussion. Recruitment invitations continued over a two-month period in Fall 2024 to improve the response rate and sample diversity.

Interested candidates completed a recruitment form where they self-reported demographic and career information. We used different screening criteria for professional developers and software engineering educators. For the professional developers, our criteria were:

- Has at least three years of professional software development experience (5-7 preferred)
- Has done professional software development or been a manager of developers
- Has experience with a software development process that includes git commits, issues, pull requests and code reviews (or equivalent artifacts in some other version control system)

For the academics, our criteria were:

- Must have taught a course with a software engineering project for at least 5 years.
- Course projects must use git commits, issues, pull requests, code reviews (or equivalent artifacts in another version control system).

In selecting from the qualified pool of candidates, we prioritized diverse representation across categories of employers (e.g. large technology companies, healthcare technology, developer tools and security, enterprise SaaS, and smaller startups), education (unique alma maters), duration of experience, and demographics. The selected pool included both industry practitioners and academics, and reflected diversity across gender and ethnicity (See Table 1).

Table 1. Participant Characteristics Across Rounds

	Round 1			Round 2		Round 3		Round 4		Core
Characteristic	**Qualified**	**Invited**	**Responded**	**Invited**	**Responded**	**Invited**	**Responded**	**Invited**	**Responded**	Panelists
	(N=80)	**(N=26)**	**(N=16)**	**(N=26)**	**(N=16)**	**(N=20)**	**(N=18)**	**(N=20)**	**(N=17)**	**(N=12)**
Gender, n (%)										
Men	58 (72.5%)	17 (65.4%)	12 (75.0%)	17 (65.4%)	10 (62.5%)	14 (70.0%)	12 (66.7%)	14 (70.0%)	13 (76.5%)	9 (64.3%)
Women	20 (25.0%)	9 (34.6%)	4 (25.0%)	9 (34.6%)	6 (37.5%)	6 (30.0%)	6 (33.3%)	6 (30.0%)	4 (23.5%)	3 (21.4%)
Other[a]	2 (2.5%)	0	0	0	0	0	0	0	0	0
Race/Ethnicity, n (%)										
White	50 (62.5%)	16 (61.5%)	11 (68.8%)	15 (57.7%)	10 (62.5%)	12 (60.0%)	11 (61.1%)	12 (60.0%)	10 (58.8%)	8 (57.1%)
Asian	16 (20.0%)	5 (19.2%)	3 (18.8%)	6 (23.1%)	4 (25.0%)	6 (30.0%)	5 (27.8%)	6 (30.0%)	5 (29.4%)	2 (14.3%)
Hispanic/Latin Am.	7 (8.8%)	3 (11.5%)	2 (12.5%)	3 (11.5%)	2 (12.5%)	2 (10.0%)	2 (11.1%)	2 (10.0%)	2 (11.8%)	2 (14.3%)
Black/African Am.	2 (2.5%)	2 (7.7%)	0	2 (7.7%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Not disclosed	5 (6.3%)	0	0	—	—	—	—	—	—	—
Role, n (%)										
Academic	14 (17.5%)	8 (30.8%)	6 (37.5%)	8 (30.8%)	5 (31.3%)	7 (35.0%)	6 (33.3%)	7 (35.0%)	7 (41.2%)	5 (35.7%)
Industry	66 (82.5%)	18 (69.2%)	10 (62.5%)	18 (69.2%)	11 (68.8%)	13 (65.0%)	12 (66.7%)	13 (65.0%)	10 (58.8%)	7 (50.0%)
Experience, M (SD)										
Professional (yrs)	10.8 (7.5)	10.8 (7.5)	11.4 (7.5)	10.9 (7.4)	11.0 (7.1)	11.2 (7.2)	11.0 (7.1)	11.0 (7.1)	10.0 (6.5)	10.5 (6.3)
Teaching (yrs)	13.3 (12.6)	13.3 (12.6)	10.4 (10.5)	10.0 (9.8)	10.0 (10.6)	10.5 (11.5)	10.0 (10.6)	10.0 (10.6)	10.0 (10.6)	12.6 (11.5)
Unique Alma Maters[b]	39	23	16	22	16	18	17	19	18	13
Unique Employers	63	24	15	23	16	18	17	17	15	11
Response Rate	—	—	61.54%	—	61.54%	—	90.00%	—	85.00%	—

[a]Includes non-binary (1) and prefer not to answer (1).
[b]Count may exceed number of participants since both undergraduate and graduate institutions were included.

Our recruiting efforts yielded 80 individuals that met the acceptance criteria; the demographics of this group and all subsequent participants appear in Table 1. From the 80 qualified individuals (Column **Qualified**), we invited 26 to fill out our ROUND 1 survey (Column **Invited**). We received responses from 16 individuals; our analysis of these responses and the responses from subsequent rounds is described in Section 4. The demographics of those invited and who completed subsequent rounds are shown in the remaining columns; of these, 12 individuals completed all four rounds and are referred to as the *Core Panelists* (Column **Core Panelists**).

