
| RESEARCH ARTICLE

Beyond AI Detection: A Framework for Assessing Learning and Integrity in AI-Assisted Programming Education

Osemengbe Uddin¹✉, Eunice Uddin² and Amenawon Fadoju³

^{1,2,3} *Edo State University Iyamho, Nigeria*

Corresponding Author: Osemengbe Uddin, **E-mail:** uddin.osemengbe@edouniversity.edu

| ABSTRACT

Generative artificial intelligence (GenAI) tools such as ChatGPT and GitHub Copilot can now solve most introductory programming assignments, undermining the assumption that submitted code is evidence of individual learning. Many institutions have responded by adopting AI-detection software, yet detectors for both natural-language text and source code show limited accuracy, are easily evaded through paraphrasing or minor edits, and produce false positives that disproportionately affect non-native English writers. This paper argues that detection is an inadequate foundation for assessment in programming education and proposes an alternative. Using an integrative literature review of peer-reviewed research published between 2016 and 2026 on GenAI in computing education, AI-text and AI-code detection, and assessment validity, the study synthesises the evidence into a design framework named TRACE. TRACE assembles five complementary sources of evidence of learning: Transparency of declared AI use, Reasoning made visible through explanation and code walkthroughs, Authentic process traces from version control and development environments, Competence checkpoints conducted under secured conditions, and Evaluative judgement demonstrated by critiquing and repairing AI-generated code. The framework is operationalised through an assessment design matrix aligned with graded levels of permitted AI use and is illustrated through a worked application to an introductory programming assignment. The analysis shows that triangulating several modest, independently gathered sources of evidence provides a more valid, fair, and pedagogically productive basis for judgement than any single detector score. Implications for instructors, programme leaders, and policy makers are discussed, together with an agenda for empirical validation.

| KEYWORDS

Generative AI; Programming education; Academic integrity; Assessment validity; AI detection

| ARTICLE INFORMATION

ACCEPTED: 18 September 2026

PUBLISHED: 19 October 2026

1. Introduction

Generative artificial intelligence (GenAI) has altered the conditions under which programming is taught and assessed more rapidly than almost any previous educational technology. Large language models (LLMs) such as those behind ChatGPT and GitHub Copilot can generate syntactically correct and functionally adequate code from a short natural-language description, explain unfamiliar code, and repair errors on request. Early evaluations showed that these systems outperform most students on authentic introductory programming examination questions (Finnie-Ansley et al., 2022) and that more recent models pass the majority of assessments in complete higher-education programming courses (Savelka et al., 2023). Because the typical programming assignment asks students to produce working code against a specification, and because automated test suites reward exactly the kind of output that LLMs produce well, the submitted program has largely ceased to be reliable evidence that a student can program (Becker et al., 2023; Denny et al., 2024a).

Institutional responses have been varied. Surveys of instructors revealed short-term reactions ranging from attempts to ban AI tools to resigned acceptance that "resistance is futile", with longer-term intentions to redesign assessment (Lau & Guo, 2023).

Analyses of policies at leading universities show a general move away from outright prohibition towards guidance on disclosure, assessment redesign, and communication with students (Moorhouse et al., 2023; Chan, 2023). Nevertheless, a common operational response has been to rely on AI-detection software, either as stand-alone services or as features embedded in existing similarity-checking platforms. Detection is attractive because it appears to preserve existing assessment designs and promises a quick, apparently objective, answer to the question "did the student write this?" (Cotton et al., 2024).

The evidence base, however, does not support this confidence. Systematic testing of fourteen publicly available text detectors concluded that none was accurate or reliable, that all tended to classify machine text as human-written, and that simple obfuscation and paraphrasing further reduced detection (Weber-Wulff et al., 2023). Detectors misclassified the majority of essays written by non-native English speakers as machine-generated, raising serious equity concerns (Liang et al., 2023). Studies combining software with academic judgement found that commercial detection identified only part of the AI-generated content in deliberately constructed submissions and that prompt engineering reduced detection further (Perkins et al., 2024a). Detecting AI-generated source code is, if anything, more difficult: code is short, highly conventional, and constrained by language syntax, so human and machine solutions to the same small exercise often converge. Classifiers trained for specific courses can reach useful accuracy in their training context (Hoq et al., 2024; Nguyen et al., 2024), but their generalisation across tasks, languages, model versions, and light human editing remains uncertain. Watermarking schemes depend on provider cooperation and can be weakened by paraphrasing, and therefore do not offer a complete solution for education (Lancaster, 2023).

