

Fluent, confident, wrong: Why LLMs' most underexploited pedagogical use is producing errors

M.Z. Naser^{a,b}

^a Artificial Intelligence Research Institute for Science and Engineering, Clemson University, Clemson, USA

^b School of Civil and Environmental Engineering & Earth Sciences, Clemson University, USA

ARTICLE INFO

Keywords:

Llms
Chatbots
Education
Stem
Critical thinking
Evaluative judgment
Error-based learning
Assessment validity

ABSTRACT

Artifact-based assessment assumes that the quality of a student's work reflects the student's underlying expertise. Large language models (LLMs) break this assumption by decoupling output quality from user knowledge, which renders artifact-only evaluation diagnostically unreliable. This shift also exposes a long-standing imbalance in engineering education, in which the profession has relied on both production and review, yet curricula and grading have emphasized production far more than the skills that make review effective. In engineering, review is the disciplined detection and correction of errors that can propagate into unsafe designs, incorrect calculations, or noncompliant decisions. Historically, teaching review at scale has been constrained by the scarcity of high-quality, realistic, and flawed artifacts. LLMs invert this constraint by generating plausible-but-wrong engineering work on demand, at controllable levels of subtlety and difficulty. Consequently, we argue that assessment validity can be strengthened by treating review performance in terms of error detection, diagnosis, and justification as a primary learning outcome rather than treating artifact production as the primary proxy for competence. We position review competence as a domain-specific form of critical thinking and evaluative judgment, anchored in the constraint structures that distinguish engineering verification from generic critique. Building on this perspective, we present a taxonomy of LLM-generated engineering errors, categorized by error type, severity, and detectability. We also provide templates for prompts and an accompanying LLM interface for generating on-demand, review-centered instructional activities.

1. Introduction

Artifact-based assessment in engineering education implies that the quality of a student's work reflects the student's underlying competence (Moskal et al., 2002). The same assumption has traditionally been reasonable because producing high-quality technical artifacts required the knowledge and skills that the assessment aimed to measure. For example, a student who submitted a correct structural analysis had, in the process, demonstrated understanding of load paths and equilibrium conditions. In other words, the coupling between output quality and producer competence was sufficiently aligned that grading the artifact served as a reliable indicator of the person who created it (Olds et al., 2005).

Large language models (LLMs) disrupt this coupling by enabling the generation of professional-quality technical content without matching user expertise (Kasneji et al., 2023; Mollick & Mollick, 2023). Students can now obtain polished calculations and plausible

E-mail address: mznaser@clemson.edu.

designs through simple prompting (independent of their own understanding of the underlying engineering principles) (Naser et al., 2024; Hostetter et al., 2024). The resulting artifacts may exhibit surface characteristics associated with competent work while reflecting little or no actual learning. Under these conditions, artifact-based assessment loses diagnostic validity because the construct it purports to measure in terms of student competence is no longer the primary determinant of the measured outcome (Messick, 1995; Kane, 2013).

Building on this loss of diagnostic validity, the resulting problem is not that students can submit work they did not author, but that the educational signal embedded in assessment becomes unreliable. As one would expect, in safety- and reliability-critical domains, the central professional question is rarely whether a document appears technically polished, but rather whether the underlying reasoning is correct under the stated assumptions, constraints, and code requirements (Mitcham, 2005; Passow, 2007). Subsequently, when instructional systems continue to reward artifact quality as the primary proxy for competence, they risk training students to optimize presentation and procedural completeness rather than developing the epistemic habits that prevent unnoticed error propagation. This is why the LLM disruption should be understood as a validity and competence issue rather than a narrow academic integrity problem.

Extending this principle to curricular design, the imbalance toward production has persisted partly because review competence is difficult to teach with conventional materials. This is because review requires repeated exposure to realistic, domain-authentic errors whose causes are interpretable and whose detectability can be deliberately calibrated. Yet, typical coursework rarely offers such controlled variation. Traditionally, instructor-authored flawed solutions are time-consuming to craft at scale, naturally occurring student mistakes are heterogeneous and inconsistent in quality, and "toy" errors often fail to resemble the subtle omissions and unjustified assumptions that characterize real engineering failures. Consequently, the scarcity of high-quality flawed artifacts has made it hard to provide deliberate practice in verification, critique, and justification, even though these are the very actions that distinguish competent engineering judgment.

Engineering practice, however, has always comprised producing technical work and verifying it. While LLMs commoditize production, they leave verification untouched. In turn, evaluating whether an analysis is correct or identifying where a design fails requires domain knowledge that prompting cannot circumvent. For instance, a student who identifies that a beam analysis neglects lateral-torsional buckling demonstrates competence through that diagnostic act in a way that submitting a polished report no longer demonstrates. Thus, review, as used in this paper, refers specifically to the disciplined engineering practice of verifying technical correctness against governing constraints, detecting errors, diagnosing their causes, and justifying evaluative judgments by reference to applicable principles, codes, or physical laws. This usage is narrower than generic peer feedback or critique (which may address communication quality, style, or persuasiveness without anchoring claims to falsifiable technical standards) and broader than mechanical checking (which may verify arithmetic without engaging the reasoning that produced it). Therefore, assessment becomes meaningful again when the performance being measured is the performance that matters.

This paper develops this argument and provides a framework for implementation. We present four contributions: 1) a validity argument for shifting assessment emphasis from production to review, 2) an operational definition of review competence as a measurable construct, 3) a taxonomy of engineering errors organized by type, severity, and detectability, and 4) implementation guidance encompassing task design, rubrics, and error-bank governance. Next, we build our argument on prior scholarship in expertise research, curriculum analysis, and error-based learning, and begin by establishing those conceptual foundations. We note at the outset that this paper is a conceptual and theoretical design contribution, in which we advance a validity argument, develop a construct definition and taxonomy, and propose an implementation architecture. Empirical validation of the framework's components constitutes a program of future work that the present paper is designed to enable.

2. Conceptual foundations and prior work

The argument that review should become a primary assessment focus rests on several independent lines of research. This section synthesizes some of the most relevant work to lay the groundwork for our central argument, which is developed in subsequent sections.

2.1. Expert-novice differences in error detection

Decades of cognitive research have documented systematic differences in how experts and novices perceive, represent, and evaluate domain-specific problems (Chi et al., 2014). At its core, experts do not simply know more facts, but they organize knowledge into richly interconnected schemas that support rapid pattern recognition and efficient retrieval. Thus, when an expert engineer examines a structural calculation, they perceive not isolated numbers but a configuration that either conforms to or violates expectations built through years of practice. This perceptual fluency enables experts to detect errors that novices overlook since novices are likely to lack the mental models against which deviations become salient. Experts can catch mistakes through multiple mechanisms, such as sanity checks (which compare results against order-of-magnitude expectations), constraint checks (which verify boundary conditions and physical plausibility), and dimensional reasoning (which ensures units propagate correctly through calculations), etc. (Ericsson et al., 1993). These checks are typically rapid/habitual for experts and often require little conscious effort, whereas novices must execute them laboriously, if at all.

The above implies that if error detection depends on pattern recognition developed through extensive experience, then students would benefit from structured opportunities to build those patterns. In parallel, merely producing correct solutions does not guarantee that students learn to recognize incorrect ones, because production and review engage partially distinct cognitive processes (Dunning et al., 2003). This means that a student who can solve a problem correctly may still fail to notice when a given solution contains a subtle flaw, particularly if the solution is superficially plausible. This also means that experts become skilled at detecting errors through

practice, not simply through practice in producing correct work. Hence, if error detection is a trainable capability, then the key question becomes whether engineering education explicitly targets it or assumes it emerges as a byproduct of producing artifacts.

2.2. Production vs. review in engineering curricula

As noted earlier, the engineering curricula have historically emphasized the production of technical deliverables (Sheppard et al., 2008). Accreditation standards reinforce this emphasis by specifying outcomes framed in terms of what students can create (e.g., the ability to apply engineering design, to formulate and solve problems, to develop and conduct experiments) (ABET, 2022). Then, assessment follows suit, with grades assigned primarily based on the submitted artifacts rather than on the quality of students' evaluative or diagnostic judgments. On the other hand, verification, peer review, and checking practices occupy a different status within this curricular architecture. These activities are often present but implicit, and often treated as elements of professional responsibility rather than as competencies warranting direct instruction and systematic assessment (Hess & Fore, 2018). For example, when students are asked to review each other's work, the reviews themselves are rarely graded with the same rigor as the original submissions (Patchan et al., 2018). When courses include verification steps, those steps may be framed as boxes to check rather than as opportunities for learning (Luxton-Reilly, 2009; Gibbs & Simpson, 2004). The result is a tacit hierarchy in which production is central, and review is peripheral.

Unfortunately, this curricular structure stands in tension with professional expectations. This is because practicing engineers spend substantial time evaluating, auditing, and justifying work produced by others or themselves at earlier stages (Trevelyan, 2010). These review activities are not ancillary to engineering practice; they are constitutive of it. Simply, errors that escape review can propagate into constructed systems with consequences ranging from costly rework to catastrophic failure. This misalignment suggests that even before LLMs, review competence was under-instructed. However, what changes now is not the importance of review, but the feasibility of teaching and assessing it at scale.

2.3. Error-based learning and productive failure

Educational research has long recognized that errors can serve as powerful learning tools when appropriately structured. The productive failure paradigm, developed by Kapur and colleagues (Kapur & Bielaczyc, 2012), demonstrates that students who struggle with problems before receiving instruction often develop a deeper conceptual understanding than students who receive instruction first. The struggle itself, provided it is followed by consolidation, activates prior knowledge, exposes gaps, and prepares students to encode new information more meaningfully. Therefore, engagement with flawed work extends this principle. For example, when students analyze artifacts that contain errors, they must compare the flawed approach with a correct one, articulate why the flaw is significant, and identify the constraints or principles that were violated. This comparative reasoning enhances conceptual discrimination and enables students to better distinguish between correct and incorrect applications of a concept (Siegler, 2002).