4 Delphi Process and Results

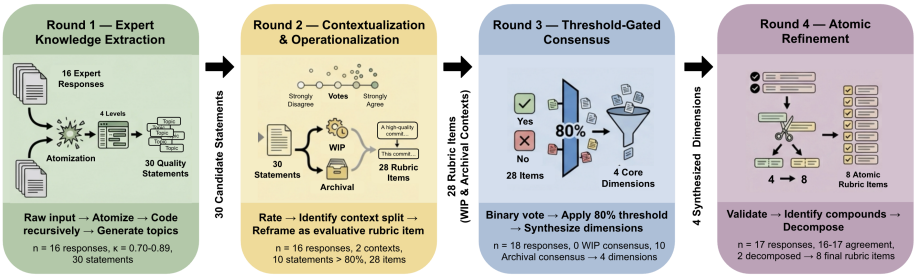

Fig. 1. Delphi Process Overview

As shown in Figure 1, our study consisted of four rounds of the Delphi process. In each round, we sent our panelists an online survey and asked for responses within 2–3 weeks. We coded and analyzed the responses to identify common themes, and areas of agreement and disagreement, leading to a new set of questions for the subsequent round. Our full survey instruments are available and raw data are available at https://osf.io/3qjs5

4.1 Delphi Process—ROUND 1

Survey Design and Distribution The ROUND 1 survey consisted solely of the following open-ended question:

What do you regard as the desirable qualities of commits?
In your answer, you may consider all of the aspects of a commit that you think are important, including but not limited to:
- the commit message
- the contents of the commit (files changed, lines changed)

For each of the qualities you identify, please explain why it is a desirable quality—i.e. what is the beneficial impact when the quality is present and/or the negative impact when it is absent? It would also be helpful if you could include examples to illustrate desirable qualities.

As shown in Table 1, we distributed the survey to 26 qualified professionals and received 16 responses.

Analysis Two raters conducted a bottom-up, inductive, semantic thematic analysis of the 16 ROUND 1 responses, following the procedures outlined by Braun and Clarke [4]. The raters used an iterative, consensus-building approach designed to ensure both rigor and adaptability as new themes emerged.

Responses were first separated into discrete segments for analysis, with each segment representing a distinct idea or statement about commit quality. Each rater then separately analyzed the responses of one of the panelists, yielding a preliminary set of coding categories organically from the data. These efforts yielded an initial coding scheme with four hierarchical dimensions: (1) the High Level Category (e.g., Observation, Quality), (2) the Commit Criteria Scope (e.g., Commit Contents, Commit Message), (3) Detailed Label (e.g. Definition, Rationale), and (4) Topic (e.g. "Commit message is concise", "Commit contents compile", etc.).

After coding an initial corpus of 118 responses from two panelists, raters achieved an initial inter-rater reliability (Cohen's Kappa) of $\kappa = 0.642$ for High Level Category, $\kappa = 0.905$ for Commit Criteria Scope, $\kappa = 0.644$ for Detailed Label, and $\kappa = 0.791$ for Topic.

The coders then proceeded through 11 rounds of iterative coding of randomly selected sets of 25–45 segments at a time.

In each round, the raters independently coded the segments using the existing coding categories, labeling unclassifiable segments as "Other" and proposing a new code. After completing each batch, the raters calculated inter-rater reliability. When any dimension fell below $\kappa \geq 0.8$, the raters resolved discrepancies and updated the category definitions.

After 11 rounds of iterative coding, the raters achieved inter-rater reliability values of $\kappa = 0.575$ for High Level Category, $\kappa = 0.801$ for Commit Criteria Scope, $\kappa = 0.729$ for Detailed Label ($\alpha = 0.731$), and $\kappa = 0.647$ for Topic. All 370 segments were subsequently reviewed through systematic convergence meetings where raters discussed and resolved every discrepancy, resulting in 100% final agreement across all segments and all four coding dimensions.

Following the completion of coding, all Topic codes were extracted and compiled for further analysis. The research team reviewed and grouped these topics

thematically, identifying conceptual overlaps and redundancies across categories. This process resulted in a final set of 30 distinct statements presented in Table 2 that capture the panel's proposed qualities of a high-quality commit. These 30 statements formed the basis for the ROUND 2 survey instrument.

Table 2. Round 2 results: 30 statements from Round 1 rated on Likert scale. The words "A high-quality commit" appeared at the start of each statement.

*SD=Strongly Disagree, D=Disagree, N=Neutral, A=Agree, SA=Strongly Agree

#	**A high-quality commit...**	SD	D	N	A	SA
1	... is focused on a single concern (e.g., part of a specific feature).	2	0	1	5	8
2	... changes a reasonable number of lines of code and files.	3	0	1	5	6
3	... contains code with a useful purpose.	3	0	2	7	3
4	... has a commit message that follows project-specific style guidelines.	1	0	1	4	10
5	... improves the project's software quality.	1	1	1	7	6
6	... tags (refers to) its associated ticket/issue.	2	4	2	2	6
7	... identifies other collaborators/co-authors in the description.	1	2	1	3	8
8	... does not break existing code and does not cause existing tests to fail.	1	0	1	6	8
9	... clearly describes what was changed.	2	1	4	3	6
10	... provides the context necessary to understand the change.	1	3	2	4	6
11	... includes a brief justification for why the change is being made.	2	3	2	5	2
12	... does not always need to have a clear commit message, especially when included in a pull request with a clear description of changes.	0	0	2	5	9
13	... contains a title that provides a clear idea of what the commit is about.	1	1	2	5	7
14	... has a message that correctly describes the contents of the commit.	1	1	4	9	1
15	... explains the high-level structure of the changes made.	1	0	1	5	9
16	... has a commit message that is understandable to other developers on the project.	1	5	3	3	4
17	... conveys the significance of the change that was made.	1	0	2	3	10
18	... is concise, but still useful.	1	1	3	6	5
19	... contains a message that describes the change at a higher level of abstraction than its code contents.	2	7	3	3	1
20	... includes a description of the solution strategy.	2	8	5	1	0
21	... includes a description of the tradeoffs that were considered in the solution.	1	1	3	9	2
22	... makes a change that was previously identified as necessary.	1	0	2	5	6
23	... is collegial.	2	0	8	4	2
24	... advances the team's understanding of a particular area of functionality.	1	1	2	6	6
25	... is accompanied by tests.	1	3	3	4	5
26	... contains code with helpful internal comments.	2	1	0	2	11
27	... contains code with well-chosen identifiers (variable names, method names, etc.).	2	1	2	5	6
28	... contains code that is free of anti-patterns that hinder maintainability.	1	0	2	4	9
29	... contains code that follows patterns of indentation, style, etc. that are consistent with the code base.	1	1	2	7	4
30	... does not need to contain information that will be conveyed in the Pull Request (such as links to issues/tickets, impact on the end user, pros/cons of alternative implementations, etc.).	1	1	2	7	4

