Research Square

Preprints are preliminary reports that have not undergone peer review. They should not be considered conclusive, used to inform clinical practice, or referenced by the media as validated information.

FORGE: A Framework for Orchestrated Role-based Generation in Education

Emmanuel Gonzalez

emmanuelgonzalez@asu.edu

Arizona State University https://orcid.org/0000-0002-3021-9842

Paul T. Grogan

Arizona State University https://orcid.org/0000-0001-8986-4806

Research Article

Keywords: generative AI, large language models, human-AI collaboration, instructional material development, design science research, role-based prompting

Posted Date: October 22nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11446615/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: The authors declare no competing interests.

FORGE: A Framework for Orchestrated Role-based Generation in Education

Emmanuel M. Gonzalez[1*] and Paul T. Grogan[1]

[1*]School of Computing and Augmented Intelligence, Arizona State University, Tempe, 85281, AZ, USA.

*Corresponding author(s). E-mail(s): emmanuelgonzalez@asu.edu; Contributing authors: paul.grogan@asu.edu;

Abstract

Research on generative AI in higher education has concentrated on students as users, while how instructors collaborate with AI on course materials remains underexamined. Without a framework for that collaboration, workflows remain improvised and prompting habits rarely transfer between instructors or deliverables. We introduce the Framework for Orchestrated Role-based Generation in Education (FORGE). FORGE specifies which roles to orchestrate, how those roles change as a material moves from requirements capture through generation to verification, and which interaction parameters vary with the material. It defines seven dimensions and four roles: Reader, Writer, Critic, and Orchestrator. Developed under design science research and grounded in distributed cognition and systems engineering process models, FORGE was evaluated through two retrospective case studies, a convergent exam and a divergent term project specification, in one undergraduate information systems engineering course. All 142 instructor prompts, 95 for the exam and 47 for the term project, were classified on four measures, including what each requested and how it was framed, whether to seek judgment or direct execution. That framing, which FORGE calls interaction style, is a parameter rarely examined. Under the same architecture, the cases produced different collaboration profiles: 26% of term project prompts sought judgment and 23% requested new content, compared to 4% and 6% of exam prompts. In each case, the instructor directed the AI, and the AI asked questions in return. Seven guidelines drawn from both cases turn improvised collaboration into practice an instructor can describe and carry to the next deliverable.

Keywords: generative AI, large language models, human-AI collaboration, instructional material development, design science research, role-based prompting

1 Introduction

The arrival of generative artificial intelligence (AI) in higher education, in the form of large language models (LLMs), produced a substantial pedagogical literature within months (Lo, 2023). That literature has concentrated on students as its users. That emphasis is understandable, as AI in the classroom is most visible as a change in what students can do. It leaves a parallel phenomenon underexamined, however. Instructors, too, use generative AI as a collaborator in producing instructional materials (Lo, 2023; Shi et al., 2026). Interviewed instructors who drew on generative AI for lesson planning and assessment development reported not sharing the practice with students or colleagues, citing the stigma attached to it (Barnes & Tour, 2026). Comparatively little scholarship addresses how that production is structured, and less still proposes a shared framework that carries across deliverables, instructors, and disciplines.

Instructional material development in higher education, whether of an exam, a term project, or a lab sequence, is a substantial and recurring workload. Each demands domain expertise, pedagogical judgment, alignment with prior course content, calibration of difficulty, and iteration against the instructor's own changing understanding of what students can do, all compressed into a semester cycle. Generative AI appears well-suited to lower this burden, as ChatGPT has written exam items of quality comparable to those of experienced examiners in roughly a tenth of the time (Cheung et al., 2023). Without a framework, AI-assisted material development remains improvised, with each instructor developing their own prompting conventions. Those conventions rarely transfer, as the design decisions behind a successful use are seldom reported and the use itself is often kept private (Barnes & Tour, 2026; Shi et al., 2026). A convention that succeeds in one session for one material may fail in another, and no shared vocabulary names the variables that distinguish the two.