In fact, the error serves as a boundary marker that clarifies the scope and limitations of the concept. More specifically, mathematics instruction has utilized the comparison of correct and incorrect worked examples to enhance procedural flexibility (Rittle-Johnson & Star, 2007). Similarly, physics education has employed erroneous solutions to prompt explanation and repair (Adams et al., 2014). These applications share the common structure, wherein errors are deliberately introduced, students engage analytically with those errors, and feedback connects the error to underlying principles. Under these conditions, exposure to errors does not teach students to be wrong, but teaches them to recognize and reason about wrongness. From this perspective, one may infer that if errors are instructionally productive, the remaining obstacle has been logistical: producing large numbers of realistic, level-appropriate, flawed artifacts, an obstacle LLMs sharply reduce.

2.4. Code review pedagogy as a transfer template

Software engineering education offers a mature analog for what review-centered assessment might look like in broader engineering contexts. In this particular example, code review has become a standard component of professional software development, and educational adaptations have demonstrated that review skills can be explicitly taught and rigorously assessed (Hundhausen et al., 2011). Furthermore, the pedagogical infrastructure supporting code review is well-developed (i.e., rubrics specify the dimensions along which reviews are evaluated, including the accuracy of defect identification, the quality of explanatory comments, and the appropriateness of suggested fixes) (Wang, 2017). Other activities can accompany code review, such as calibration exercises, which allow students to compare their reviews against expert benchmarks and receive feedback on missed defects and false positives. Evidently, research has shown that students improve their code review skills through deliberate practice, and that review performance correlates with a deeper understanding of programming concepts (Indriasari et al., 2020; Strickroth, 2023).

It is of interest to this work to point out that several structural features of code review pedagogy can transfer readily to engineering review more broadly. These may include: 1) review tasks designed with known defect inventories (to enable criterion-referenced assessment), 2) checklists to guide students through systematic examination of common error categories, 3) severity weighting (to teach students to distinguish critical flaws from minor issues), and 4) false-positive management (to address the problem of over-flagging) (Porter et al., 1995). These mechanisms, along with others, can help provide a template for developing review-based assessments across various engineering disciplines.

2.5. Assessment validity and the construct-shift argument

Assessment validity concerns whether an assessment measures what it purports to measure. In the framework advanced by Messick and elaborated upon by Kane, validity is not a property of a test instrument in isolation, but instead of the interpretations and uses to which test scores are put (Messick, 1995; Kane, 2013). Thus, an assessment is valid to the extent that evidence and argument support the inferences drawn from performance. Artifact-based grading in engineering relies on an earlier-mentioned inferential chain, where the quality of the submitted artifact is taken as evidence of the student's underlying understanding and capability. However, this inference becomes unwarranted when other factors/tools (e.g., LLMs) can produce the same artifact quality without requiring the same level of competence (Haladyna & Downing, 2004). Worse, tool-generated artifacts often contain subtle errors that are masked by their confident presentation. This means that grading such artifacts as evidence of competence produces invalid inferences (Ji et al., 2023; Borji & Ai, 2023).

To overcome the above issue, one possible solution is to shift the assessed construct toward performances that LLMs cannot replicate without genuine human understanding. In particular, a review performance requires the reviewer to apply domain knowledge to an artifact produced by someone or something else. The reviewer cannot outsource this judgment to an LLM because the task is precisely to evaluate what an LLM or similar source might have produced. Assessment of review can therefore restore the coupling between performance and competence that artifact-based assessment has lost, provided that the review task is designed with features that make the coupling operative: a bounded artifact with a known-defect inventory, a requirement that the reviewer supply constraint-based justification (not merely identification), and scoring that accounts for both missed errors and false positives. Under these conditions, the validity advantage is relative rather than absolute — review-based assessment does not guarantee valid inference, but it preserves tighter construct relevance than artifact-only production under the same conditions of LLM access.

2.6. Review competence as domain-instantiated critical thinking

The construct of review competence developed here intersects established lines of work on thinking skills. For instance, critical thinking, as characterized across philosophical and educational psychology traditions, centrally involves evaluating claims against evidence, identifying unstated assumptions and providing reasoned justification for judgments (Ennis, 1989; Dwyer et al., 2014). Evaluative judgment, as theorized by Tai et al. (Tai et al., 2018), refers to the ability to assess the quality of one's own and others' work — a capability the present framework targets directly through structured engagement with flawed artifacts. Then, metacognition enters through the self-monitoring demands of review, which one can leverage to distinguish genuine errors from unusual-but-correct choices (the 'confounds' dimension in Section 6) requires the reviewer to calibrate confidence and recognize the limits of their own knowledge. Feedback literacy, as conceptualized by Carless and Boud (Carless & Boud, 2018), encompasses the ability to interpret, judge, and act upon evaluative information — processes that review-centered assessment elicits by design. Finally, the error-based learning tradition discussed in Section 2.3 connects to self-explanation research, wherein explaining why a solution is wrong activates deeper processing than confirming why a correct solution works (Chi et al., 1994). As one can see, these connections position review competence not as a narrow professional skill but as a domain-specific instantiation of evaluative reasoning, anchored in the constraint structures and verification practices that distinguish engineering from generic critical thinking.

3. The cost structure argument

The previous section established that review is a teachable capability, that engineering curricula have historically underemphasized it relative to production, and that LLMs disrupt the validity of artifact-based assessment. These foundations converge on a question of resource allocation; given finite instructional time and assessment capacity, what should educators prioritize? This section argues that the answer depends on cost structure, specifically on which capabilities are expensive to develop and which artifacts are expensive to generate. LLMs have inverted the pre-existing cost structure in ways that make review the rational focus for both instruction and assessment.

3.1. Instructional optimization and scarcity

In general, educational systems tend to optimize around scarcity. In this approach, curricula allocate time to what students cannot easily acquire elsewhere, and assessments focus on performances that differentiate levels of competence (Wiggins & Jay, 2005; Biggs & Tang, 2020). For example, when a capability is difficult to develop, instruction targets it, and when an artifact is difficult to produce, assessment values it. While this optimization is not always conscious or explicit, it shapes curricular priorities over time as programs adapt to the constraints they face.

Before LLMs (and the rise of generative AI), producing high-quality engineering artifacts was genuinely difficult. Simultaneously, producing realistic, flawed artifacts for instructional purposes was also difficult because an instructor had to create each flawed example manually to ensure that errors were plausible, appropriately calibrated to the student level, and varied enough to prevent pattern matching. One should also keep in mind that any instructor-calibrated examples that are contrived or obviously flawed often fail pedagogically because they do not prepare students to catch the subtle mistakes that matter in practice (Stark et al., 2011). As one can see, review-centered pedagogy faced a supply problem under these conditions. Effective review instruction requires a large volume of artifacts for students to evaluate, and those artifacts must contain errors realistic enough to teach diagnostic reasoning. Generating such examples at scale, with sufficient variety to prevent memorization and appropriate difficulty progression, exceeded what most

instructors could practically accomplish.

LLMs invert the identified cost structure. These systems can generate both correct and flawed artifacts with minimal marginal cost and on demand (Sarsa et al., 2022; Macneil et al., 2023). Such an inversion applies asymmetrically to production and review. As one can see, LLMs commoditize production by enabling anyone to generate plausible artifacts regardless of their own understanding. Thus, production skill becomes abundant and therefore less differentiating. Review skill, by contrast, remains scarce because explaining why the failure/errors matter requires domain knowledge that cannot be outsourced to an LLM. This implies that the reviewer must understand the principles well enough to recognize when they are violated. By extension, if assessment is meant to differentiate levels of competence, it should focus on the capability that remains differentiating.

3.2. Following the bottleneck

Building on the above, our cost-structure argument can be summarized as a principle, *assessment should follow the bottleneck*. In any system with multiple interdependent capabilities, the bottleneck is the capability that constrains overall performance and cannot be easily bypassed or outsourced (Goldratt, 1990). Before LLMs, production was a bottleneck because generating quality artifacts required substantial skill. After LLMs, production no longer becomes a bottleneck because skill requirements can be circumvented. Thus, review becomes the new bottleneck, the capability that determines whether an engineer can ensure quality in a world where generating content is trivially easy (see Fig. 1).

Now, if review is the bottleneck, then instruction should allocate a proportionally greater amount of time to developing review capabilities. This means that students need to practice detecting errors and not avoiding them in their own work. They also need feedback on their diagnostic judgments, and not only on their productions. Curricula designed around the old cost structure, in which production was central and review was peripheral, require recalibration to match the new reality. It must be stressed that the cost-structure argument does not imply that production should be abandoned. Thus, the evidentiary relationship between production and assessment has changed. Production remains instructionally valuable (but becomes less valid as an assessment mechanism), while review, previously difficult to assess at scale, becomes both feasible and necessary.