4.2 Delphi Process—ROUND 2

Survey Design and Distribution In ROUND 2, we sought to determine the level of panel agreement or disagreement with the 30 statements derived from our analysis of the ROUND 1 Survey. Each statement was presented to the panel,

with a Likert-scale allowing panelists to indicate their level of agreement (see Figure 2 for an example). An additional response option, "Unable to answer (explain below)", was included to capture uncertainty or ambiguity.

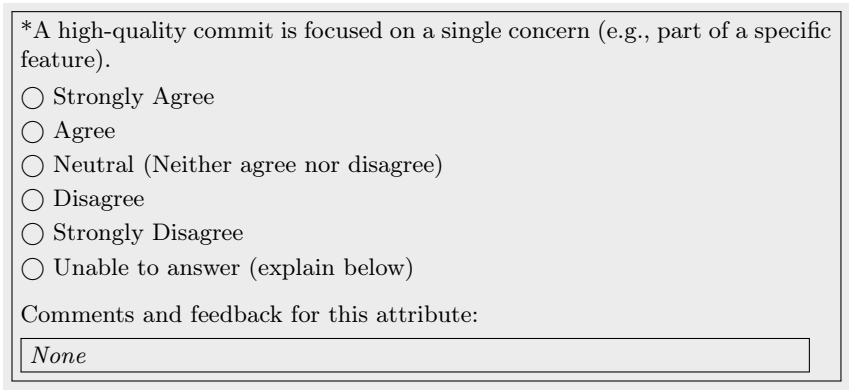

Fig. 2. Example of ROUND 2 Survey Question

In addition, the questionnaire included two broader open-ended questions:

1. **Commit Context**. *What aspects of a commit are possible to evaluate outside of its broader context? What aspects of a commit require further context to evaluate? Please elaborate on your answer below.*
2. **WIP vs. Archival Commits**. *Two commit practices that we've observed are Work in Progress (WIP) commits and Archival commits. WIP commits might not have all of the attributes that we are associating with "high quality commits", but at some stage (perhaps before making a pull request) the developer "squashes and merges" these into a smaller number of Archival commits with higher quality. What are your thoughts on these practices? Are the criteria for evaluating commits different for WIP commits vs. Archival commits? Please elaborate below.*

We included the question about commit context because we knew that the target rubric would eventually be used to evaluate commits in isolation as a means of evaluating student software engineering skills.

The question about WIP commits was motivated by the responses of a few ROUND 1 panelists who noted that their organizations' workflow is to squash and merge all commits in a pull request (PR) into single commit before the PR is reviewed. These panelists expressed skepticism about the value of reviewing individual commits outside of that context.

For ROUND 2, the same experts from ROUND 1 were invited, with one exception: one panelist opted out and was replaced by a new academic expert, resulting in the same total number of invitees across both rounds.

Analysis As shown in Table 2, none of the 30 proposed statements achieved 100% consensus in terms of receiving either "agree" or "strongly agree" ratings. Each of the two open ended questions were analyzed by two coders for four binary attributes; for both questions, the coders initially agreed on 57 of these 64 decisions $\kappa = 0.78$, and easily resolved each of these disagreements after discussion.

Analysis of sixteen responses to the question about commit context identified two emergent patterns, suggesting the need for greater precision in how experts interpret and evaluate statements about commit quality. Four panelists emphasized that individual commits are rarely evaluated in isolation and that the pull request (PR), rather than the commit, is the primary unit of review. Five participants indicated that they rely on PR descriptions, issue tracker links, and diff walkthroughs to assess intent, tradeoffs, and correctness. Four respondents expressed difficulty interpreting the question itself, reinforcing the view that commit evaluation is inherently contextual.

Six panelists did indicate that some surface-level and structural aspects of a commit can be meaningfully evaluated in isolation; they mentioned message clarity, adherence to team conventions, coding style, atomicity, size, and the absence of syntax errors or obvious defects. In contrast, deeper qualities such as correctness, necessity, security impact, alignment with project goals, and justification for design decisions were described as requiring broader contextual information, including project norms, team expectations, customer needs, and the surrounding PR discussion. Overall, responses suggested a strong consensus around the view that, while limited formal properties of commits can be assessed in isolation, meaningful evaluation of commit quality typically depends on contextual information beyond the commit itself.

Analysis of the open-ended responses about WIP versus archival commits revealed two themes. First, all sixteen panelists indicated that they distinguished between WIP commits, which support iterative and non-linear development, and archival commits, which are intended for long-term integration into the main branch. Twelve respondents indicated that their understanding of "high-quality commits" primarily applied to archival commits, especially those produced through squash-and-merge at the pull request stage. In contrast, WIP commits were often described as temporary, developer-focused artifacts that should not be held to the same standards. Second, three panelists explicitly stated that WIP commits should not be formally evaluated, emphasizing that strict criteria at this stage could hinder developer productivity. Others noted that evaluation standards might depend on visibility—for example, WIP commits in shared branches may warrant higher quality expectations. Additionally, three responses suggested that the pull request, rather than individual commits, is the appropriate unit for quality evaluation, highlighting the importance of broader context.