A framework for AI-assisted instructional material development would supply that vocabulary and carry it across instructors and deliverables. Across 50 generative-AI articles, 66 evidence syntheses, 88 classroom implementations, and 28 instructor-AI co-design studies reviewed, no such framework has been proposed (Bond et al., 2024; Lo, 2023; Shi et al., 2026; Z. Wang et al., 2026). However, the practices such a framework would organize are already emerging. Iterative refinement and instructor feedback loops recur throughout those studies without theoretical anchoring, but how instructors decide when to accept, revise, or reject AI-generated output remains an open question (Z. Wang et al., 2026). Those decisions are what role decomposition makes explicit, by assigning source reading, content generation, draft critique, and final acceptance to separate roles. Multi-agent LLM systems, in which several model instances each take a defined role, implement role decomposition but have not been applied to instruction (Chowa et al., 2025; Wu et al., 2023). None of 108 screened studies spanning healthcare, robotics, software engineering, and finance applies role decomposition to instructional work (Chowa et al., 2025). Multi-agent architectures thus supply one part of such a framework, the role partition (Wu et al., 2023). The other two parts come from distributed cognition, which explains why work divided among people, tools, and agents still functions as one system (Hollan et al., 2000; Hutchins, 1995), and from systems engineering, whose process standards define a lifecycle of requirements, design, and verification across which the human's role shifts

(INCOSE, 2015). We are aware of no work that brings the three together for instructional material development. Without that synthesis, instructors reinvent the practices that systems engineering codified long ago, project after project.

We address three research questions. First, which design-space parameters characterize AI-assisted instructional material development, and which of them does current practice leave underspecified? Second, how do those parameters vary between convergent and divergent materials and across the lifecycle phases of requirements capture, generation, and verification? Third, what transferable guidelines follow from that variation, and how do they align with established process norms for developing complex artifacts? To answer these questions, we introduce the Framework for Orchestrated Role-based Generation in Education (FORGE), a seven-dimension design space for AI-assisted instructional material development. The framework was developed under design science research and is grounded in distributed cognition and systems engineering process models. The Reader, Writer, Critic, and Orchestrator roles instantiate it across three lifecycle phases, and seven guidelines govern its use. We evaluate FORGE through two case studies in one undergraduate course, a convergent exam and a divergent term project specification.

2 Background and Theory

2.1 Generative AI in Education

Research on AI in education falls into three lines, two of them well established (Bond et al., 2024; Lo, 2023). The first concerns students' encounters with generative AI, including academic integrity and detection, assessment validity when coursework can be AI-generated, shifts in learning behavior, and equity of tool access (Lo, 2023). Of 66 syntheses in a decade-long meta-review of AI in higher education, only three focused on assessment (Bond et al., 2024). Each examined student responses to assessments that already existed, through plagiarism detection, automatic scoring of open-ended answers, or learning-analytics feedback (Albluwi, 2019; Banihashem et al., 2022; del Gobbo et al., 2023). This line positions the student as the AI user and the instructor as a policy-setter, and instructors appear mainly through their concerns about integrity, validity, and the erosion of critical thinking (Mhlanga, 2023; Sallam, 2023).

The second line consists of single-course studies, each adopting a specific AI tool for a specific purpose. Across 88 classroom implementations, the most common purposes are intelligent tutoring, general learning support, and automated assessment and feedback (Shi et al., 2026). Each study reports what worked with a particular tool in a particular course, without stating the design decisions behind that success, so its findings rarely generalize. These studies also treat AI assistance as a one-shot generation task and ask whether a single generated artifact can be adopted as it stands (Megahed et al., 2024). That framing overlooks the iteration after the first draft, in which the instructor finds defects and calibrates the material to the course.

The third line concerns how instructors themselves draft, calibrate, and revise materials with AI assistance, and it has received less attention than the other two (Shi et al., 2026). Instructor-side use is cataloged by function, such as generating course materials, drafting assessment tasks, and translating content, rather than by how the

collaboration producing those materials should be structured (Lo, 2023). Only nine of the 88 classroom implementations generate learning content, and those report single deployments rather than transferable designs (Shi et al., 2026). The nearest precedent is automatic item generation (AIG), which predates generative AI by more than a decade and derives large numbers of items from expert-authored templates or cognitive models (Gierl et al., 2012). Like AI-assisted material development, AIG hands the production of assessment items to a machine, but only after a designer has fixed each item's form in a template. In AI-assisted development, the form is still being decided during the collaboration, so the instructor contributes judgment about what each material should be rather than a template written in advance.