Beyond technical accuracy, detection asks the wrong question. In programming education, the purpose of assessment is to support and certify learning: the ability to decompose problems, reason about program behaviour, test and debug, and make defensible design decisions. Recent research shows that GenAI can either scaffold or short-circuit this learning. Novices given access to an AI code generator performed better on authoring tasks without loss on later manual tasks (Kazemitabaar et al., 2023a), yet many novices delegated entire tasks to the tool when given the opportunity (Kazemitabaar et al., 2023b), and students with weaker metacognitive skills were found to be harmed by GenAI, developing an illusion of competence that widened the gap with stronger peers (Prather et al., 2024). Whether a student used AI is therefore less important than whether the student learned, and assessment validity, rather than cheating detection, should be the organising concern (Dawson et al., 2024).

This paper adopts that position and is informed by three theoretical lenses. First, evidence-centred and sociocognitive views of measurement hold that an assessment is an argument linking claims about a learner to observable evidence elicited by tasks (Mislevy, 2018); if the link between product and competence is broken by GenAI, new kinds of evidence are required. Second, the concept of evaluative judgement, the capability to appraise the quality of one's own and others' work, has been proposed as a central learning outcome for a time when AI produces plausible but imperfect output (Bearman et al., 2024). Third, the learner-as-leader paradigm of AI in education (Ouyang & Jiao, 2021) frames AI as a tool that should augment, not replace, learner agency. Together these lenses suggest that integrity is best secured by designing assessments in which genuine learning is both necessary and visible.

Several partial responses already exist. The Artificial Intelligence Assessment Scale (AIAS) specifies graded levels of permitted AI use (Perkins et al., 2024b); national guidance recommends that assessment emphasise appropriate AI engagement and that trustworthy judgements draw on multiple, inclusive, and contextualised approaches (Lodge et al., 2023); computing educators have proposed new task types such as prompt problems (Denny et al., 2024b) and guardrailed AI assistants that support learning without handing over solutions (Liffiton et al., 2024); and authentic and oral assessments have long been discussed as integrity measures (Sotiriadou et al., 2020), although authenticity alone does not prevent outsourcing (Ellis et al., 2020). What is missing is an integrated, programming-specific framework that tells instructors which combination of evidence to gather, how to align it with the level of AI use they permit, and how to combine it into a defensible judgement without relying on detector scores.

The present study addresses this gap. Its research question is: *how can programming assessments be designed so that judgements about student learning and academic integrity remain valid and fair when students have access to GenAI, without depending on AI detection?* The contribution is threefold: (i) a synthesis of why detection-centred approaches are insufficient for programming education; (ii) the TRACE framework, which specifies five complementary sources of evidence of learning; and (iii) an assessment design matrix and worked example that operationalise TRACE across graded levels of permitted AI use. The novelty lies in shifting the unit of assessment from the code artefact to a triangulated body of evidence about the student's process, reasoning, and judgement, and in doing so specifically for the programming context, where artefacts are short, executable, and especially easy for GenAI to produce.

2. Materials & Methods

2.1 Research design

The study follows a qualitative, interpretivist paradigm and uses a conceptual research design in two linked phases: an integrative literature review followed by framework development and analytical validation. Integrative reviews are appropriate when a topic is emerging, the evidence is spread across disciplines, and the aim is to synthesise diverse sources into a new

conceptual model rather than to estimate a pooled effect (Torraco, 2016). This suited the present problem, which spans computing education, educational measurement, academic integrity, and machine-learning research on detection.

2.2 Search strategy and sources

Peer-reviewed literature was searched in Scopus, Web of Science, the ACM Digital Library, and IEEE Xplore. Search strings combined three concept groups: (a) "generative AI" OR "large language model" OR ChatGPT OR Copilot OR Codex; (b) "programming education" OR "computing education" OR CS1 OR "introductory programming"; and (c) "academic integrity" OR plagiarism OR "AI detection" OR assessment OR validity. Backward and forward citation searching was performed on key papers. The time window was limited to 2016–2026 in line with the conference requirement for recent sources, with the substantive GenAI literature concentrated from 2022 onwards.