4. Review competence as a technical construct

The previous section identified review as the bottleneck. In practical terms, this means that review competence must be expressible in observable outputs that can be scored with repeatable rules. Accordingly, the construct must also be anchored to engineered task conditions (e.g., artifacts with controlled defect sets and specified evaluation criteria) so that performance can be interpreted as evidence of underlying verification knowledge. The construct was developed by triangulating three sources: the expert–novice literature on verification behaviors documented in Section 2.1, the pedagogical infrastructure of code review in software engineering (Section 2.4), and the author's professional experience conducting engineering review across structural, fire, and forensic engineering contexts. The resulting operations were organized into a dependency sequence, with each operation requires the output of the preceding one and refined to satisfy two criteria: each operation must produce a distinct observable trace, and the sequence as a whole must map onto the assessment-validity requirements established in Section 2.5. This section defines review competence as a structured sequence of cognitive performances, distinguishes genuine engineering verification from superficial critique, identifies what review measures that production no longer reliably measures under LLM access, and maps review competence to developmental progressions across course

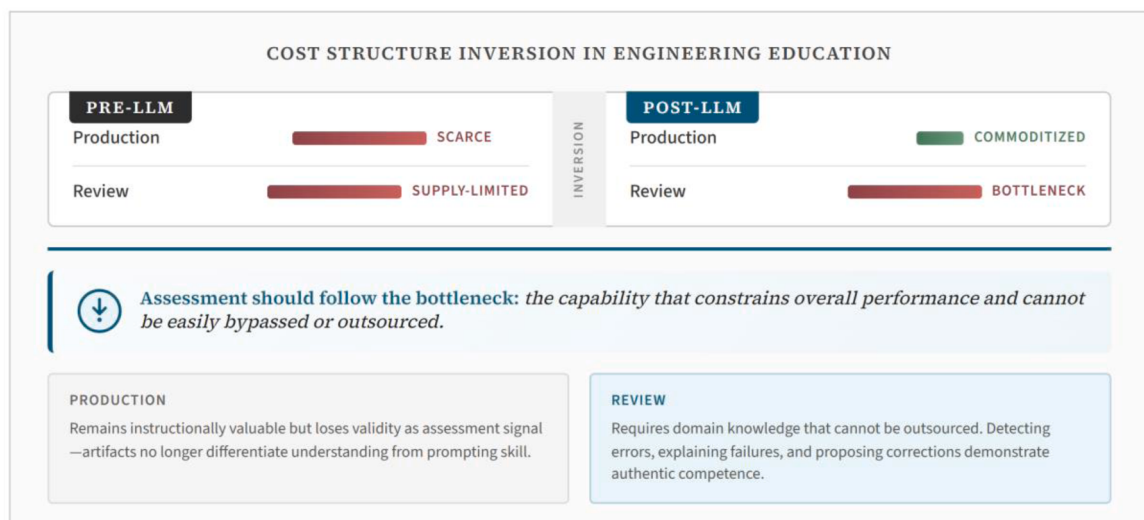


Fig. 1. The cost structure inversion.

levels (see Fig. 2).

4.1. Operational definition

Review competence, as conceptualized here, comprises five interconnected operations (see Table 1 and Fig. 2). These operations form a progression in which each step depends on and extends the previous one. Operationally, each step produces a distinct trace that can be independently assessed: detection identifies locations or stated anomalies; diagnosis determines error classification and localization; prioritization assesses a severity rating tied to functional consequences; justification articulates observations to violated principles/constraints/standards; and proposal recommends a corrective action that can be checked for technical adequacy.

To illustrate how the five operations and their scoring interact in practice, consider a structural analysis artifact in which the designer has applied a live-load reduction factor to a transfer girder that does not qualify for reduction under the applicable code. A student performing detection correctly flags the load value as inconsistent with the stated tributary configuration (scored as a hit; failure to flag would be a miss). At the diagnosis stage, the student identifies the error as a constraint/standard misapplication (specifically, incorrect application of the live-load reduction provision) and locates it in the demand calculation (scored against the intended error-type label and location). For prioritization, the student rates the error as high severity because the unconservative reduction directly affects member sizing for a primary gravity element (scored against the instructor key's consequence-weighted rating). The student's justification cites the specific code section governing live-load reduction eligibility and explains why the tributary configuration does not meet the minimum-area or multiple-occupancy requirement (scored for correct principle invocation and specificity). Finally, the student's proposal recommends removing the reduction factor and recalculating the demand using the unreduced live load (scored for whether the correction resolves the root cause). A student who correctly detects and diagnoses but assigns low severity would receive credit for the first two operations while losing credit on prioritization, demonstrating partial but incomplete review competence. For example, a student who detects an error but cannot diagnose it has incomplete review competence. Likewise, a student who diagnoses an error but cannot justify the diagnosis lacks the principled understanding that distinguishes expert from novice performance. Moreover, the assessment can be designed as criterion-referenced by using artifacts that contain a "known-defect inventory," where each embedded issue has a defined technical identity (type, location, consequence, and repair). This structure enables partial credit models that distinguish correct detection from incorrect diagnosis, or correct diagnosis with unsupported justification.

One must note that not all criticism constitutes legitimate review competence. In measurement terms, the boundary condition is whether a comment is anchored to a falsifiable claim about correctness. A statement counts as review evidence only when it identifies a checkable violation because only those claims can be evaluated against an external technical referent. In some instances, students may offer surface-level observations that do not engage the technical substance of an artifact. For instance, comments such as "this seems unclear" or "more detail is needed" may be valid feedback on communication, but they do not demonstrate engineering verification.

On the other hand, adequate examples can be seen in: 1) constraint checks (which verify that solutions satisfy boundary conditions, compatibility requirements, and problem specifications), 2) standards checks (which confirm that calculations follow applicable codes,

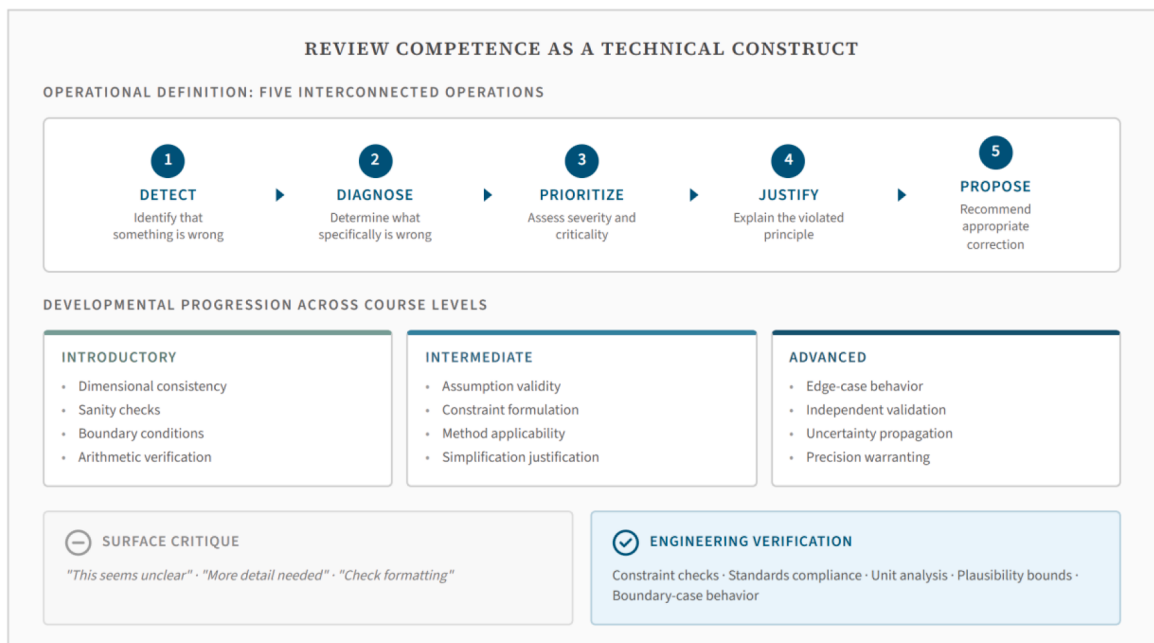


Fig. 2. Review competence as interconnected cognitive operations.

Table 1
Summary of operational definitions and assessment methods.

Operation	Definition	Assessment Method	Performance Method	Scoring
Detection	Identification that something is wrong; recognition that an artifact deviates from its intended state	Present artifacts with embedded errors and ask students to flag anomalies	Summarized with defensible metrics. For detection, performance can be represented as a hit/false-alarm profile (flagging true defects while avoiding spurious flags)	Should account for where actual errors go undetected, and false positives. A scoring scheme that counts only hits without penalizing false positives encourages over-flagging, which does not represent competent review. Balanced approaches might weight misses more heavily than false positives while still imposing some cost for incorrect flags, or might use signal detection metrics that account for both hit rates and false alarm rates.
Diagnosis	Determination of what specifically is wrong by locating the error and characterizing its nature	Require students to specify the nature and location of each error	Represented as localization accuracy and correct error-type labeling	Requires specifying acceptable characterizations for each embedded error and awarding credit based on how closely student responses match the intended diagnosis. Partial credit is appropriate when students identify a related but not precisely correct characterization.
Prioritization	Assessment of how severe the issue is; distinguishing critical failures from minor concerns	Severity rankings with justification	As agreement with consequence-weighted severity (e.g., code-noncompliance or critical errors weighted more than stylistic defects)	Expects establishing severity ratings for each embedded error and comparing student ratings against those benchmarks.
Justification	Articulation of why something is wrong by explaining the principle, constraint, or standard that the error violates	Evaluate whether students correctly reference the principles, constraints, or standards violated	With how the student correctly invokes the governing principle or standard with sufficient specificity to support the claim	Requires evaluating whether students reference the correct principles/constraints/standards that the error violates. Rubric descriptors might distinguish among justifications that are fully correct and complete, partially correct, vaguely relevant, or incorrect.
Proposal	Recommendation for how to address the issue by suggesting a correction that targets the underlying problem rather than masking symptoms	Evaluate the quality and appropriateness of suggested corrections	By whether the recommended correction resolves the underlying failure mechanism rather than masking symptoms	A defensible scoring design treats the proposal as a downstream remediation dimension that is either reported separately or gated on correct diagnosis and justification, so that students cannot obtain credit for plausible-sounding fixes that are not anchored to a correct identification of what failed and why it failed.

accepted methodologies/conventions), 3) unit checks (which ensure dimensional consistency), 4) plausibility checks (which compare results against expectations to catch gross errors), and 5) boundary-case (which checks examine whether solutions behave appropriately at limiting conditions) (Redish & Kuo, 2015). These verification modes, along with others, correspond to how experts actually evaluate engineering work, as established in Section 2.1. Assessment of review competence must therefore elicit and evaluate performances that demonstrate domain-specific verification, not generic critique (e.g., the expert engineer reviewing a structural design does not only ask whether the report is well-written; they also check whether deflections are within code limits, whether load combinations are correctly applied, etc.).