4.3 Delphi Process—Round 3

Survey Design and Distribution For Round 3 of the Delphi survey, we condensed the 30 statements from Round 2 into 28 candidate rubric items suitable for educational assessment by merging conceptually similar items.

Panelists were instructed to assess each rubric item in both *Work-in-Progress (WIP)* and *Archival* commit contexts. For each rubric item and applicable context, panelists selected one of three response options (Yes, No, or Neutral) to indicate whether the item should be included as an evaluative criterion.

Panelists were then instructed to evaluate each statement according to ideal professional expectations that would be desirable for students to learn, even if such practices may not be uniformly feasible in professional settings. Figure 3 shows one example question from the instrument; all 28 questions were presented in this way.

Additionally, the survey instrument incorporated structured feedback from the previous round by presenting panelists with aggregated responses and representative comments associated with each statement. We did this because an important principle of the Delphi process is that it isn't simply an opinion survey; it is intended to be an indirect moderated discussion among the expert panel. That is, the panelists need to hear from one another, but through an intermediary so that anonymity is preserved. In this round we took special care to ensure that that panelists not only were able to answer the questions, but were able to see what answers other panelists gave, along with their rationale.

The survey also included questions that allowed panelists to indicate whether a given attribute may be more appropriately assessed in another software artifact—for example, pull requests, issues, or code reviews. Our intent was to distinguish between disagreeing with the software engineering principles in the rubric and disagreeing with the commit as the right place to evaluate them.

We invited 20 experts to participate in the Round 3 survey and received 18 responses. To preserve the integrity of the Delphi process, invitations were limited to panelists who had responded to at least one of the two prior rounds.

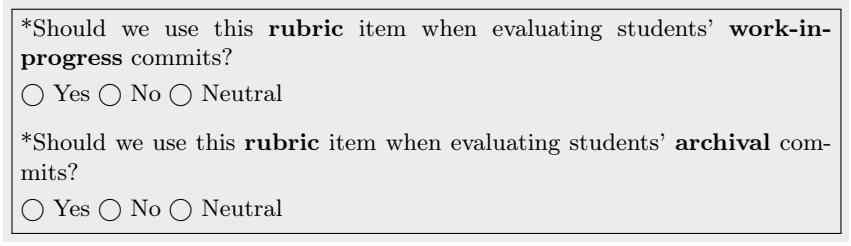

Fig. 3. Example of Round 3 Survey Question

Table 3. Round 3: Rubric items evaluated by the panel. Items with more than 80% "Yes" agreement are bolded. Y/N/– signifies Yes/No/Neutral

#	Statement	Work In Progress			Archival		
		Y	N	–	Y	N	–
1	This commit contains code that follows patterns of indentation, style, etc. that are consistent with the code base.	12	4	2	**16**	1	1
2	This commit has a commit message that follows project-specific style guidelines.	8	10	0	**15**	1	2
3	This commit does not hurt the project's software quality.	6	9	3	**15**	2	1
4	This commit clearly describes what was changed.	9	6	3	**16**	1	1
5	This commit has a message that correctly describes the contents of the commit.	6	9	3	**15**	0	3
6	This commit has a commit message that is understandable to other developers on the project.	9	9	0	**15**	3	0
7	This commit contains code that is free of anti-patterns that hinder maintainability.	7	9	2	**16**	1	1
8	The number of files and lines changed by this commit is small enough or straightforward enough that it can be effectively code reviewed.	10	7	1	14	3	1
9	This commit includes code/content that serves a clear and necessary purpose.	9	5	4	**15**	2	1
10	This commit does not break existing code and does not cause existing tests to fail.	7	9	2	**15**	3	0
11	This commit contains a title that provides a clear idea of what the commit is about.	10	5	3	**15**	3	0
12	This commit contains code with well-chosen identifiers (variable names, method names, etc.).	11	4	3	14	2	2
13	This commit is focused on a single concern (e.g., part of a specific feature, or addressing a single user goal).	9	8	1	13	4	1
14	This commit contains a message that describes the change at a higher level of abstraction than its code contents.	10	6	2	13	3	2
15	This commit contains whatever documentation and/or internal comments are needed to be maintainable.	7	6	5	12	3	3
16	This commit tags (refers to) its associated ticket/issue.	8	7	3	13	3	2
17	This commit is accompanied by tests.	4	8	6	11	2	5
18	This commit makes a change that was previously identified as necessary.	5	8	5	11	2	5
19	This commit message communicates in a collegial, inclusive and respectful manner.	8	8	2	11	3	4
20	This commit explains the high-level structure of the changes made.	6	9	3	12	4	2
21	This commit identifies other collaborators/co-authors in the description.	5	10	3	10	4	4
22	This commit includes a description of the trade-offs that were considered in the solution.	3	14	1	4	10	4
23	This commit provides the context necessary to understand the change.	4	11	3	9	5	4
24	This commit includes a brief justification for why the change is being made.	5	10	3	10	5	3
25	This commit describes the impact and purpose of the change for code reviewers.	3	12	3	5	7	6
26	This commit advances the team's understanding of a particular area of functionality.	2	12	4	5	7	6
27	This commit includes a description of the solution strategy.	2	14	2	4	13	1
28	A high-quality commit does not need information that will be in the Pull Request (links, pros/cons, etc.).	x	x	x	9	4	5

Analysis Table 3 presents a summary of panelists responses to the 28 ROUND 3 survey questions. For the responses to a given rubric item to achieve acceptable agreement, the research team established a threshold requiring at least 80% of panelists to select "Yes" for the rubric item in a given context. We note that when Diamond et al.[10] conducted a review of 72 Delphi studies, a percentage agreements was by far the most commonly used definition of consensus, used in 25 of the studies, and that the median threshold of agreement among these studies was 75%. Using this threshold, none of the rubric items achieved consensus for WIP commits. In contrast, 10 rubric items in Table 3 (see **bolded** items) met the consensus threshold when evaluated in the archival context.