Academic integrity and the accuracy and reliability of AI output are two concerns that cut across all three lines of research (Lo, 2023; Mhlanga, 2023; Sallam, 2023), and in instructional material they meet. Assessments that test critical thinking rather than recall are a standard response to the integrity concern, but producing them at quality is labor-intensive. However, delegating that labor to an LLM raises concerns about accuracy, as unreliable output may take the form of hallucination, bias, and unreferenced claims. The time an LLM saves in generation therefore comes at a risk to accuracy. To save time while maintaining accuracy, a workflow must divide the work, with the LLM supplying generation bandwidth and the instructor keeping the judgment that governs accuracy. None of the three lines of research says how that division should be made.

2.2 Multi-Agent LLM Architectures

Multi-agent LLM systems offer one established way to divide that work. A single prompt that concentrates requirements capture, generation, and verification into one exchange scales poorly as the task grows. A multi-agent system instead distributes the labor across specialized agents that exchange intermediate artifacts. The pattern has been formalized as a general-purpose multi-agent conversation framework in which human and LLM agents coordinate to complete tasks, with software-engineering pipelines as the canonical example (Wu et al., 2023). In most such systems, an expert fixes each agent's role at the outset, so, as in AIG, the human's design work comes before generation begins. Unlike an AIG template, which fixes the form of every output, a role fixes only who does which part of the work, and the agents plan their own steps (Chowa et al., 2025).

Most multi-agent architectures, like the software-engineering pipelines that serve as their canonical example, were developed outside education. The pattern transfers, but the roles and coordination must be re-specified against the artifacts and quality criteria that instructional material imposes. The number of agents, the assignment of their roles, the form of their interaction, and the scope of automation remain open questions even in the general case (Wu et al., 2023). Education-specific systems have begun to appear, generating lesson plans, worksheets, and multiple-choice items through staged pipelines in which one agent drafts and another critiques. However, some of these systems show the instructor only the finished artifact (J. Wang et al., 2025), which removes the iteration after the first draft in which the instructor finds defects and calibrates the material to the course.

The instructor's participation in iteration depends on where pedagogical judgment resides. Adapting multi-agent architectures for instructional material development therefore requires a decision on that question as well as a domain-specific role partition. One answer moves that judgment into the architecture itself, encoding instructional design principles in the agents so the system can generate without the instructor (J. Wang et al., 2025). The other keeps judgment with the instructor and uses the architecture to organize the collaboration around it, the arrangement commonly called human-in-the-loop (Wu et al., 2023). The roles to define and the placement of pedagogical judgment remain unsettled.

2.3 Human-AI Collaboration and Teaming

Evidence on where judgment should reside comes from research on human-AI collaboration, which treats the human-AI pair as the unit of performance and asks how to divide a decision so that the pair performs best. Its benchmark is complementary team performance, in which the pair outperforms both the AI alone and the human alone. A common test gives the human an AI recommendation and varies what accompanies it, such as the AI's confidence score or an explanation of its reasoning. Across three tasks and over 1,500 participants, explanations did not improve the pair's accuracy beyond what the confidence score alone achieved (Bansal et al., 2021). Explanations increased the rate at which the human accepted the recommendation, whether or not it was correct, so the pair's accuracy rose on items the AI got right and fell on items it got wrong (Bansal et al., 2021). The same pattern appears across 35 experimental studies covering 19,774 participants, where explanations raised a system's perceived acceptability without improving decision accuracy and themselves became heuristics that substituted for scrutiny (Romeo & Conti, 2026). Judgment about correctness therefore cannot rest on the AI's account of its own output. The AI can supply the recommendation, but verification must stay with a human who brings knowledge the AI lacks.

Guidance on dividing work between people and AI predates LLMs, but it addresses only the interface, meaning how an AI product presents its output and how users accept, dismiss, or correct it. Mixed-initiative interaction, a design stance in which users and intelligent agents collaborate, set out 12 principles for that interface (Horvitz, 1999). The most relevant here holds that an agent unsure of the user's intent should ask, provided the question is worth the interruption (Horvitz, 1999). Later work consolidated two decades of such recommendations into 18 design guidelines and tested them on 20 AI products (Amershi et al., 2019). Developing instructional material depends instead on the workflow behind the interface, the sequence of requests, drafts, and checks that produces a material. No guidance has yet been written for that workflow.