Conceptually, the five-operation model is best understood as a domain-specific instantiation of critical thinking and evaluative judgment rather than a standalone construct. Detection and diagnosis correspond to the analytical identification and characterization of anomalies — core operations in most critical thinking taxonomies. Prioritization operationalizes evaluative judgment by requiring discrimination among identified problems based on their consequences. Justification maps onto the warrant-giving that epistemologists and assessment theorists identify as the hallmark of reasoned evaluation. The proposal bridges evaluative and constructive reasoning, connecting the model to creative problem-solving by requiring corrections to address root causes rather than surface symptoms. This positioning clarifies that engineering review competence is not a separate faculty from critical thinking, but rather critical thinking exercised under the specific constraint structures — codes, physical laws, dimensional requirements — that define engineering practice.

4.2. What review measures that production does not

To further clarify the validity argument presented in Section 2.5, it is useful to specify what review performance reveals about a

student that production performance, under conditions of LLM access, may not. Specifically, review tasks can be constructed so that success depends on domain constraints: the student is given a fixed artifact and a bounded context (problem statement, assumptions, and standards excerpts, as appropriate), and is required to produce check-based evidence that can be evaluated independently of the artifact's rhetoric. This format preserves construct relevance because the score is driven by verified correctness judgments (what failed, why it failed, and what correction resolves it), not by the superficial plausibility that LLMs can readily supply.

In essence, review performance requires the student to bring independent knowledge to bear on an external artifact. The artifact may have been produced by another student, the instructor, or an LLM. However, in each case, the reviewer must evaluate it without access to the reasoning process that generated it. So, a student cannot prompt an LLM to "detect the error in this calculation" with any reliability, because the LLM may have produced the error in the first place and lacks the metacognitive capacity to recognize its own failures (Stechly et al., 2025). This does not mean that students cannot use LLMs at all during review (they may still generate candidate critiques, surface potentially relevant code provisions, or draft justification language). The key issue is that review competence is not merely the ability to generate candidate critiques, but rather the ability to adjudicate correctness under specific constraints. Accordingly, the validity advantage of review-based assessment is not absolute imperviousness to tool use, but rather that well-designed review tasks — those requiring constraint-based justification and independent verification — preserve a tighter coupling between performance and competence than artifact-only production tasks under the same conditions of LLM access. The assessment, therefore, requires the student to supply the constraint-based warrant for each claim, which is exactly the component that cannot be credited when it is echoed from a tool rather than generated from understanding.

Review performance also externalizes reasoning in ways that production may not. When a student produces an artifact, the reasoning behind it remains largely internal and unobservable. In contrast, when a student reviews an artifact, the reasoning is expected to be articulated. This articulation makes the student's understanding visible and assessable. In a way, the misconceptions that might remain hidden in production become exposed in review because the student must reason explicitly about correctness.

It can then be inferred that review competence develops along a trajectory that corresponds to increasing sophistication. This developmental mapping allows review competence to be taught and assessed progressively. At its basic level, the review is expected to focus on foundational checks that require essential knowledge, where students learn to verify dimensional consistency and to catch errors in which units do not propagate correctly. Students can also perform sanity checks by comparing numerical results with intuitive expectations to confirm that the execution is correct. These checks are accessible to novices because they rely on principles established early in the curriculum, yet they catch a significant class of errors that frequently appear in practice (Litzinger et al., 2011).

Then, at the intermediate level, review expands to encompass assumptions, constraints, and methodological appropriateness. Here, students evaluate whether the assumptions underlying a solution are satisfied by the problem context, whether constraints are correctly formulated, and whether the solution method is applicable to the problem class. The students assess whether the simplifications introduced into the analysis are justified or compromise validity. At this level, checks require a deeper understanding of domain knowledge because students must not only know how to apply a method but also when it is applicable.

At the advanced level, review grows in complexity to address edge cases, validation, and uncertainty. For example, students may examine whether solutions behave appropriately at limiting conditions where methods may break down. They may also consider whether results have been validated against independent checks, experimental data, or alternative analyses. The students are also expected to evaluate how uncertainty propagates and whether the reported precision is warranted. These checks approach the evaluative sophistication expected of practicing professionals and prepare students for the independent judgment that professional practice demands (Atman et al., 2007).

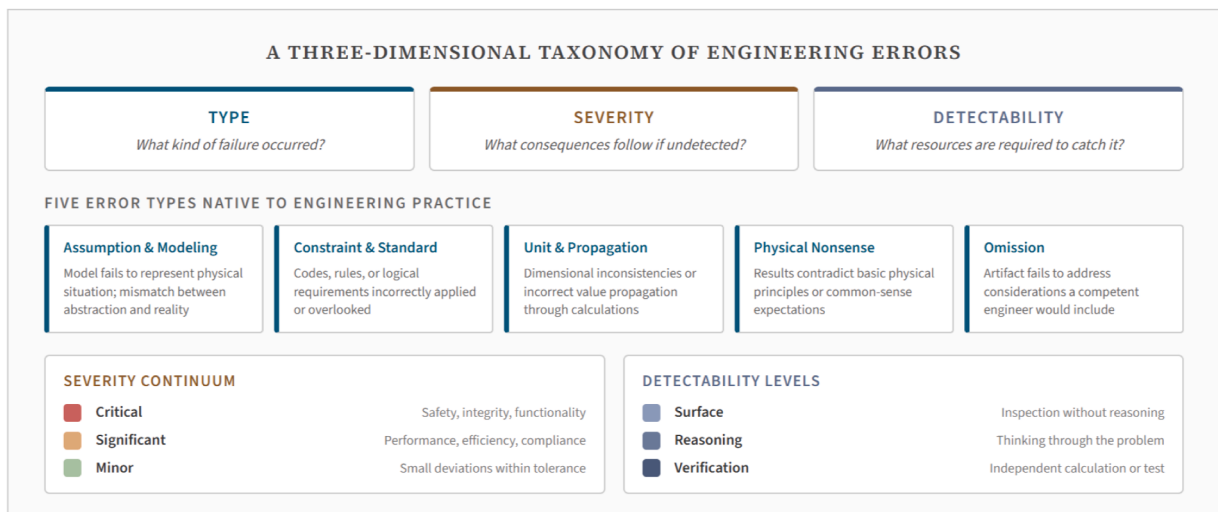


Fig. 3. Three-dimensional taxonomy for classifying engineering errors.

5. A taxonomy of engineering errors

The established definition of review competence in [Section 4](#) requires a corresponding specification of the error space against which review is exercised. Thus, a taxonomy provides the structure needed to ensure coverage and design assessment tasks that target specific competencies. This section presents a three-dimensional taxonomy that classifies engineering errors by type (specifies what kind of failure occurred), severity (what consequences follow if the failure goes undetected), and detectability (what cognitive and procedural resources are required to catch it) – see [Fig. 3](#). This taxonomy was synthesized from three established frameworks: Reason's latent-failure classification in human factors ([Redish & Kuo, 2015](#)), which supplies the conceptual distinction between error types and their systemic consequences; engineering code-compliance structures, which inherently separate the nature of a violation from its severity; and cognitive-load considerations from the expertise literature ([Section 2.1](#)), which motivate detectability as a dimension capturing the procedural and cognitive resources required for detection. The five error-type categories were abstracted from recurring failure patterns in structural, environmental, and general civil engineering practice, and cross-checked against forensic engineering case literature ([Blockley, 2013](#)) and common student misconceptions documented by the first author over multiple course iterations. This section also elaborates on five type categories native to engineering practice and provides illustrative examples demonstrating what competent review looks like across the taxonomy.

5.1. The three dimensions

The *type* dimension answers the question of what went wrong at a technical level. This builds on the fact that different error types violate different principles and require different domain knowledge to recognize. For example, a unit error involves dimensional inconsistency, whereas an assumption error involves a mismatch between model and reality. Thus, classifying errors by type enables targeted instruction, wherein students can practice detecting specific error classes until they develop reliable recognition patterns, then progress to mixed sets that require differentiation among types.

The *severity* dimension addresses the question of how significant a particular error is (since not all errors carry equal consequences). For instance, at the low end, minor errors result in slight deviations that do not affect the fitness for purpose. At the moderate level, significant errors compromise performance, efficiency, or compliance without creating immediate danger. Then, at the high end, critical errors threaten safety, structural integrity, or fundamental functionality. The classification is context-dependent, as an error that is minor in a preliminary feasibility study may be critical in final construction documents. Therefore, severity classification aligns with risk-based thinking prevalent in engineering practice, in which verification resources are allocated according to consequence ([Blockley, 2013](#)). One should note that teaching students to prioritize errors by severity prepares them to exercise professional judgment about where to focus verification effort when time and resources are constrained.

The *detectability* dimension answers the question of how hard the error is to catch. Some errors announce themselves through obvious cues, e.g., a result reported in wrong units, a negative value for an inherently positive quantity, or a magnitude grossly inconsistent with expectations. Other errors hide beneath plausible surfaces that require deep reasoning or independent verification to uncover ([Reason, 1993](#)). As one can see, detectability can be partitioned into three levels: 1) surface-detectable errors can be caught through inspection of the artifact itself (e.g., obvious unit mismatches, or values that violate common sense on their face), 2) reasoning-detectable errors necessitate the reviewer to think through the problem to check whether assumptions/constraints hold, or the solution method applies, and 3) verification-detectable errors, which require the reviewer to perform independent calculations, consult external references, or conduct tests to determine whether the result is correct. This progression corresponds roughly to an increase in cognitive and procedural costs, and can help inform both task design and rubric calibration.

It should be noted that the three dimensions and their levels are proposed as an analytically useful organizing structure, not as an empirically validated classification system. Validation would require structured procedures (e.g., independent expert classification of sample artifacts to assess inter-rater reliability, comparison with established error typologies in adjacent fields, and classroom piloting to test whether the categories produce consistent and instructionally meaningful distinctions). These steps constitute a necessary program of future work that the present framework is designed to support.

5.2. Error types native to engineering practice

Engineering errors often cluster in five types, namely 1) assumption and modeling errors, 2) constraint and standard misapplication, 3) unit and propagation errors, 4) physical nonsense and sanity violations, and 5) omission errors. Each corresponds to distinct failure modes that practicing engineers encounter and that engineering education should prepare students to recognize.