We conducted a thematic analysis on the 10 items that met the 80% threshold in order to identify higher-level dimensions of commit quality. Our goal was to see if any of the rubric items could be consolidated. Through this process, four topics were identified: *code quality, code style, commit message/title, and commit purpose*. All researchers independently annotated each of the ten items using these topic categories and subsequently resolved the discrepancies through discussion until consensus was reached. The 10 rubric items were then distilled into the four synthesized rubric items shown in Table 4.

4.4 Delphi Process—ROUND 4

Survey Design and Distribution The ROUND 4 survey included the four items listed in Table 4. Panelists were asked to indicate whether each should be included in the final rubric ("Yes" or "No"). To support consensus-building, we presented statistics from the previous round indicating which rubric items met the 80% agreement threshold in the instrument. Figure 4 presents an example proposed rubric item with a yes/no question and associated open-ended question eliciting comments and concerns.

Proposed Rubric Item 1: This commit does not hurt the project's software quality, meaning that it (a) does not break existing code, (b) does not cause tests to fail, and (c) is free of known anti-patterns.

Should we include this item in our final list of rubric items for assessing **archival commits** in student software projects?

◯ Yes ◯ No

Do you have any **comments or concerns** about the **proposed rubric item 1**.

None

Fig. 4. Example of ROUND 4 Survey Question

In addition to the synthesized items, panelists were presented with the original set of 28 archival-context rubric items from the previous round. Panelists

Table 4. Proposed Rubric Items For Survey Round 4. Among the 20 participants invited in this round, 17 responded.

#	Topic	Proposed Rubric Item	% Agreement Include?	% Agreement Top 7
1	Code quality	This commit does not hurt the project's software quality, meaning that it (a) does not break existing code, (b) does not cause tests to fail, and (c) is free of known anti-patterns.	94.1%	94.1%
2	Code style	This commit follows good code style, meaning indentation and other style elements are consistent with the code base.	100.0%	94.1%
3	Message / Title	This commit has a title or commit message that clearly and correctly describes what was changed, is understandable to all developers on the project, and follows any project-specific guidelines that are in place.	100.0%	94.1%
4	Purpose	This commit contains only changes that serve a clear and necessary purpose.	94.1%	100.0%

were asked to finalize a set of seven rubric items by selecting up to three additional items from this list. They were permitted to exclude any of the proposed synthesized items. This approach enabled convergence on a concise rubric while allowing flexibility in determining the most essential criteria for evaluating archival commits. In Round 4 of the Delphi survey, 17 out of 20 invited panelists responded.

Analysis Table 4 summarizes panelists' agreement with the proposed commit rubric items. The "Include ?" column indicates the percentage of panelists that agreed that the item should be included in the rubric. The "Top 7" column indicates the percentage that indicated the item should be included in the top 7 most important rubric items. In each case, the decision was either unanimous (100%), or there was only a single dissenting vote ($16/17 = 94.1\%$). This high level of agreement suggests that the panel broadly views these rubric items as essential characteristics of high-quality commits, supporting their inclusion in the finalized rubric. (One panelist was inconsistent in their response for item 4, indicating "No" for Include? and "Yes" for "Top 7".)

In Round 4, panelists provided qualitative feedback indicating that two of the synthesized rubric items combined multiple evaluative dimensions, which could complicate their use in educational assessment. In particular, two panelists noted that the first proposed rubric item bundled several distinct criteria—breaking existing code, causing test failures, and introducing known anti-patterns—raising concerns about how partial satisfaction of these subcomponents should be scored and about the appropriateness of the item across different stages of a software engineering course. One panelist emphasized that expectations regarding regression testing and anti-pattern avoidance are often introduced progressively, suggesting that this criterion may be more suitable for later coursework rather than early projects. In response to this feedback, we decomposed the rubric item into three separate items addressing (1) whether a commit breaks existing code, (2) whether it causes tests to fail, and (3) whether it introduces

known anti-patterns, thus allowing instructors to apply these criteria selectively based on course context and learning objectives.

A similar concern arose for the third rubric item, which combined clarity, correctness, understandability, and adherence to project-specific conventions into a single criterion. To improve instructional flexibility, we similarly decomposed this item into three distinct rubric items focusing on (1) the clarity of the commit title, (2) adherence to project-specific message guidelines, and (3) the extent to which the commit message clearly and correctly describes the commit contents in a manner understandable to other developers. After decomposing the two rubric items, we finalized the rubric item in Table 5.

Table 5. Final Rubric Items On Commit Quality

#	Rubric Item
1	This commit does not hurt the project's software quality by breaking existing code.
2	This commit does not cause tests to fail.
3	This commit is free of known anti-patterns.
4	This commit follows good code style, with indentation and stylistic elements consistent with the codebase.
5	This commit contains a title that provides a clear idea of what the commit is about.
6	This commit has a commit message that follows project-specific style guidelines.
7	This commit has a message that clearly and correctly describes the contents of the commit, which is understandable to other developers on the project.
8	This commit contains only changes that serve a clear and necessary purpose.

In addition to rating the four proposed rubric items, panelists were invited to select additional items from the original set of 28 to identify three further items to form a final rubric of seven. Table 6 presents all rubric items that were selected by at least one panelist. Although some items received multiple selections, no additional item achieved majority agreement among the panel. Instead, selections were distributed across a range of items, indicating the absence of consensus for including any further rubric items beyond the four proposed ones.