2.3 Inclusion criteria and corpus

Sources were included if they (i) reported empirical findings on GenAI capabilities or use in programming education; (ii) evaluated the performance, bias, or robustness of AI-text or AI-code detection; or (iii) offered theoretical or policy contributions on assessment validity, evaluative judgement, or academic integrity in the presence of AI. Opinion pieces without an evidential or conceptual argument were excluded. Priority was given to journal articles and conference proceedings indexed in Scopus or Web of Science; a scholarly monograph and a national guidance report were retained where they made a unique conceptual or technical contribution. The final analytic corpus comprised the 31 sources listed in the reference list, of which 29 are peer-reviewed journal articles or indexed conference papers and two are secondary sources (under 10 per cent).

2.4 Analysis and framework development

Analysis proceeded in three steps. First, each source was coded for its claims about (a) what evidence of learning programming assessments can collect, (b) the threats GenAI poses to the validity of that evidence, and (c) proposed mitigations. Second, codes were grouped thematically into candidate evidence sources, and each candidate was retained only if it was supported by at least two independent sources and could be operationalised in ordinary university courses. Third, following the claim–evidence–task logic of evidence-centred design (Mislevy, 2018), the retained evidence sources were organised into a framework, mapped against graded levels of permitted AI use adapted from the AIAS (Perkins et al., 2024b), and specified in terms of task types, artefacts, and scoring guidance.

2.5 Analytical validation

Because the framework is new, validation at this stage is analytical rather than empirical. The framework was evaluated against five criteria derived from the literature: validity (does the evidence support claims about learning?), robustness (does the evidence remain informative when AI is used covertly?), fairness (does it avoid systematic bias, for example against second-language writers?), feasibility (can it be implemented with typical staff resources?), and pedagogical value (does gathering the evidence itself support learning?). Each criterion was applied to detection-centred assessment and to the TRACE framework, and the framework was then illustrated through a worked application to a representative introductory programming assignment. The worked application is a design illustration, not a report of collected student data.

3. Results and Discussions

3.1 Why detection is an insufficient foundation

The synthesis identified five recurrent limitations of detection-centred approaches, summarised in Table 1. The first is limited and unstable accuracy. Independent evaluations of text detectors report accuracy that is too low for high-stakes decisions and that degrades under paraphrasing, translation, or minor manual editing (Weber-Wulff et al., 2023; Perkins et al., 2024a). Code-specific classifiers can perform well on the data they were trained on (Hoq et al., 2024; Nguyen et al., 2024), but programming exercises in introductory courses have a small solution space, so correct human solutions and machine solutions often look alike; the shorter and more constrained the task, the weaker the signal.

The second limitation is bias and the resulting harm to students. False positives are not randomly distributed: detectors penalise writing with lower lexical variety, which characterises many second-language writers (Liang et al., 2023). In programming courses, novices who follow textbook idioms or instructor templates may similarly produce "typical" code. An accusation founded on a probabilistic score that the student cannot contest is procedurally unfair and damages trust (Cotton et al., 2024).

Beyond AI Detection: A Framework for Assessing Learning and Integrity in AI-Assisted Programming Education

The third limitation is evasion. Light refactoring, renaming, reordering, or asking the model to imitate a novice style are trivial for students and erode detection, a dynamic that reproduces the long-running arms race between plagiarism tools and obfuscation (Lancaster, 2023). The fourth is the moving target: detectors calibrated on one generation of models must be retrained as models change, and watermarking requires cooperation from every provider (Lancaster, 2023). The fifth, and most fundamental, is a validity mismatch. Even a perfect detector would answer only whether AI was used, not whether the student understands the code. In courses where some AI use is permitted or encouraged, the detection question becomes largely irrelevant, while the learning question remains (Dawson et al., 2024; Denny et al., 2024a).

Table 1. Limitations of detection-centred assessment in programming education identified in the review.

Limitation	Description	Key sources
Limited, unstable accuracy	Accuracy too low for high-stakes decisions; degrades with paraphrasing and editing; short, conventional code gives weak signal.	Weber-Wulff et al. (2023); Hoq et al. (2024)
Bias and harm	False positives concentrated among second-language writers and students producing typical, template-like work.	Liang et al. (2023); Cotton et al. (2024)
Evasion	Refactoring, renaming, or style imitation defeats detectors, reproducing the plagiarism arms race.	Lancaster (2023); Perkins et al. (2024a)
Moving target	Detectors must be retrained as models change; watermarking needs every provider's cooperation.	Lancaster (2023); Nguyen et al. (2024)
Validity mismatch	Detects AI use, not learning; irrelevant where AI use is permitted.	Dawson et al. (2024); Denny et al. (2024a)

These findings converge with policy analyses recommending that institutions avoid reliance on detectors for misconduct decisions and invest instead in assessment design (Moorhouse et al., 2023; Lodge et al., 2023). They also explain instructors' reported move towards redesign rather than policing (Lau & Guo, 2023). The implication is not that integrity no longer matters, but that it should be secured by the design of evidence rather than by the inspection of products.