Assumption and modeling errors occur when the mathematical or computational model fails to represent the physical situation it is meant to capture. These errors arise at the interface between reality and idealization. For example, a structural analysis that treats a connection as pinned when it actually provides moment resistance contains an assumption error. Another example can be seen in the case of a thermal analysis that neglects convective losses to ambient air. Herein, detecting assumption errors requires the reviewer to understand both the model's internal logic and the physical context to which it is applied, then to assess whether the mapping between them is valid ([Oreskes et al., 1994](#)). In addition, the reviewer considers whether the method's unstated assumptions are appropriate and assesses whether simplifications introduce acceptable or unacceptable deviations from reality. For example, a reviewer examining a beam analysis should verify that the support conditions match the idealized boundary conditions and that the loading is accurately represented.

Constraint and standard misapplication errors occur when applicable rules, codes, or logical requirements are incorrectly applied

(or entirely overlooked). Errors in this category include using outdated code provisions, applying factors incorrectly, misinterpreting compliance thresholds, and violating logical constraints inherent in the problem formulation. For instance, a water treatment calculation that compares effluent concentration to the wrong regulatory limit misapplies a constraint. Overcoming these errors requires familiarity with applicable standards and the ability to verify that the artifact conforms to them. To mitigate this type of error, the reviewer verifies that the correct version of the code is referenced and applied as specified, and that compliance demonstrations address the relevant criteria. This review mode is particularly amenable to checklist-based approaches, as noted in [Section 2.4](#), in the context of code review pedagogy.

Unit and propagation errors occur when dimensional inconsistencies arise or when quantitative values are incorrectly carried through calculations. Despite their elementary nature, these errors are remarkably common and can yield results that are orders of magnitude off ([Redish & Kuo, 2015](#)). A competent review of unit and propagation errors involves a systematic tracing, where the reviewer follows the dimensional analysis through each step and/or spot-check intermediate values to verify that inputs propagate as expected. The reviewer may also apply order-of-magnitude estimation to confirm that the final results are plausible given the inputs. Such checks are foundational and correspond to the introductory level in the developmental progression described in [Section 4.4](#).

Physical nonsense and sanity violations occur when results contradict basic physical principles or common-sense expectations. Some examples that belong to this family of errors include: a structural analysis that reports tensile stress in a material subjected only to compression (i.e., which contains physical nonsense), and a traffic model that predicts negative vehicle counts (which violates an obvious constraint). These errors are often highly detectable because they produce results that a knowledgeable reviewer recognizes as impossible, yet they can escape notice when reviewers focus narrowly on procedural correctness without stepping back to assess plausibility. In response, a capable reviewer is expected to maintain awareness of what answers should look like by asking whether the result makes physical sense.

Finally, omission errors occur when the artifact fails to address something it should have addressed. Unlike the previous types, which involve incorrect content, omission errors involve missing content. For example, a structural design that checks strength but neglects serviceability criteria contains an omission. Similarly, a risk assessment that considers primary hazards but ignores secondary or cascading effects leaves critical scenarios unexamined. It can be inferred that omission errors are particularly insidious because the artifact may appear complete and correct within its stated scope while failing to address considerations that a competent engineer would recognize as essential. Tackling this type of error can be cognitively demanding since it requires generating expectations rather than evaluating what is given (i.e., the reviewer must ask not only "Is what is here correct?" but also "Is everything that should be here actually here?").

A classification principle is needed for boundary cases where an error could plausibly belong to more than one type category. For example, the omission of a required assumption check could be classified as an assumption/modeling error (because the underlying issue is model–reality mismatch) or as an omission error (because the artifact fails to address a required consideration). Similarly, a misapplied standard may interact with a modeling choice when the standard's applicability depends on assumptions about the structural system. In such cases, we recommend classifying by the primary cognitive operation the reviewer must perform to detect the error: if detection requires recognizing that a stated assumption is inappropriate, the error is an assumption/modeling error regardless of whether it also involves a missing check; if detection requires generating an expectation about what should have been addressed, the error is an omission. These categories are thus best understood as analytically useful partitions organized around distinct verification demands, not as mutually exclusive natural kinds, and instructors should expect that some items will involve reasoned judgment about

Table 2
Examples of error types.

Error type	What it is & competent review checks	Micro-examples	Classification
Assumption & modeling	Model–reality mismatch; reviewer validates boundary conditions, mechanisms included/omitted, and whether parameters/assumptions actually apply in the physical context (including unstated assumptions).	Foundation design adopts soil bearing capacity from an adjacent-site geotech report, but the subject site contains fill absent at the reference location; artifact is internally consistent, but applicability is wrong.	<i>Severity:</i> Moderate <i>Detectability:</i> Reasoning-level
Constraint & standard misapplication	Incorrect/omitted regulatory thresholds, or logical constraints; reviewer confirms correct code version/provision selection, and correct compliance target (often checklist-friendly).	Seismic design utilizes a response modification factor for the incorrect structural system, which reduces design forces below code requirements. Detection requires verifying that the factor selection aligns with code tables and system classification.	<i>Severity:</i> High <i>Detectability:</i> Verification-level
Unit & propagation	Dimensional inconsistency or incorrect carry-through of quantities; reviewer traces units, spot-checks intermediate values, and uses order-of-magnitude estimates to confirm plausibility.	Dosage calculation reports milligrams when the input was in grams, creating a 1000× error; a dimensional check catches it immediately.	<i>Severity:</i> High <i>Detectability:</i> Surface-level
Physical nonsense & sanity violations	Output contradicts physical principles or common-sense constraints; reviewer steps back from procedure to ask whether sign, magnitude, direction, and qualitative behavior are plausible.	Cost estimate totals to −\$50,000 for a residential renovation; the sign error is obvious once the reviewer interprets what the number implies.	<i>Severity:</i> Low <i>Detectability:</i> Surface-level
Omission	Missing required checks/considerations (not "wrong content" but absent content); reviewer must generate expectations, including secondary requirements and separate compliance categories.	Stormwater plan addresses peak flow attenuation but omits mandated water-quality treatment; detection requires knowing water quality is a separate ordinance requirement and noticing it is unaddressed.	<i>Severity:</i> Moderate <i>Detectability:</i> Reasoning-level

primary classification. For completeness, [Table 2](#) lists additional examples to further clarify each of the discussed error types.

6. Building and governing a scalable error bank

The taxonomy presented in [Section 5](#) provides a classification system; however, instructors require a supply of engineering work products that contain embedded errors, suitable for student evaluation. This section identifies sources of error, articulates principles to ensure that flawed artifacts remain realistic, describes calibration mechanisms to control difficulty, and specifies ethical guardrails to prevent instructional materials from enabling harm.

6.1. Plausibility and sources for error identification

Errors in instructional artifacts must be plausible in the sense that a reasonable author could have made them without obvious carelessness. Consequently, errors should be embedded in otherwise correct, professional-quality work, ideally as a few well-placed defects rather than a scatter of apparent problems. This structure mirrors practice, where review occurs against a background of mostly competent work, and it increases instructional value by forcing students to separate local failure from global competence rather than treating the artifact as a "gotcha" document ([Große & Renkl, 2007](#)). In addition, errors should be consistent with a coherent knowledge state, wherein a misapplied formula should reflect a recognizable misconception (or a plausible confusion between similar provisions) so that the artifact "makes sense" even in its incorrectness.

Plausible errors can be drawn from contexts where such mistakes actually occur to anchor the authenticity of the error bank in real failure modes. Some easily accessible sources may include professional standards and codes, which often provide commentary explaining the misuse a provision is meant to prevent; design guides frequently warn against specific misapplications; and published errata or addenda documents that document mistakes that survived formal review. In parallel, textbooks and instructional materials often include worked examples, and older solutions manuals may contain errors that are later corrected. Additionally, instructors' accumulated knowledge of recurring student misconceptions can be translated into coherent, plausible error patterns ([Brown, 2014](#)). Furthermore, practice-facing artifacts help ensure that the bank accurately reflects how engineering reviews are actually performed. For example, practitioner checklists and quality-control protocols reveal what experienced engineers actively inspect, and by doing so, effectively define which error families matter enough to institutionalize. Forensic engineering literature then supplies failure mechanisms tied to identifiable errors in design, analysis, or construction, which can be abstracted into instructionally manageable scenarios without recreating full case histories ([Delatte, 2015](#)).

A scalable error bank should support calibrated difficulty to remain comparable across assessment instances. One control dimension is subtlety, defined herein as the degree to which the error announces itself directly. A second dimension is density, as a single isolated defect is generally easier to diagnose than multiple interacting errors whose effects compound or obscure one another. A third dimension is confounds, where an artifact includes unusual but correct choices that can be mistaken for mistakes, thereby raising difficulty by penalizing indiscriminate over-flagging and rewarding discriminating judgment. Thus, errors and artifacts can be tagged with difficulty attributes (subtlety, density, confounds, and error type), and assessment forms can be assembled by sampling to match targeted profiles.

6.2. Ethics and safety guardrails

Errors should be situated in benign or generic contexts that preserve pedagogical value without producing realistic templates for misuse. For example, a generic beam check can teach the same review habit as a critical infrastructure example, but with lower downstream risk if the document circulates. In addition, error types should be screened for reversibility risk since, in some domains, even an intentionally incorrect procedure can function as a partial recipe that becomes actionable once "fixed," so such topics should be avoided or handled only under exceptional controls. Building on this risk framing, standard measures (e.g., restricted access to authorized instructors, item rotation, etc.) should be applied to error banks in the same way they are applied to conventional test banks ([Lane et al., 2016](#)).

7. LLM-assisted authoring workflow for scalable error-bank generation

The central technical objective is to use commonly available LLMs to generate professional-format artifacts at high throughput while ensuring that 1) each artifact contains a known and bounded set of embedded errors aligned to the [Section 5](#) taxonomy, 2) the error is calibrated on the dimensions of severity and detectability, and 3) the instructor retains verifiable control over ground truth through independent checks and an instructor-only key.