5 Discussion

Our study produced a set of eight rubric criteria for evaluating archival commits that is grounded in empirical expert consensus. We now reflect on what these results contribute beyond prior work, and what they reveal about the nature of commit quality as a construct.

Empirical grounding for criteria identified in prior work. The most directly comparable prior work is that of Hundhausen et al. [16], whose rubric for commit quality included four criteria: *atomicity*, *accuracy*, *precision*, and *justification*. Our study both confirms and refines this rubric. Atomicity maps closely to our criterion that a commit contain only changes that serve a clear and necessary purpose. Accuracy and precision both find expression in our criteria

Table 6. Additional rubric items selected by ($n = 17$) panelists in Round 4.

Rubric Item	Selected By
This commit is accompanied by tests.	6
This commit is focused on a single concern (e.g., part of a specific feature, solving a single architectural problem, or addressing a single user goal).	6
This commit contains code with well-chosen identifiers (e.g., variable and method names).	4
The number of files and lines changed by this commit is small enough or straightforward enough to allow effective code review.	3
This commit identifies other collaborators or co-authors in the description.	2
This commit contains a message that describes the change at a higher level of abstraction than its code contents.	2
This commit's message tags or refers to its associated ticket or issue.	2
This commit message communicates in a collegial, inclusive, and respectful manner.	1
This commit provides the context necessary to understand the change.	1
This commit describes the impact and purpose of the change for code reviewers.	1

concerning the commit message: that it clearly describe what was changed, that it accurately reflect the commit's contents, and that it be understandable to other developers. However, where Hundhausen et al. derived their criteria from a subjective synthesis of industry discussions, our panelists arrived at a similar set of properties through the structured Delphi process, giving them a stronger empirical foundation. Notably, justification—the "why" of a commit—did not emerge as a consensus criterion in our study, even though it was featured in Hundhausen et al.'s rubric and in the simplified rubric proposed by Tian et al. [32]. This may reflect genuine disagreement among practitioners about whether the rationale for a change belongs in the commit message or in adjacent artifacts such as issues or pull requests—a question our panelists raised explicitly in their responses.

Commits as wholes, not just messages. Tian et al. [32] focused exclusively on commit messages, arguing that a good commit message should convey *what* was changed and *why*. Our rubric expands this view in two directions. First, two of our eight criteria concern the *contents* of a commit rather than its message: that it not hurt the project's software quality (including avoiding broken code, test failures, and known anti-patterns), and that it follow the code style conventions of the codebase. These properties cannot be evaluated from the message alone. Second, even within the commit message, our panel drew finer distinctions than the what/why framing of Tian et al., separating clarity of the commit title, correctness and completeness of the message description, and adherence to project-specific conventions into distinct evaluable criteria. This granularity may be especially valuable in educational settings, where students can benefit from targeted, specific feedback.

The WIP/archival distinction. One of the most substantive contributions of our study is the explicit distinction between work-in-progress (WIP) commits and archival commits. This distinction was not a premise of our study design but emerged organically from ROUND 1 responses, in which several panelists noted that their organizations routinely squash commits before merging pull requests. When we probed this theme in ROUND 2, it became clear that many panelists hold fundamentally different expectations for WIP commits vs. archival commits. In their view, WIP commits are ephemeral, developer-facing artifacts, whereas archival commits are permanent, team-facing records of change. This finding suggests that prior rubrics, including those of Hundhausen et al. and Tian et al., may have conflated two distinct categories of artifact. Applying archival-quality standards to WIP commits risks penalizing legitimate iterative development practices. Conversely, failing to distinguish the two may explain some of the skepticism our panelists expressed about evaluating individual commits at all. For educators, this finding recommends being explicit about which type of commit is under evaluation, and calibrating assessment expectations accordingly.

The role of context in commit evaluation. Our panelists consistently noted that many aspects of commit quality cannot be assessed from the commit in isolation. Correctness, necessity, security impact, and alignment with project goals all require contextual information—e.g. pull request descriptions, issue tracker links, team norms—that is unavailable when a commit is viewed on its own. This finding resonates with the broader assessment literature, which emphasizes that authentic evaluation requires sufficient context to mirror real-world professional practice [35,25]. It also has a practical implication for rubric design: the criteria that survived to our final rubric are, by and large, those that *can* be evaluated in relative isolation: code style, message clarity, message accuracy, and the absence of obvious defects. This is not a weakness of our rubric but rather a principled consequence of designing for the educational assessment context, where evaluators typically assess student commits without access to the full project history or team deliberation.

Implications for AI-generated commit messages. As noted in Section 2, there is a growing body of work on the automated generation of commit messages [34,15]. The consensus criteria we have established take on additional relevance in this context: if students are increasingly likely to encounter or produce AI-generated commit messages, they need an explicit framework for evaluating whether those messages meet professional expectations. Our rubric can serve precisely this purpose, giving students a principled basis for assessing generated messages rather than simply accepting them.

6 Threats to Validity

Geographic and temporal scope. All panelists were recruited from North America, so findings may not generalize globally. Our data were collected in

2024–2025; what practitioners regard as desirable commit practices may evolve, particularly as AI tools increasingly automate commit message generation.

Panel composition and attrition. Our panel size of 16–17 is consistent with Delphi norms [1], but participation across four voluntary rounds may have selected for a particular kind of respondent. We cannot rule out attrition bias in the views that shaped the final consensus.

Consensus threshold. We defined consensus as $\geq 80\%$ of panelists selecting "Yes." This is a methodological choice supported by prior practice [10]; nevertheless, a lower threshold would have admitted additional items. Readers should note that the items in Table 6 attracted meaningful support and may be more or less appropriate in specific instructional and research contexts.