3.2 The TRACE framework

The second result is the TRACE framework (Fig. 1), which organises evidence of learning in GenAI-assisted programming into five complementary sources. Each source responds to a specific weakness of product-only assessment, and each is supported by multiple strands of the reviewed literature. The framework's central principle is triangulation: confidence in a judgement about learning comes from the agreement of several independent, moderately reliable sources rather than from any single indicator.

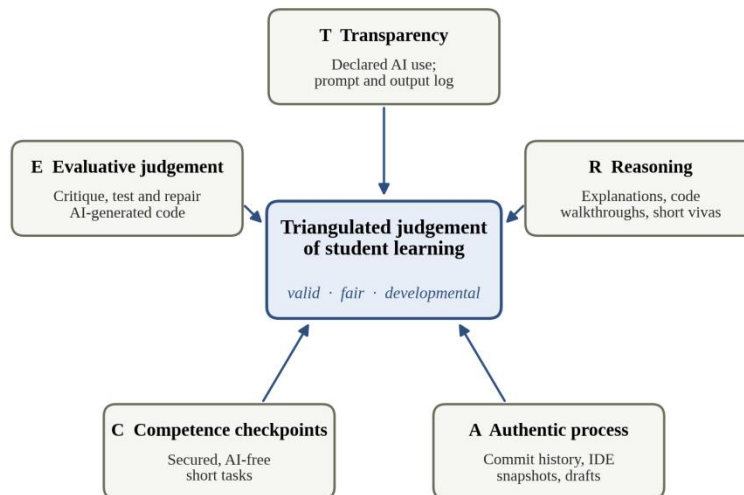


Figure 1. The TRACE framework: five complementary sources of evidence are triangulated into a judgement of student learning; their relative weight is set by the assessment design matrix according to the permitted level of AI use (source: author's analysis).

Transparency (T) requires students to declare whether and how they used AI, typically through a short structured statement and, where appropriate, an appendix of key prompts and responses. Transparency normalises legitimate use, makes expectations explicit, and gives instructors context for interpreting other evidence (Chan, 2023; Perkins et al., 2024b). It is not, on its own, a

guarantee of honesty, which is why it is combined with the other sources. Its primary value is pedagogical and cultural: it converts a hidden behaviour into a discussable one and lets instructors see how students actually use AI.

Reasoning (R) makes understanding visible through explanation. Students explain the design of their solution, trace its execution on given inputs, predict the effect of a proposed change, or justify a choice between alternatives. Reasoning can be elicited in writing, but its most robust form is a brief synchronous code walkthrough or viva of five to ten minutes. Interactive oral formats have been found to promote skill development while preserving integrity (Sotiriadou et al., 2020), and LLMs' ability to generate fluent code explanations (Denny et al., 2024a) means that written explanations alone are weaker evidence than live ones. Because a walkthrough focuses on the student's own submitted code, students who genuinely understand AI-assisted code are not penalised, while students who cannot explain code they submitted are identified on the basis of a transparent and contestable conversation rather than an opaque score.

Authentic process (A) collects traces of how the solution developed: version-control histories with meaningful incremental commits, periodic snapshots from the development environment, test-case evolution, and short reflective development logs. Programming process data are known to carry information about expertise; even keystroke-level typing patterns correlate with programming experience (Leinonen et al., 2016). Process evidence is not used here as a covert detector but as an openly assessed component: students know that a credible development history is part of the deliverable. A solution that appears in a single commit with no testing or debugging history does not prove misconduct, but it lowers confidence and indicates that other sources, particularly Reasoning, should be weighted more heavily for that student.

Competence checkpoints (C) are short, secured tasks completed without AI, such as in-class code-tracing questions, small writing or modification tasks in a locked environment, or parts of a supervised examination. They anchor the overall judgement by establishing a baseline of what the student can do unaided, consistent with the principle that some assessment must remain secure when certification depends on it (Lodge et al., 2023). Kazemitabaar et al. (2023a) used precisely this kind of unaided post-test to show that AI-supported practice need not reduce independent ability; checkpoints let instructors observe that outcome in their own courses. Checkpoints should be frequent and low-stakes rather than a single high-stakes event, which reduces anxiety and gives early feedback to students who are over-relying on AI.