It should be acknowledged that this objective shifts, rather than eliminates, the authoring challenge: whereas the prior bottleneck was the manual effort of crafting realistic flaws, the new challenge is ensuring that LLM-generated errors are precisely bounded. The two-stage architecture described below addresses this shift directly by separating correct-artifact generation from controlled defect injection, so that the model is never asked to 'fail' in an unconstrained way. A typical workflow begins with the structured specification of artifacts. The workflow presented below is a design contribution: it specifies the architectural logic, constraint structure, and verification requirements for scalable error-bank generation, and provides implementation materials (prompt templates, notebook, and interface) to support adoption. Systematic empirical validation, including faculty plausibility review of generated artifacts, defect-control audits confirming single-error boundaries, and classroom feasibility trials, constitutes a necessary next stage that the present

design is intended to enable.

Faculty first define an "item spec" that specifies the domain, scope, and permissible methods, then request an artifact with the same surface features that students would encounter in practice (e.g., units, code references where appropriate). This spec should also define *what must be correct* (the surrounding work) and *what may be wrong* (the injected defect). Accordingly, the model is not asked to "make mistakes" in an unconstrained way, but is asked to produce a correct baseline artifact that is later modified in a controlled injection step. This two-stage structure reduces the likelihood of multi-error documents and improves plausibility, which better aligns with the empirical structure of real review tasks (see Fig. 4).

The next stage is error injection with constraint satisfaction. Here, instead of letting the model introduce arbitrary flaws, faculty are expected to explicitly request an error of a specified type (assumption/modeling, constraint/standard misapplication, unit/propagation, sanity violation, omission), paired with a target severity and detectability level. The injection prompt should impose hard constraints: "introduce exactly one primary error," "do not change formatting," "do not create secondary inconsistencies," and "ensure internal coherence so the artifact remains plausible." This is the stage where subtlety and density can be controlled. We suggest that, in most instructional settings, the default should be one primary error per artifact, with occasional higher-density items reserved for advanced cohorts. Importantly, an injection should preserve narrative continuity, with a modeling error that reads like a coherent modeling choice, and an omission that reads like a professional report that simply fails to address a required check, rather than an obviously incomplete draft.

The pipeline must include verification gates that are external to the model's logic. A practical design is to require two independent checks: 1) an instructor (or TA) reproduces the relevant computation pathway or verifies the cited provision and selection logic, and 2) an automated consistency check is applied where feasible (e.g., dimensional analysis scripts and unit tests that recompute key values

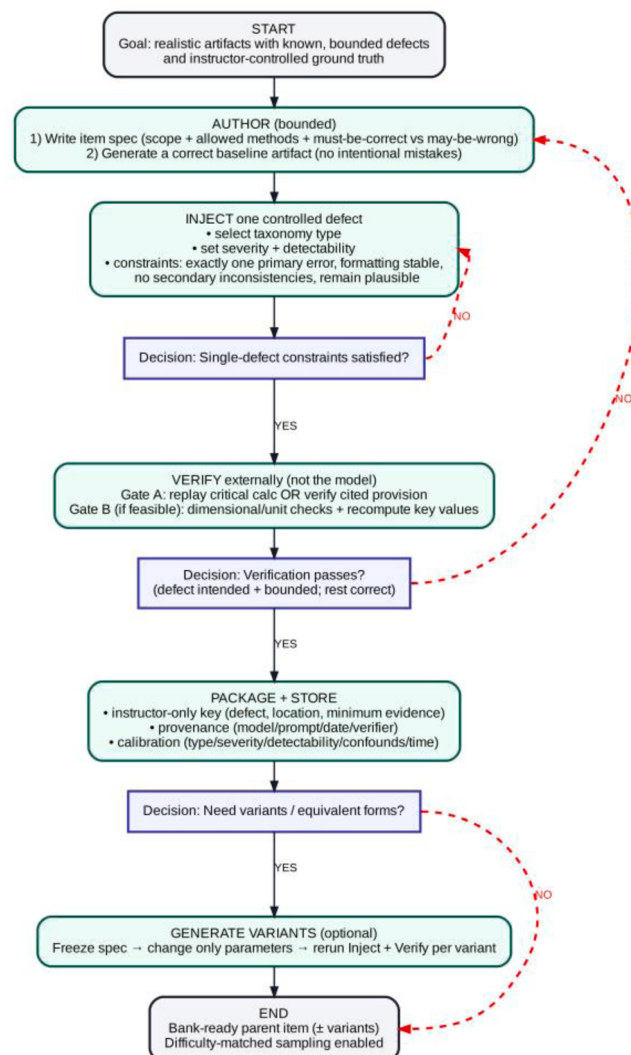


Fig. 4. Flowchart for a typical LLM-assisted authoring pipeline (Note: see prompt templates).

from stated inputs). Even when automation is minimal, a lightweight "calculation replay" step, where the instructor recomputes the critical segment using the shown numbers, can catch unintended defects. Building on these checks, the instructor then generates an instructor-only key that identifies the intended defect, its location, its classification tags, and the minimum evidence a competent reviewer should cite to justify the diagnosis.

Finally, once a validated "parent" item exists, faculty can generate families of variants by changing surface context and numeric parameters while holding the conceptual defect invariant. Technically, this is best done by 1) freezing the item spec, 2) regenerating only the parameter block (i.e., inputs, loads, dimensions, regulatory thresholds), and 3) re-running the injection and verification gates. We recommend that each item in the bank include provenance fields (model name/version, prompt version, generation date, and verifier identity) and calibration fields (error type, severity, detectability, confounds, and expected time-on-task). With these fields, an instructor can assemble equivalent forms by sampling matched difficulty profiles rather than relying on informal judgment.

Below are some readily available prompts that accompany our flowchart, and interested readers might use or extend to their course material:

Prompt no 1 Item specification prompt (baseline artifact: "mostly correct")

ROLE: You are generating a professional engineering work product for instructional review.

TASK: Produce a baseline artifact that is technically correct and professionally formatted.

DOMAIN: [e.g., structural / environmental / transportation]

ARTIFACT TYPE: [calc package | design memo | short technical report | spreadsheet-style calc narrative]

AUDIENCE: [junior engineer | plan reviewer | project manager]

STUDENT LEVEL: [sophomore | senior | graduate]

SCOPE (INCLUDE): [list what must be addressed]

SCOPE (EXCLUDE): [list what must NOT be addressed]

GIVEN DATA: [inputs with units]

REQUIRED OUTPUTS: [what results must be reported]

ALLOWED METHODS: [methods, formulas, code references allowed]

FORBIDDEN METHODS: [methods not allowed]

FORMAT REQUIREMENTS: Use sections and numbered equations where appropriate.

- Keep units explicit at every step.

- Use professional tone and realistic headings.

- Do NOT include any intentional errors.

OUTPUT RULES: Output only the artifact (no commentary, no rubric, no explanation).

Prompt no 2 Controlled error injection prompt to introduce exactly one defect

ROLE: You are modifying an existing engineering artifact to embed a single targeted error.

INPUT: The baseline artifact appears below.

TARGET ERROR TYPE (from taxonomy):

1) Assumption/modeling

2) Constraint/standard misapplication

3) Unit/propagation

4) Physical nonsense/sanity violation

5) Omission

TARGET SETTINGS: Primary error type: [choose one]

- Severity: [low | moderate | high]

- Detectability: [surface | verification | reasoning]

- Density: EXACTLY ONE primary error (no secondary errors)

CONSTRAINTS: Preserve formatting, structure, and professional tone.

- Keep everything else correct.

- Ensure the artifact remains plausible and internally coherent.

- Do NOT insert hints like "this is wrong" or "check this."

OUTPUT RULES: Output only the revised artifact.

- Do not describe the error.

BASELINE ARTIFACT:

<<<PASTE BASELINE HERE>>>

Prompt no 3 Instructor key + metadata prompt (kept private)

ROLE: You are producing an instructor-only key for a review-based assessment item.

INPUT: The flawed artifact appears below.

OUTPUT REQUIREMENTS (structured):

1) Error synopsis (1–2 sentences)

2) Error location (section/page/equation/line reference using the artifact's headings)

3) Taxonomy label (error type)

4) Severity justification (why low/moderate/high, grounded in consequences)

5) Detectability justification (surface/verification/reasoning, grounded in what is required to detect)

- 6) Minimal competent evidence: what a good student must cite or compute to prove the diagnosis
- 7) Expected correction (what should be changed)
- 8) Common false positives (two likely over-flags and why they are not errors)
- 9) Time-on-task estimate (minutes) for a prepared student

OUTPUT RULES: Produce the key only (do not rewrite the artifact).

- Be specific and verifiable.

FLAWED ARTIFACT:

<<<PASTE FLAWED HERE>>>

Prompt no 4 Variant generation prompt (parallel forms with matched difficulty)

ROLE: You are generating parallel variants of an existing validated assessment item.

GOAL: Create N variants that preserve the same conceptual structure and the same primary error type, while changing surface context and numeric inputs.

INVARIANTS (must remain the same): Primary error type: [e.g., constraint misapplication]

- Severity level: [e.g., high]
- Detectability level: [e.g., verification]
- Artifact structure (section headings and steps)
- Skill being assessed (what the reviewer must do)

VARIABLES (may change): Numeric values (within realistic ranges)

- Names/locations (fictional)
- Non-essential context details

OUTPUT: Provide N complete flawed artifacts, labeled Variant A, B, C...

- For each variant, provide an instructor-only metadata block (NOT visible to students): {error_type, severity, detectability, expected_time, confounds_flag}

BASE VALIDATED ITEM:

<<<PASTE VALIDATED FLAWED ARTIFACT HERE>>>

Prompt no 5 Red-team sanity check prompt (catch unintended extra errors)

ROLE: You are auditing a flawed assessment artifact for unintended issues.

TASK: Identify any additional defects, ambiguities, or grading hazards beyond the intended primary error.

OUTPUT: List any secondary errors (if present) with location and why they matter.