In human-in-the-loop verification, the AI produces candidate output and the human accepts or rejects it. This arrangement keeps the human in charge of consequential decisions and guards against inaccurate output, but it does not say how the human's role should change as a material moves from requirements capture through generation to verification. Studies of human-AI teams typically measure performance at a single decision (Bansal et al., 2021), which suits LLM use poorly. Over repeated prompts, the human's and the model's contributions become hard to separate, and the

human's reliance on the model builds over a session rather than at one point (Ibrahim et al., 2026). None of this work specifies the human's role in each phase of development or when that role should change.

2.4 Instructional Material Quality and Validity

The finished material, however its production is divided, must still assess the intended learning outcomes, a requirement formalized as validity. Validity concerns the conclusions drawn from scores, which hold to the extent that the test covers what it is intended to measure and nothing else. One way validity fails is *construct-irrelevant easiness*, where a cue in the item lets students answer correctly without the knowledge being tested, so their scores overstate what they know (Messick, 1995). Item-writing guidance treats this failure as a first-order concern (Haladyna et al., 2002). The risk is high in programming assessments, where the code in a question can reveal its own answer.

Validity also depends on the wrong answers a multiple-choice item offers. In assessments of conceptual understanding, these wrong answers, or *distractors*, are drawn from documented student misconceptions (Streveler et al., 2008). Plausible distractors are difficult to write, even for professional item writers. A four-option item carries three wrong answers, yet across four standardized tests most items had only one distractor that students selected, and fewer than one item in 10 had all three distractors functioning (Haladyna et al., 2002).

An LLM-assisted workflow must produce items that give no cue to their answer and distractors that students select, but neither check can be left to the AI alone. The construct an item is meant to measure resides in the instructor's course design, and no improvement in generation substitutes for the judgment that defines it. Such a workflow must therefore separate checks that rules can govern, which an agent can carry, from judgments about the construct, which only the instructor can make. The boundary between these two kinds of check has not been specified, and specifying it requires an account of how cognitive work can be divided between people and their tools.

2.5 Distributed Cognition and Systems Engineering

Distributed cognition provides an account of how cognitive work is divided between people and their tools, treating cognition as a property of a system of people, artifacts, instruments, and representations rather than of an isolated mind (Hutchins, 1995). Extended to human-computer interaction, distributed cognition treats a person's working materials as part of the cognitive process and extends that process across time, so what one episode produces becomes the input to the next (Hollan et al., 2000). Automated histories of interaction therefore become an important source of evidence about how such work is done (Hollan et al., 2000).

An instructor developing material with LLM agents and a curated set of course materials is a distributed cognitive system with three components, the instructor, the AI, and the materials. No single component needs to carry the full cognitive load (Hutchins, 1995), so each should carry what it is best positioned to carry. The AI

carries work governed by explicit rules, such as formatting, reordering, and checking syntax. That work divides further into reading, generating, and verifying, and one model can perform all three functions, provided each prompt asks it to perform only one. The instructor carries the pedagogical judgment that no written rule or document captures. The curated materials carry the course content the AI reads, and scoping them per interaction keeps the AI's functions divided, since broad or unscoped inputs risk returning the divided work to a single agent.

Distributed cognition explains how work divides among people and tools but not how that division changes as a product develops, and systems engineering supplies that account. Three of its ideas transfer to instructional development. First, requirements definition, design, and verification and validation (V&V) are distinct processes, each with defined inputs, activities, and outputs (INCOSE, 2015). Capturing requirements, generating a draft, and verifying it are likewise distinct activities in developing instructional material. Second, the processes do not run in sequence, and verification in particular runs across every stage, with the method for verifying a requirement specified when the requirement is first written (INCOSE, 2015). Third, an integrator owns the properties that no subsystem owns on its own, namely coherence, consistency, and correctness at the interfaces (INCOSE, 2015). In instructional development, that role belongs to the instructor.