Evaluative judgement (E) assesses the capability that GenAI makes more, not less, important: appraising generated code. Tasks present students with AI-generated solutions that contain subtle defects, inefficiencies, or security issues and ask them to test, critique, and repair them, or ask students to write prompts that elicit code satisfying a specification and to verify the result (Denny et al., 2024b). Such tasks align assessment with professional practice, develop the evaluative judgement that Bearman et al. (2024) identify as essential, and directly address the "illusion of competence" risk observed in weaker students (Prather et al., 2024).

Table 2 summarises, for each TRACE component, the claim it supports, typical evidence, its robustness to covert AI use, and its main cost. The analysis shows a useful complementarity. The components that are cheapest to collect at scale (Transparency and Authentic process) are the least robust to determined misuse, whereas the most robust components (Reasoning through live walkthroughs and Competence checkpoints) are the most resource-intensive. Combining them allows instructors to use inexpensive sources for all students and to target expensive sources where the cheaper evidence is thin or inconsistent.

Table 2. Components of the TRACE framework.

Component	Claim supported	Typical evidence	Robustness to covert AI use	Main cost
T Transparency	Student uses AI knowingly and responsibly	AI-use statement; prompt and output log	Low (self-report)	Very low
R Reasoning	Student understands the submitted code	Code walkthrough, trace and predict tasks, short viva	High (live)	Moderate to high
A Authentic process	Solution was developed iteratively by the student	Commit history, IDE snapshots, own tests, development log	Moderate	Low
C Competence checkpoints	Student can perform core skills unaided	Secured, short AI-free tasks	High	Moderate
E Evaluative judgement	Student can appraise, test and repair code	Critique and repair of AI-generated code; prompt problems	Moderate to high	Low to moderate

3.3 Combining evidence into a judgement

To make the triangulation logic explicit and auditable, the framework expresses the overall confidence in a claim about a student's learning, J , as a weighted combination of the ratings obtained from each evidence source:

$$J = \sum w_k s_k \quad \sum w_k = 1, \quad w_k \leq w_{k,x} \quad (1)$$

where $s_k \in [0, 1]$ is the rating of the evidence from source k (for example, a rubric score for the walkthrough), w_k is the weight the instructor assigns to that source for the assessment, and $w_{k,x}$ is a cap that prevents any single source from dominating; a cap of 0.4 is suggested. The cap formalises a central lesson of the detection literature: no single indicator, however sophisticated, should determine an outcome on its own. Weights are set in advance and published to students as part of the assessment brief, which supports transparency and procedural fairness. Equation (1) is a decision aid rather than an automated grading rule: when sources disagree markedly, for example a high product score with a low walkthrough score, the discrepancy itself is the trigger for a follow-up conversation, not an automatic penalty.

3.4 Assessment design matrix

The third result operationalises TRACE through an assessment design matrix (Table 3). Adapting the graded approach of the AIAS (Perkins et al., 2024b) to programming, four levels of permitted AI use are distinguished: L0, no AI; L1, AI for explanation and debugging help only; L2, AI as a collaborator for generating and refining code, with disclosure; and L3, AI-integrated work in which effective use of AI is itself part of the learning outcome. For each level the matrix indicates which TRACE components are essential and how weight should shift. As permitted AI use increases, the product carries less evidential weight and the process, reasoning, and judgement components carry more. At L0, competence checkpoints are the main guarantor, because the claim is about unaided ability. At L3, evaluative judgement and transparency become central, because the claim concerns the student's ability to direct, verify, and take responsibility for AI-generated work.

Table 3. Assessment design matrix: emphasis on TRACE components by level of permitted AI use.

Level of AI use	Product	T	R	A	C	E
L0 No AI	Medium	Low	Medium	Medium	High	Low
L1 AI for explanation and debugging only	Medium	Medium	Medium	Medium	High	Medium
L2 AI as collaborator, with disclosure	Low	High	High	Medium	Medium	High
L3 AI-integrated; AI use is a learning outcome	Low	High	High	Low	Medium	High

Note. High, Medium and Low indicate the relative weight recommended for each source in the overall judgement; Product denotes the submitted code.