- List ambiguities that could produce grading disputes.
- List places where students might reasonably interpret two valid methods.
- Provide a PASS/FAIL verdict: "Safe to deploy as a single-error item" or "Revise required."

FLAWED ARTIFACT:

<<<PASTE FLAWED ARTIFACT HERE>>>

It is worth noting that expert faculty in programming can also operationalize this workflow through a LLM's Application Programming Interface (API), rather than through ad hoc prompt-and-paste interactions to significantly enable reproducibility, auditability, and scalability. In an API-based pipeline, each item becomes the output of a version-controlled "item compiler" that takes a structured specification (e.g., JSON fields for domain, student level, scope constraints, etc.) and produces artifacts through staged calls. This approach supports deterministic regeneration (by storing seeds and prompt versions), automated variant generation (parameter sweeps within bounded ranges), and automated screening (unit consistency scripts, range checks, and red-team audit calls that flag unintended secondary errors or ambiguities). Thus, the resulting bank is not only a collection of documents but also a reproducible build system, in which faculty can recompile items when code changes, rebalance difficulty distributions by sampling metadata tags, and maintain assessment security by rotating variants generated from the same specification while preserving equivalence.

Looking ahead, the staged pipeline described above is also compatible with agentic AI architectures, in which a generator agent produces the artifact, a verifier agent audits it against taxonomy constraints and dimensional-consistency checks, and a reflection loop iterates until all specifications are satisfied — an approach that could further reduce instructor time while maintaining quality assurance. Such a system would treat each item specification as a goal state, with the reflection pattern enabling automated self-correction of unintended secondary errors or plausibility failures before the item reaches the human verification gate. While promising, this extension does not eliminate the need for instructor review at the final stage, particularly for high-stakes items, because the verifier agent itself may share failure modes with the generator. As multi-agent orchestration and tool-use capabilities mature, however, this architecture offers a natural path toward greater automation of error-bank maintenance.

We provide an accompanying Jupyter notebook (LLM_ErrorBank_Pipeline.ipynb) to operationalize this paper's error-bank generation workflow by turning a structured item specification (domain, artifact type, scope, given data, and required outputs) and taxonomy-aligned error tags (primary error type, severity level, and detectability level) into a paired assessment item with a technically correct baseline artifact and a visually comparable "flawed" artifact created by injecting exactly one targeted defect while preserving professional formatting. The notebook then produces an instructor-only key as structured JSON, which records the defect synopsis, where it appears, the expected correction, grading anchors, and likely false positives. It also optionally supports an audit step to check for unintended secondary errors or ambiguity, as well as variant generation for parallel forms. For classroom deployment and sharing, the pipeline exports the artifacts to browser-ready HTML with MathJax-enabled equation rendering and embedded generation metadata (e.g., timestamped headers) to support traceability and reproducibility. This notebook will be published alongside this paper.

The interface of the developed notebook can be seen in Fig. 5. At the moment, this interface is designed to support OpenAI LLMs, but it can be extended to cover any LLM provider. The notebook can be downloaded from the author's website.

8. Objections and limitations

This section addresses three primary objections and acknowledges the limitations of the proposed approach.

8.1. Objections and mitigations

The first objection holds that review itself can be gamed, especially once students learn that assessment has shifted to review; they may seek ways to memorize common error patterns, apply heuristics, or use LLMs to generate review responses. A possible response may build on drawing assessment items from a sufficiently large and varied error bank to prevent memorization from substituting for understanding (because students cannot anticipate which specific errors they will encounter). In addition, requiring justification makes it challenging to succeed through pattern matching alone, since justification demands articulation of principles that rote recognition does not provide. Thus, including verification requirements, where students must demonstrate independent calculations or reference checks that confirm their diagnoses, can ground assessment in performances that cannot be faked. While no assessment is immune to gaming, these mitigations raise the bar substantially (Koretz, 2017).

The second objection holds that production still matters and should not be displaced. This objection is correct because engineers must know how to produce (not only how to review). Our discussion does not propose eliminating production, but rather repositioning it in relation to review. What changes is the evidentiary role of production in assessment. Under conditions of LLM access, production artifacts may still be assigned but should be weighted less heavily or assessed in controlled environments where LLM use is restricted. Review becomes the primary evidence of competence because it retains the validity that production-based evidence has lost. Production and review are thus complementary rather than competing, with each serving distinct purposes in a coherent curricular architecture.

The third objection concerns feasibility/practicality since implementing review-based assessment requires substantial effort to develop error banks and train graders. Instructors already face competing demands, and adding a new assessment paradigm may be impractical. This objection is serious but does not undermine the value of our argument. For example, adoption can be incremental, with the minimal viable implementation involving the incorporation of occasional review exercises using flawed artifacts generated ad hoc or sourced from the categories identified in Section 6.1. As instructors accumulate materials, these can be organized into structured error banks. Sharing across institutions reduces duplication of effort, and the framework's domain-general architecture enables the transfer of task formats, scoring logic, and error taxonomies, even when the defect instances themselves remain domain-specific. One might keep in mind that the workload objection is also context-dependent, as LLMs increasingly compromise the validity of traditional assessment, the cost of not adapting may exceed the cost of adaptation.

8.2. Limitations

The proposed framework is not intended to replace all forms of engineering assessment since several outcome classes may not be validly measured through review. For example, psychomotor and embodied competencies (e.g., laboratory techniques, instrument use) require observation of physical execution and cannot be inferred from a student's critique of an artifact. Likewise, advanced open-ended design competence requires synthesis, trade-off reasoning, and creative generation of viable solutions. Although review can measure a student's ability to detect and justify flaws in a provided design, it does not directly measure the ability to originate an effective design under constraints. More broadly, there is a pedagogical risk that privileging review could narrow instruction toward fault-finding at the expense of the generative, integrative, and creative capacities that engineering and design education must also develop. Review competence and production competence serve complementary developmental functions: review builds evaluative discrimination and principled justification, while production builds synthesis, trade-off reasoning, and the capacity to generate viable solutions under open-ended constraints. A curriculum that shifts too far toward review may produce students who are skilled at identifying what is wrong but underprepared for the constructive demands of professional practice. The framework proposed here should therefore be understood as advocating a rebalancing of assessment emphasis, not a wholesale replacement of production with review. In addition, implementation is inherently domain-specific: the taxonomy and five-operation sequence can generalize as organizing structures, but the instantiated error types, defect inventories, acceptance criteria, and justification standards must be defined within each discipline's governing principles and codes.

9. Conclusions

Large language models have broken the inferential link between artifact quality and student competence that traditional engineering assessment presumed. This disruption, while challenging, also resolves a long-standing constraint with regard to the difficulty of mass-producing realistically flawed artifacts. This paper argues that assessment validity can be strengthened by shifting the emphasis from production to review. The shift we propose corrects that imbalance by aligning assessment with professional reality because engineers must be able to catch errors, not only avoid making them. Our presented argument treats LLM disruption not as a problem to manage but as an opportunity to assess what engineering education has not historically prioritized. The cost structure inversion makes review the scarce and differentiating capability (i.e., while anyone can generate, students with stronger domain

Fig. 5. Interface of the developed LLM error generator [Note: different tabs for different outputs].

understanding are better positioned to verify). From this lens, we formalized review competence as a five-operation construct, namely detect, diagnose, prioritize, justify, and propose correction, which can be taught and assessed at progressive levels of sophistication. In addition, we presented a three-dimensional taxonomy of engineering errors, classified by type, severity, and detectability, which provides the structured error space needed for systematic assessment. Finally, we present prompts and a LLM-based interface for generating review materials on-demand.

Funding

This material is based upon work supported by the National Science Foundation under Grant No. 2447649. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

CRediT authorship contribution statement

M.Z. Naser: Writing – review & editing, Conceptualization.

Declaration of competing interest

The author declares no conflict of interest.

Acknowledgements

None.

Data availability

Data will be made available on request.