2.6 Research Gaps

Four gaps remain in research on AI-assisted instructional material development. The generative-AI literature catalogs instructor-side use by function but not how the collaboration producing a material should be structured (Section 2.1). Multi-agent architectures supply a role partition but not where pedagogical judgment resides (Section 2.2). Human-AI collaboration research prescribes the division of labor at the interface and at a single decision but not across a multi-artifact lifecycle (Section 2.3). Assessment validity sets principles a workflow must encode without delegating the judgment behind them (Section 2.4). Distributed cognition supplies the account of why a functional role partition works, and systems engineering supplies the lifecycle along which the human role is reallocated (Section 2.5). FORGE, derived from two development efforts in one course (Section 3), combines the two accounts into a seven-dimension design space (Section 4). The same two cases exercise that design space (Section 5), and seven guidelines operate on it (Section 6).

3 Research Setting

The two materials analyzed here were developed for one undergraduate information systems engineering course (IEE 305, Fall 2025), taught by the first author. Every session used one model, Claude Sonnet 4.5, accessed through Claude Code, a command-line tool that reads and writes files in directories the instructor names. The two materials differ in whether their completion state is known before AI-assisted development begins.

Exam 3 is a summative assessment covering the final third of the course, namely object-relational mapping, representational state transfer (REST) architecture, and

FastAPI. It is a *convergent* material, whose success criteria are fixed before development begins, so the instructor knows the completion state in advance. Every question must test a specified outcome, resolve to a single defensible answer, avoid leaking that answer in the surrounding code, and be gradable within a bounded time. A shuffled Version B and a practice exam must stay consistent with the exam.

The Term Project is a semester-long group assignment in which teams build a full-stack information system, from data model to frontend, around a public application programming interface (API) drawn from an instructor-provided set. The material developed here is the specification itself, covering candidate APIs, scope, structured query language (SQL) query requirements, rubric language, a proposal template, and a slide outline. It is a *divergent* material, which admits multiple valid configurations, so the instructor does not know the completion state in advance. Its requirements evolve as candidate APIs are explored, and the instructor settles them one piece at a time.

Claude Code keeps a local append-only prompt history (`history.jsonl`) recording every instructor prompt with a millisecond timestamp and a session identifier. That history is the primary record, an automated history of interaction of the kind distributed cognition treats as evidence of how such work is done (Section 2.5). The delivered materials and the curated input trees are the secondary sources. The history retains instructor prompts but not agent responses, which were not preserved beyond the sessions in which they occurred. Agent behavior was therefore observed during the sessions and recalled in analysis, not recovered from the record. The quotations in this article are instructor prompts reproduced verbatim as typed, and the one quoted agent question is hypothetical and marked as such.

The study is retrospective, and the instructor is also the analyst. The same person taught the course, operated the AI agents in both cases, and coded the record reported here. That arrangement removes the independence an outside analyst would bring. It also supplies the course knowledge needed to read the prompts, since recognizing that an item tested material never covered in lecture requires knowing what was covered.

Neither the framework nor the analysis was pre-registered. FORGE was formulated after both efforts concluded, and its dimensions were derived from the record. Design science calls this entry point client-initiated, meaning that a practical solution observed to work is taken up afterward and rigor is applied retroactively (Section 5.1.1; Peffers et al., 2007). The two cases are therefore a source of the framework as well as a demonstration of it, and they cannot independently confirm the dimensions they generated.

4 The FORGE Framework

4.1 Framework Overview

FORGE defines a design space of seven dimensions that instructors use to configure a multi-agent generative AI system for developing instructional material (Figure 1). The dimensions fall into five categories. The architectural category holds Agents and Roles (D1), and the procedural category holds Process Phases (D2) and Interaction Mode (D3). The interactional category holds Interaction Style (D4) and Inquiry Direction (D5). The allocative category holds Task Allocation (D6), and the contextual category

holds Input Curation (D7). Each dimension leaves the instructor a question to settle, from how the work divides into roles to what the AI may read (Figure 2). Of the seven, Interaction Style and Inquiry Direction are new to the design space. Interaction Style names a rarely examined parameter, whether a prompt seeks judgment or directs execution, and Inquiry Direction treats the AI's offer of options as a form of asking. Although FORGE was derived from the development of an exam and a term project, it is intended for instructional materials generally, including lecture slides, rubrics, syllabi, homework, and case materials.