The matrix also clarifies the role of guardrailed tools. Course-specific assistants that offer hints and explanations but refuse to output complete solutions (Liffiton et al., 2024) make L1 practical at scale, allowing students to obtain help while keeping the cognitive work of producing the solution with the student. Courses can deliberately sequence levels, beginning with L0 and L1 for foundational concepts and moving towards L2 and L3 as students develop the knowledge needed to judge AI output, an approach consistent with recommendations to treat AI literacy as a progressively developed outcome (Denny et al., 2024a; Prather et al., 2023).

3.5 Illustrative application to an introductory programming assignment

To illustrate how TRACE functions in practice, consider a representative second-half CS1 assignment in which students implement a small inventory-management program in Python using functions, lists, dictionaries, and file input/output, with a provided test suite. Under a detection-centred design, the assignment is set as take-home work, submissions are auto-graded, and code is passed through a detector; the resulting grade rests almost entirely on the product, which GenAI can produce in minutes (Savelka et al., 2023).

Under a TRACE design at level L2, the same programming task is retained, but the deliverables and weights change. Students submit (T) a one-paragraph AI-use statement with the prompts they found most useful; (A) a repository with at least five meaningful commits and their own additional test cases; and (E) a short report identifying and fixing two seeded defects in an AI-generated variant of one function. In the following lab, each student completes (C) a fifteen-minute AI-free checkpoint modifying a small function similar to one in the assignment, and a sample of students, together with any student whose evidence is inconsistent, completes (R) a five-minute walkthrough of their own code. Using Equation (1), the instructor might weight the product at 0.25, evaluative judgement at 0.25, reasoning at 0.20, the checkpoint at 0.20, and process and transparency together at 0.10.

Applying the five evaluation criteria shows the expected differences (Table 4). The TRACE design supports claims about understanding rather than only about output, remains informative when AI is used covertly because the checkpoint and

walkthrough are directly observed, reduces dependence on biased indicators, and turns integrity measures into learning activities: explaining code, writing tests, and critiquing AI output are themselves valued programming skills. The main cost is staff time for walkthroughs; sampling, peer-assisted walkthroughs in tutorials, and focusing follow-up on inconsistent evidence keep this manageable. For a cohort of 200 students, five-minute walkthroughs for a 25 per cent sample amount to approximately four hours of staff time per assignment, comparable to the time often spent investigating a small number of contested detector flags.

Table 4. Analytical comparison of the two designs for the illustrative CS1 assignment (source: author's analysis).

Criterion	Detection-centred design	TRACE design (L2)
Validity	Product score; weak link to understanding	Multiple sources linked to explicit claims about learning
Robustness	Low; easily evaded by editing	Higher; checkpoint and walkthrough directly observed
Fairness	Opaque, biased scores; hard to contest	Transparent, published weights; contestable evidence
Feasibility	Low marking cost; high cost of disputes	Moderate; walkthroughs sampled and targeted
Pedagogical value	Little; integrity measure is external to learning	High; testing, explaining and critiquing are learning activities

3.6 Implications, equity, and limitations

The findings have implications at three levels. For instructors, the main implication is to redesign rather than police: retain meaningful programming tasks but make reasoning, process, and judgement assessable components, and state the permitted AI level explicitly for every task. For programme leaders, the framework supports a programme-level view in which competence checkpoints are distributed across courses and AI levels are sequenced deliberately, rather than every course attempting to secure every assessment. For institutions and policy makers, the analysis supports guidance that discourages using detector output as evidence of misconduct and that resources assessment redesign instead (Chan, 2023; Moorhouse et al., 2023; Lodge et al., 2023).

Equity considerations run through the framework. By replacing opaque probabilistic scores with evidence that students can see, anticipate, and contest, TRACE reduces the risk of disproportionate harm to second-language writers and novices identified in the detection literature (Liang et al., 2023). However, oral components may disadvantage students with communication anxiety or certain disabilities, and process monitoring raises privacy concerns (Yan et al., 2024). The framework therefore recommends low-stakes, practised walkthroughs with reasonable adjustments, written alternatives where needed, and transparent, consent-based collection of process data limited to what is assessed.

The study has limitations. As a conceptual framework derived from an integrative review, TRACE has not yet been validated with student data; the weights and the cap in Equation (1) are reasoned defaults rather than empirically calibrated values; and the worked example illustrates design logic rather than measured outcomes. The literature is also evolving rapidly, and model capabilities will continue to change. These limitations define the next steps: comparative studies of TRACE and conventional designs on learning outcomes, the reliability of walkthrough rubrics, student perceptions of fairness, and staff workload across different institutional contexts, including resource-constrained settings.