Researcher influence. Despite using systematic thematic analysis procedures [4] and two independent raters, we were unable to achieve $\kappa \geq 0.8$ across all coding dimensions during independent coding. All disagreements were resolved through structured convergence meetings, but the codes cannot be guaranteed to be entirely free of researcher perspective.

Construct validity. Panelists may have interpreted "commit quality" differently depending on whether their organizational context emphasized WIP or archival commits, particularly in early rounds before we explicitly introduced that distinction.

7 Summary and Future Work

We applied the Delphi method with a panel of 16–17 diverse software engineering experts from industry and academia to establish empirically grounded consensus criteria for evaluating commits. Over four rounds, we derived a final rubric of four high-level criteria and eight detailed items, each endorsed by at least 16 of 17 panelists. The criteria address two dimensions: the *contents* of a commit (software quality and purposefulness of changes) and the *commit message* (clarity of title, accuracy of description, and adherence to conventions). A key emergent finding was that consensus applies specifically to *archival* commits; panelists did not endorse applying the same standards to WIP commits. Beyond the rubric itself, this study demonstrates the feasibility of using the Delphi process to derive empirically grounded rubrics for software engineering artifacts, guiding future similar work on issues, pull requests, code reviews, and other artifacts central to software engineering practice. Immediate future work includes classroom validation: specifically, whether multiple instructors can apply the rubric with $\kappa \geq 0.8$ interrater reliability, and whether instruction organized around these criteria produces measurable improvement in student commit quality. Longer-term directions include investigating automated assessment of rubric criteria, cross-cultural replication with non-North American panels, and extending the Delphi approach to other software engineering artifacts.

Acknowledgments. The authors would like to thank our Delphi panelists; while we would prefer to thank them publicly, Delphi practice is that they must remain anonymous. We also thank Brian Mulanda of Oregon State University for his contributions

to the analysis of Round 1 data. This work was supported in part by a grant from the National Science Foundation (USA), awards 2337269, 2337271, 2337270.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Adler, M., Ziglio, E.: Gazing Into the Oracle: The Delphi Method and Its Application to Social Policy and Public Health. Jessica Kingsley Publishers, London (1996)
2. Agrawal, K., Amreen, S., Mockus, A.: Commit quality in five high performance computing projects. In: Proceedings of the 2015 International Workshop on Software Engineering for High Performance Computing in Science. p. 24–29. SE4HPCS '15, IEEE Press (2015), https://doi.org/10.1109/SE4HPCS.2015.11
3. AlOmar, E.A., Mkaouer, M.W., Ouni, A.: Can refactoring be self-affirmed? an exploratory study on how developers document their refactoring activities in commit messages. In: Proceedings of the 3rd International Workshop on Refactoring. p. 51–58. IWOR '19, IEEE Press (2019). https://doi.org/10.1109/IWoR.2019.00017, https://doi.org/10.1109/IWoR.2019.00017
4. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative Research in Psychology **3**(2), 77–101 (2006). https://doi.org/10.1191/1478088706qp063oa, https://doi.org/10.1191/1478088706qp063oa
5. Chacon, S., Straub, B.: Pro Git. Apress, New York, 2 edn. (2014), https://git-scm.com/book/en/v2, freely available online
6. Clayton, M.J.: Delphi: a technique to harness expert opinion for critical decision-making tasks in education. Educational Psychology **17**(4), 373–386 (1997). https://doi.org/10.1080/0144341970170401, https://doi.org/10.1080/0144341970170401
7. Conventional Commits Contributors: Conventional commits 1.0.0. https://www.conventionalcommits.org/en/v1.0.0/ (2018), accessed: 2024-05-22
8. Craig, M., Conrad, P., Lynch, D., Lee, N., Anthony, L.: Listening to early career software developers. J. Comput. Sci. Coll. **33**(4), 138–149 (Apr 2018)
9. Dalkey, N., Helmer, O.: An experimental application of the Delphi method to the use of experts. Management Science **9**(3), 458–467 (1963). https://doi.org/10.1287/mnsc.9.3.458
10. Diamond, I.R., Grant, R.C., Feldman, B.M., Pencharz, P.B., Ling, S.C., Moore, A.M., Wales, P.W.: Defining consensus: A systematic review recommends methodologic criteria for reporting of delphi studies. Journal of Clinical Epidemiology **67**(4), 401–409 (2014). https://doi.org/10.1016/j.jclinepi.2013.12.002
11. Fauzan, R., Siahaan, D., Solekhah, M., Saputra, V.W., Bagaskara, A.E., Karimi, M.I.: A systematic literature review of student assessment framework in software engineering courses. Journal of Information Systems Engineering and Business Intelligence **9**(2), 264–275 (Nov 2023). https://doi.org/10.20473/jisebi.9.2.264-275, https://e-journal.unair.ac.id/JISEBI/article/view/45480
12. Hao, Q., Smith IV, D.H., Iriumi, N., Tsikerdekis, M., Ko, A.J.: A systematic investigation of replications in computing education research. ACM Trans. Comput. Educ. **19**(4) (Aug 2019). https://doi.org/10.1145/3345328, https://doi.org/10.1145/3345328