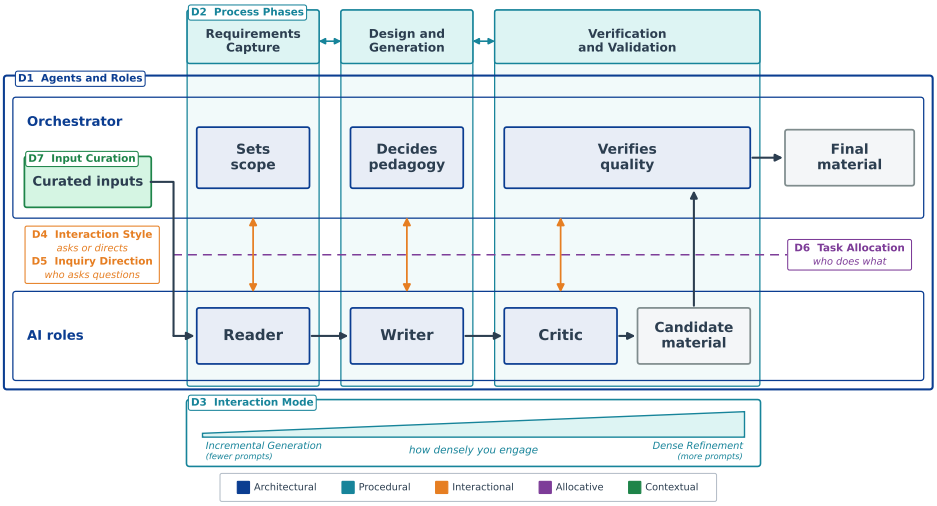

Fig. 1 FORGE framework overview. The Orchestrator, held by the instructor in both case studies, works with three AI roles, the Reader, Writer, and Critic (D1), through three phases, Requirements Capture, Design and Generation, and Verification and Validation (D2). The Orchestrator curates the inputs the Reader reads. The Writer drafts from the Reader's output, the Critic checks the draft, and the candidate material returns to the Orchestrator, who sets scope, decides pedagogy, verifies quality, and releases the final material. Interaction Mode (D3) sets the cadence of the work, Interaction Style (D4) and Inquiry Direction (D5) describe how the Orchestrator and the AI roles communicate, Task Allocation (D6) divides the work between them, and Input Curation (D7) fixes what the Reader may read. Color marks the five categories.

4.2 D1 — Agents and Roles

FORGE defines four roles, the Reader, Writer, Critic, and Orchestrator (Figure 1). The Reader summarizes source materials selected by the instructor, such as prior exams and assignments, lecture content, and documentation. The Writer generates candidate content, such as questions, prose, specifications, and code, from the Reader's output and the Orchestrator's instructions. The Critic evaluates Writer outputs against pedagogical, structural, or contextual criteria and flags defects. The Orchestrator coordinates the other three roles, makes decisions that cross agent boundaries, and holds authority over pedagogical choices, including Final Acceptance. D1 leaves open who holds the Orchestrator role. In both case studies, one instructor held it. The role