4. Conclusion

This paper has argued that AI detection is an insufficient and potentially harmful foundation for assessing learning and integrity in programming education, because detectors are inaccurate, biased, easily evaded, quickly outdated, and, most importantly, answer whether AI was used rather than whether the student learned. In response, it proposed TRACE, a framework that triangulates five complementary sources of evidence: Transparency of AI use, Reasoning made visible, Authentic process traces, Competence checkpoints, and Evaluative judgement, combined through a capped weighting rule and aligned with graded levels of permitted AI use in an assessment design matrix.

The analysis indicates that such designs support more valid, fair, and pedagogically productive judgements than detection-centred approaches at a manageable cost, and that they prepare students for professional practice in which directing and verifying AI-generated code is a core skill. The framework offers instructors a practical basis for redesigning assignments, programme leaders a way to distribute secure assessment across a curriculum, and policy makers a rationale for shifting investment from detection to assessment design; future work should empirically validate the framework, calibrate its weights, and examine its effects on learning, fairness, and workload across diverse institutional contexts.

Conflict of Interest

The authors declare that there is no conflict of interest with any financial, personal, or other relationships with other people or organisations related to the material discussed in this manuscript.

References

- [1] Bearman, M., Tai, J., Dawson, P., Boud, D., & Ajjawi, R. (2024). Developing evaluative judgement for a time of generative artificial intelligence. *Assessment & Evaluation in Higher Education*, 49(6), 893–905. <https://doi.org/10.1080/02602938.2024.2335321>
- [2] Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard – or at least it used to be: Educational opportunities and challenges of AI code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 500–506). ACM. <https://doi.org/10.1145/3545945.3569759>
- [3] Chan, C. K. Y. (2023). A comprehensive AI policy education framework for university teaching and learning. *International Journal of Educational Technology in Higher Education*, 20, Article 38. <https://doi.org/10.1186/s41239-023-00408-3>
- [4] Cotton, D. R. E., Cotton, P. A., & Shipway, J. R. (2024). Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. *Innovations in Education and Teaching International*, 61(2), 228–239. <https://doi.org/10.1080/14703297.2023.2190148>
- [5] Dawson, P., Bearman, M., Dollinger, M., & Boud, D. (2024). Validity matters more than cheating. *Assessment & Evaluation in Higher Education*, 49(7), 1005–1016. <https://doi.org/10.1080/02602938.2024.2386662>
- [6] Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B. N., Santos, E. A., & Sarsa, S. (2024a). Computing education in the era of generative AI. *Communications of the ACM*, 67(2), 56–67. <https://doi.org/10.1145/3624720>
- [7] Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024b). Prompt problems: A new programming exercise for the generative AI era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 296–302). ACM. <https://doi.org/10.1145/3626252.3630909>
- [8] Ellis, C., van Haeringen, K., Harper, R., Bretag, T., Zucker, I., McBride, S., Rozenberg, P., Newton, P., & Saddiqui, S. (2020). Does authentic assessment assure academic integrity? Evidence from contract cheating data. *Higher Education Research & Development*, 39(3), 454–469. <https://doi.org/10.1080/07294360.2019.1680956>
- [9] Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference* (pp. 10–19). ACM. <https://doi.org/10.1145/3511861.3511863>
- [10] Hoq, M., Shi, Y., Leinonen, J., Babalola, D., Lynch, C., Price, T., & Akram, B. (2024). Detecting ChatGPT-generated code submissions in a CS1 course using machine learning models. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 526–532). ACM. <https://doi.org/10.1145/3626252.3630826>
- [11] Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023a). Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Article 455). ACM. <https://doi.org/10.1145/3544548.3580919>
- [12] Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D., & Grossman, T. (2023b). How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (Article 3). ACM. <https://doi.org/10.1145/3631802.3631806>
- [13] Lancaster, T. (2023). Artificial intelligence, text generation tools and ChatGPT – does digital watermarking offer a solution? *International Journal for Educational Integrity*, 19, Article 10. <https://doi.org/10.1007/s40979-023-00131-6>
- [14] Lau, S., & Guo, P. (2023). From "ban it till we understand it" to "resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V. 1* (pp. 106–121). ACM. <https://doi.org/10.1145/3568813.3600138>
- [15] Leinonen, J., Longi, K., Klami, A., & Vihavainen, A. (2016). Automatic inference of programming performance and experience from typing patterns. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 132–137). ACM. <https://doi.org/10.1145/2839509.2844612>
- [16] Liang, W., Yuksekogonul, M., Mao, Y., Wu, E., & Zou, J. (2023). GPT detectors are biased against non-native English writers. *Patterns*, 4(7), Article 100779. <https://doi.org/10.1016/j.patter.2023.100779>
- [17] Liffiton, M., Sheese, B. E., Savelka, J., & Denny, P. (2024). CodeHelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (Article 8). ACM. <https://doi.org/10.1145/3631802.3631830>
- [18] Lodge, J. M., Howard, S., Bearman, M., Dawson, P., & Associates. (2023). *Assessment reform for the age of artificial intelligence*. Tertiary Education Quality and Standards Agency.
- [19] Mislevy, R. J. (2018). *Sociocognitive foundations of educational measurement*. Routledge. <https://doi.org/10.4324/9781315871691>
- [20] Moorhouse, B. L., Yeo, M. A., & Wan, Y. (2023). Generative AI tools and assessment: Guidelines of the world's top-ranking universities. *Computers and Education Open*, 5, Article 100151. <https://doi.org/10.1016/j.caeo.2023.100151>
- [21] Nguyen, P. T., Di Rocco, J., Di Sipio, C., Rubei, R., Di Ruscio, D., & Di Penta, M. (2024). GPTSniffer: A CodeBERT-based classifier to detect source code written by ChatGPT. *Journal of Systems and Software*, 214, Article 112059. <https://doi.org/10.1016/j.jss.2024.112059>
- [22] Ouyang, F., & Jiao, P. (2021). Artificial intelligence in education: The three paradigms. *Computers and Education: Artificial Intelligence*, 2, Article 100020. <https://doi.org/10.1016/j.caeai.2021.100020>
- [23] Perkins, M., Roe, J., Postma, D., McGaughan, J., & Hickerson, D. (2024a). Detection of GPT-4 generated text in higher education: Combining academic judgement and software to identify generative AI tool misuse. *Journal of Academic Ethics*, 22(1), 89–113. <https://doi.org/10.1007/s10805-023-09492-6>
- [24] Perkins, M., Furze, L., Roe, J., & MacVaugh, J. (2024b). The Artificial Intelligence Assessment Scale (AIAS): A framework for ethical integration of generative AI in educational assessment. *Journal of University Teaching and Learning Practice*, 21(6). <https://doi.org/10.53761/q3azde36>