References

- ABET. (2022). *PROPOSED program criteria for civil and similarly named engineering programs*. ABET. https://abet.co1.qualtrics.com/jfe/form/SV_0V8Nw9KMIykC8IM accessed July 14, 2023.
- Adams, D. M., McLaren, B. M., Durkin, K., Mayer, R. E., Rittle-Johnson, B., Isotani, S., & Van Velsen, M. (2014). Using erroneous examples to improve mathematics learning with a web-based tutoring system. *Computers in Human Behavior*, 36, 401–411. <https://doi.org/10.1016/j.chb.2014.03.053>
- Atman, C. J., Adams, R. S., Cardella, M. E., Turns, J., Mosborg, S., & Saleem, J. (2007). Engineering design processes: A comparison of students and expert practitioners. *Journal of Engineering Education*, 96, 359–379. <https://doi.org/10.1002/j.2168-9830.2007.tb00945.x>
- Biggs, J., & Tang, C. (2020). Constructive alignment: An outcomes-based approach to teaching anatomy. *Teaching Anatomy: A Practical Guide*. https://doi.org/10.1007/978-3-030-43283-6_3
- Blockley, D. (2013). On communicating the uncertainty of risk. *Academia Education*, 24–33. https://www.academia.edu/download/68336673/On_Communicating_the_Uncertainty_of_Risk20210727-2941-2sbgn9.pdf#page=25 accessed January 3, 2026.
- Borji, A., & Ai, Q. (2023). A categorical archive of ChatGPT failures. <https://arxiv.org/pdf/2302.03494> accessed January 3, 2026.
- Brown, D. E. (2014). Students' Conceptions as dynamically emergent structures. *Science and Education*, 23, 1463–1483. <https://doi.org/10.1007/s11191-013-9655-9>
- Carless, D., & Boud, D. (2018). The development of student feedback literacy: Enabling uptake of feedback. *Assessment and Evaluation in Higher Education*, 43, 1315–1325. <https://doi.org/10.1080/02602938.2018.1463354>
- Chi, M., Glaser, R., & Farr, M. (2014). *The nature of expertise*. Psychology Press. <https://doi.org/10.4324/9781315799681>
- Chi, M. T. H., De Leeuw, N., Chiu, M., & Lavancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science*, 18, 439–477. https://doi.org/10.1207/s15516709cog1803_3
- Delatte, N. J. (2015). *Systems for structural failure investigations in the united states*, in: *Forensic eng* (pp. 569–578). Reston, VA: American Society of Civil Engineers. <https://doi.org/10.1061/9780784479711.055>, 2015.
- Dunning, D., Johnson, K., Ehrlinger, J., & Kruger, J. (2003). Why people fail to recognize their own incompetence. *Current Directions in Psychological Science*, 12, 83–87. <https://doi.org/10.1111/1467-8721.01235>
- Dwyer, C. P., Hogan, M. J., & Stewart, I. (2014). An integrated critical thinking framework for the 21st century. *Thinking Skills and Creativity*, 12, 43–52. <https://doi.org/10.1016/j.tsc.2013.12.004>
- Ennis, R. H. (1989). Critical thinking and subject specificity: Clarification and needed research. *Educational Researcher*. <https://doi.org/10.3102/0013189x018003004>
- Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, 100(3), 363–406. <https://doi.org/10.1037/0033-295x.100.3.363>
- Gibbs, G., & Simpson, C. (2004). Conditions under which assessment supports students' Learning. *Learning in Teaching in Higher Education*.
- Goldratt, E. M. (1990). Theory of constraints. <http://brharnet.edu.in/br/wp-content/uploads/2018/11/5.pdf> accessed January 3, 2026.
- Große, C. S., & Renkl, A. (2007). Finding and fixing errors in worked examples: Can this foster learning outcomes? *Learning and Instruction*, 17, 612–634. <https://doi.org/10.1016/j.learninstruc.2007.09.008>
- Haladyna, T. M., & Downing, S. M. (2004). Construct-irrelevant variance in high-stakes testing. *Educational Measurement: Issues and Practice*, 23(1), 17–27. <https://doi.org/10.1111/j.1745-3992.2004.tb00149.x>
- Hess, J. L., & Fore, G. (2018). A systematic literature review of US engineering Ethics interventions. *Science and Engineering Ethics*. <https://doi.org/10.1007/s11948-017-9910-6>
- Hostetter, H., Naser, M. Z., Huang, X., & Gales, J. (2024). The role of large language models (AI chatbots) in fire engineering: An examination of technical questions against domain knowledge. *Natural Hazards Research*, 4, 669–688. <https://doi.org/10.1016/J.NHRES.2024.06.003>
- Hundhausen, C., Agarwal, P., & Trevisan, M. (2011). Online vs. face-to-face pedagogical code reviews: An empirical comparison. In *SIGCSE'11 - proc. 42nd ACM tech. symp. comput. sci. educ.* <https://doi.org/10.1145/1953163.1953201>
- Indriasari, T. D., Luxton-Reilly, A., & Denny, P. (2020). A review of peer code review in higher education. *ACM Transactions on Computing Education*, 20, 1–25. <https://doi.org/10.1145/3403935>
- Z. Ji, T. Yu, Y. Xu, N. Lee, E. Ishii, P. Fung, Towards mitigating LLM hallucination via self reflection, in: 2023. <https://doi.org/10.18653/v1/2023.findings-emnlp.123>.
- Kane, M. T. (2013). Validating the interpretations and uses of test scores. *Journal of Educational Measurement*, 50, 1–73. <https://doi.org/10.1111/jedm.12000>
- Kapur, M., & Bielaczyc, K. (2012). Designing for productive failure. *Journal of the Learning Sciences*, 21, 45–83. <https://doi.org/10.1080/10508406.2011.591717>
- Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, Article 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
- Koretz, D. (2017). *The testing charade: Pretending to make schools better*. University of Chicago Press.
- Lane, S., Raymond, M. R., Haladyna, T. M., & Downing, S. M. (2016). Test development process. In *Handb. test dev. second ed.* <https://doi.org/10.4324/9780203102961-2>
- Litzinger, T. A., Lattuca, L. R., Hadgraft, R. G., Newstetter, W. C., Alley, M., Atman, C., DiBiasio, D., Finelli, C., Diefes-Dux, H., Kolmos, A., Riley, D., Sheppard, S., Weimer, M., & Yasuhara, K. (2011). Engineering education and the development of expertise. *Journal of Engineering Education*, 100, 123–150. <https://doi.org/10.1002/j.2168-9830.2011.tb00006.x>
- Luxton-Reilly, A. (2009). A systematic review of tools that support peer assessment. *Computer Science Education*, 19, 209–232. <https://doi.org/10.1080/08993400903384844>
- Macneil, S., Tran, A., Hellas, A., Kim, J., Sarsa, S., Denny, P., Bernstein, S., & Leinonen, J. (2023). Experiences from using code explanations generated by large language models in a web software development E-book. In *SIGCSE 2023 - Proc. 54th ACM tech. symp. comput. sci. educ.* <https://doi.org/10.1145/3545945.3569785>
- Messick, S. (1995). Validity of psychological assessment: Validation of inferences from persons' responses and performances as scientific inquiry into score meaning. *American Psychologist*, 50(9), 741–749. <https://doi.org/10.1037/0003-066X.50.9.741>

- Mitcham, C. (2005). National Society of Professional Engineers (NSPE) Code of Ethics. *Encyclopedia of Science, Technology, and Ethics*, 4.
- Mollick, E. R., & Mollick, L. (2023). Using AI to implement effective teaching strategies in classrooms: Five strategies, including prompts. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.4391243>
- Moskal, B. M., Leydens, J. A., & Pavelich, M. J. (2002). Validity, reliability and the assessment of engineering education. *Journal of Engineering Education*, 91, 351–354. <https://doi.org/10.1002/j.2168-9830.2002.tb00714.x>
- Naser, M. Z., Ross, B., Ogle, J., Kodur, V., Hawileh, R., Abdalla, J., & Thai, H.-T. (2024). Evaluating the performance of artificial intelligence chatbots and large language models in the FE and PE structural exams. *Practice Periodical on Structural Design and Construction*, 29, Article 02524001. <https://doi.org/10.1061/PPSCFX.SCENG-1369>
- Olds, B. M., Moskal, B. M., & Miller, R. L. (2005). Assessment in engineering education: Evolution, approaches and future collaborations. *Journal of Engineering Education*, 94, 13–25. <https://doi.org/10.1002/j.2168-9830.2005.tb00826.x>
- Oreskes, N., Shrader-Frechette, K., & Belitz, K. (1994). Verification, validation, and confirmation of numerical models in the earth sciences. *Science (New York, N.Y.)*, 263, 641–646. <https://doi.org/10.1126/science.263.5147.641>
- Passow, H.J., "What competencies should engineering programs emphasize? A meta-analysis of practitioners' opinions informs curricular design." *Proceedings of the 3rd International CDIO Conference*. 2007.
- Patchan, M. M., Schunn, C. D., & Clark, R. J. (2018). Accountability in peer assessment: Examining the effects of reviewing grades on peer ratings and peer feedback. *Studies in Higher Education*, 43, 2263–2278. <https://doi.org/10.1080/03075079.2017.1320374>
- Porter, A. A., Basili, V. R., & Votta, L. G. (1995). Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on Software Engineering*, 21, 563–575. <https://doi.org/10.1109/32.391380>
- Reason, J. (1993). The identification of latent organizational failures in complex systems. In *Verif. Valid. Complex syst. hum. factors issues*. https://doi.org/10.1007/978-3-662-02933-6_13
- Redish, E. F., & Kuo, E. (2015). Language of physics, Language of math: Disciplinary culture and dynamic epistemology. *Science and Education*, 24, 561–590. <https://doi.org/10.1007/s11191-015-9749-7>
- Rittle-Johnson, B., & Star, J. R. (2007). Does comparing solution methods facilitate conceptual and procedural knowledge? An experimental study on learning to solve equations. *Journal of Educational Psychology*, 99, 561–574. <https://doi.org/10.1037/0022-0663.99.3.561>
- Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. (2022). Automatic generation of programming exercises and code explanations using large language models. In *ICER 2022 - Proc. 2022 ACM conf. int. comput. educ. res.*. <https://doi.org/10.1145/3501385.3543957>
- Sheppard, S. D., Pellegrino, J. W., & Olds, B. M. (2008). On becoming a 21st century engineer. *Journal of Engineering Education*, 97, 231–234. <https://doi.org/10.1002/j.2168-9830.2008.tb00972.x>
- Siegler, R. S. (2002). Microgenetic studies of self-explanation. *Microdevelopment: Transition processes in development and learning*, 31, 58. <https://doi.org/10.1017/cbo9780511489709.002>
- Stark, R., Kopp, V., & Fischer, M. R. (2011). Case-based learning with worked examples in complex domains: Two experimental studies in undergraduate medical education. *Learning and Instruction*, 21, 22–33. <https://doi.org/10.1016/j.learninstruc.2009.10.001>
- Stechly, K., Karthik, V., & Subbarao, K. (2025). On the self-verification limitations of large language models on reasoning and planning tasks. *International Conference on Learning Representations*, 2025.
- Strickroth, S. (2023). Does peer code review change my mind on my submission?. In *Annu. Conf. Innov. Technol. Comput. Sci. Educ.*. ITiCSE. <https://doi.org/10.1145/3587102.3588802>
- Tai, J., Ajjawi, R., Boud, D., Dawson, P., & Panadero, E. (2018). Developing evaluative judgement: Enabling students to make decisions about the quality of work. *Higher Education*, 76, 467–481. <https://doi.org/10.1007/s10734-017-0220-3>
- Trevelyan, J. (2010). Reconstructing engineering from practice. *Engineering Studies*, 2, 175–195. <https://doi.org/10.1080/19378629.2010.520135>
- Wang, W. (2017). Using rubrics in student self-assessment: Student perceptions in the English as a foreign language writing context. *Assessment and Evaluation in Higher Education*, 42, 1280–1292. <https://doi.org/10.1080/02602938.2016.1261993>
- Wiggins, G.P., & Jay, M. (2005). *Understanding by design*. Ascd.