13. Haughney, K., Wakeman, S., Hart, L.: Quality of feedback in higher education: A review of literature. Education Sciences **10**(3) (2020). https://doi.org/10.3390/educsci10030060, https://www.mdpi.com/2227-7102/10/3/60
14. Hu, A., Liu, Q., Daniel, B.: Digital technologies in authentic assessment in higher education: A systematic literature review and narrative synthesis. SAGE open **15**(3), 21582440251357198 (2025)
15. Huang, Z., Huang, Y., Chen, X., Zhou, X., Yang, C., Zheng, Z.: An empirical study on learning-based techniques for explicit and implicit commit messages generation. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. p. 544–556. ASE '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3691620.3695025, https://doi.org/10.1145/3691620.3695025
16. Hundhausen, C., Carter, A., Conrad, P., Tariq, A., Adesope, O.: Evaluating commit, issue and product quality in team software development projects. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education. p. 108–114. SIGCSE '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3408877.3432362, https://doi.org/10.1145/3408877.3432362
17. Hutterer, P.: On commit messages. Who-t (December 2009), https://who-t.blogspot.com/2009/12/on-commit-messages.html, accessed: 2025-05-14
18. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering (v2.3). Tech. Rep. EBSE-2007-01, Technical Report. Keele University and Durham University Joint Report (2007), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.117.471
19. Kuang, H., Zhang, N., Gao, H., Zhou, X., Assuncao, W.K.G., Ma, X., Shao, D., Rong, G., Zhang, H.: Brevity is the soul of wit: Condensing code changes to improve commit message generation. In: Proceedings of the 16th International Conference on Internetware. p. 389–401. Internetware '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3755881.3755977, https://doi.org/10.1145/3755881.3755977
20. Li, F., Paxson, V.: A large-scale empirical study of security patches. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 2201–2215. CCS '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134072, https://doi.org/10.1145/3133956.3134072
21. Li, J., Faragó, D., Petrov, C., Ahmed, I.: Only diff is not enough: Generating commit messages leveraging reasoning and action of large language model. Proc. ACM Softw. Eng. **1**(FSE) (Jul 2024). https://doi.org/10.1145/3643760, https://doi.org/10.1145/3643760
22. O'Sullivan, B.: Mercurial: The Definitive Guide. O'Reilly Media, Sebastopol, CA (2009), http://hgbook.red-bean.com
23. Parker, M.C., Guzdial, M., Tew, A.E.: Uses, revisions, and the future of validated assessments in computing education: A case study of the fcs1 and scs1. In: Proceedings of the 17th ACM Conference on International Computing Education Research. p. 60–68. ICER 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3446871.3469744, https://doi.org/10.1145/3446871.3469744
24. Perforce Software, Inc.: Helix Core P4 Command Reference. Perforce Software, Inc. (2025), https://help.perforce.com/helix-core/server-apps/cmdref/current/Content/CmdRef/p4_change.html

25. Prickett, T., McChesney, I., Norling, E., Hayes, A., Chrysikos, A., Riddle, S., Davenport, J.H., Irons, A., Crick, T.: Towards a framework for mapping authentic assessment to competency in university computing education in the uk. In: 2025 IEEE Global Engineering Education Conference (EDUCON). pp. 1–10 (2025). https://doi.org/10.1109/EDUCON62633.2025.11016439
26. Reis, S., Abreu, R., Erdogmus, H., Păsăreanu, C.: Secom: towards a convention for security commit messages. In: Proceedings of the 19th International Conference on Mining Software Repositories. p. 764–765. MSR '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3524842.3528513, https://doi.org/10.1145/3524842.3528513
27. Reis, S., Abreu, R., Pasareanu, C.: Are security commit messages informative? not enough! In: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering. p. 196–199. EASE '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3593434.3593481, https://doi.org/10.1145/3593434.3593481
28. Sadler, D.R.: Indeterminacy in the use of preset criteria for assessment and grading. Assessment & Evaluation in Higher Education **34**(2), 159–179 (2009). https://doi.org/10.1080/02602930801956059, https://doi.org/10.1080/02602930801956059
29. Shing Cheng, J.Y., Adrian Pang, Z.J., Huai-En Lim, E.I., Hin Chan, S.W., Xian Sim, L.W., Koko, M., Cao, Q., Keoh, S.L.: Strategies and implications of peer assessment in software engineering education. In: 2024 IEEE Frontiers in Education Conference (FIE). pp. 1–7 (2024). https://doi.org/10.1109/FIE61694.2024.10892827
30. Tao, Y., Dang, Y., Xie, T., Zhang, D., Kim, S.: How do software engineers understand code changes? an exploratory study in industry. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. FSE '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2393596.2393656, https://doi.org/10.1145/2393596.2393656
31. Tian, Y., Zhang, Y., Stol, K.J., Jiang, L., Liu, H.: What makes a good commit message? (data and scripts). Zenodo (2021). https://doi.org/10.5281/zenodo.5763753, https://doi.org/10.5281/zenodo.5763753, dataset and replication package
32. Tian, Y., Zhang, Y., Stol, K.J., Jiang, L., Liu, H.: What makes a good commit message? In: Proceedings of the 44th International Conference on Software Engineering. p. 2389–2401. ICSE '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3510003.3510205, https://doi.org/10.1145/3510003.3510205
33. Wang, G., Sun, Z., Dong, J., Zhang, Y., Zhu, M., Liang, Q., Hao, D.: Is it hard to generate holistic commit message? ACM Trans. Softw. Eng. Methodol. **34**(2) (Jan 2025). https://doi.org/10.1145/3695996, https://doi.org/10.1145/3695996
34. Wu, Y., Li, Y., Yu, S.: Commit message generation via chatgpt: How far are we? In: Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering. p. 124–129. FORGE '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3650105.3652300, https://doi.org/10.1145/3650105.3652300
35. Zhan, Y., Boud, D., Du, Z.: Designing for authentic assessment: a scoping review. Higher Education (Nov 2025). https://doi.org/10.1007/s10734-025-01588-9, https://doi.org/10.1007/s10734-025-01588-9