- [25] Prather, J., Denny, P., Leinonen, J., Becker, B. A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B. N., & Savelka, J. (2023). The robots are here: Navigating the generative AI revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 108–159). ACM. <https://doi.org/10.1145/3623762.3633499>
- [26] Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., Kimmel, B., Wright, J., & Briggs, B. (2024). The widening gap: The benefits and harms of generative AI for novice programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research V. 1* (pp. 469–486). ACM. <https://doi.org/10.1145/3632620.3671116>
- [27] Savelka, J., Agarwal, A., An, M., Bogart, C., & Sakr, M. (2023). Thrilled by your progress! Large language models (GPT-4) no longer struggle to pass assessments in higher education programming courses. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V. 1* (pp. 78–92). ACM. <https://doi.org/10.1145/3568813.3600142>
- [28] Sotiriadou, P., Logan, D., Daly, A., & Guest, R. (2020). The role of authentic assessment to preserve academic integrity and promote skill development and employability. *Studies in Higher Education*, 45(11), 2132–2148. <https://doi.org/10.1080/03075079.2019.1582015>
- [29] Torraco, R. J. (2016). Writing integrative literature reviews: Using the past and present to explore the future. *Human Resource Development Review*, 15(4), 404–428. <https://doi.org/10.1177/1534484316671606>
- [30] Weber-Wulff, D., Anohina-Naumeca, A., Bjelobaba, S., Foltýnek, T., Guerrero-Dib, J., Popoola, O., Šigut, P., & Waddington, L. (2023). Testing of detection tools for AI-generated text. *International Journal for Educational Integrity*, 19, Article 26. <https://doi.org/10.1007/s40979-023-00146-z>
- [31] Yan, L., Sha, L., Zhao, L., Li, Y., Martinez-Maldonado, R., Chen, G., Li, X., Jin, Y., & Gašević, D. (2024). Practical and ethical challenges of large language models in education: A systematic scoping review. *British Journal of Educational Technology*, 55(1), 90–112. <https://doi.org/10.1111/bjet.13370>