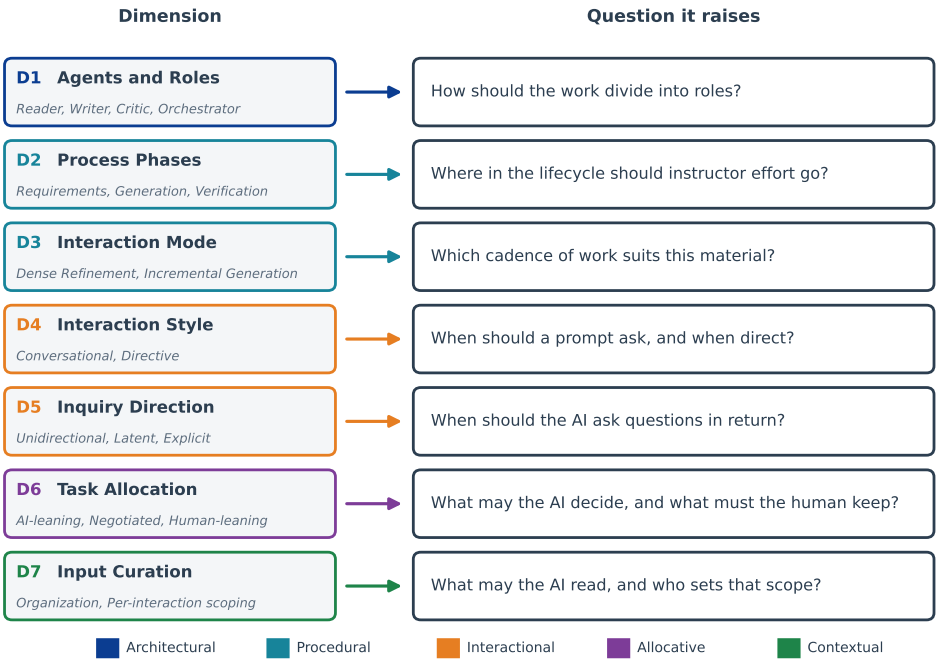

Fig. 2 The seven FORGE dimensions and the question each raises for the instructor. Color marks the five categories.

could instead be shared by a teaching team or, in a fully automated pipeline, given to another AI agent. The remaining dimensions describe the instructor as Orchestrator, as in both case studies, and the same distinctions apply to any other holder of the role. The role partition follows the distributed cognition account in Section 2.5, and separating generation from verification has improved results in other multi-agent systems (Wu et al., 2023). Both cases realized the Reader, Writer, and Critic within one model by scoping each prompt to a single role, and the opening prompt of Exam 3 named all three roles in lifecycle order.

4.3 D2 — Process Phases

FORGE organizes instructional material development into three phases, Requirements Capture, Design and Generation, and V&V, adapted from the process structure of systems engineering (Section 2.5). Requirements Capture fixes the material's purpose, scope, constraints, and success criteria. Design and Generation produces candidate content that meets those requirements, such as questions, prose, specifications, and examples. V&V checks whether the candidate meets the requirements, flags quality concerns, and returns feedback for revision. The Reader, Writer, and Critic each serve one phase (Figure 1), and the Orchestrator takes part in all three, setting scope, deciding pedagogy, and verifying quality. The phases do not run in strict order. When V&V

uncovers a gap, the material returns for fresh scoping and a new draft, an iteration that systems engineering treats as expected rather than exceptional (INCOSE, 2015).

4.4 D3 — Interaction Mode

FORGE distinguishes two interaction modes. *Dense Refinement* iterates on a single deliverable through many exchanges per session until it is finalized. *Incremental Generation* produces one piece of a larger material at a time, in short sessions of few prompts, and releases each piece as it settles. The choice of mode follows from the material. A convergent material, whose completion state is known in advance, changes through small revisions to a deliverable of fixed shape, so it suits Dense Refinement. A divergent material, whose completion state is settled one piece at a time, is built from distinct, partially independent pieces, so it suits Incremental Generation. Exam 3 and the Term Project specification are one material of each type (Section 3).

4.5 D4 — Interaction Style

Interaction Mode governs how sessions are structured, and Interaction Style governs how each prompt within them is framed. FORGE distinguishes two styles. A *conversational* prompt poses an open question and invites the AI to propose options or tradeoffs. A *directive* prompt specifies intent and asks the AI to carry it out. The same instructor shifts between the two within a single session.

The appropriate style depends on the decision at stake. While scope is still being settled, conversational prompts serve better, since the instructor is looking for options rather than directing execution. Once the task is repairing a draft against settled scope, directive prompts serve better, since the pedagogical judgment has already been made and the AI need only execute. Scoping usually happens early in development, so style often tracks the lifecycle phase, but the decision at stake determines the appropriate style. In the Term Project, conversational prompts reopened whether the project's premise made sense (Section 5.3.3), and Exam 3's refinement ran on directive prompts, such as an instruction to remove the enumeration of multiple-choice questions. Treating style as a dimension separate from mode lets an instructor recognize two failures. Conversational prompts that continue after scope is settled delay convergence on a final material, and directive prompts issued while scope is still open close off options the AI could have proposed.

4.6 D5 — Inquiry Direction

FORGE identifies three levels of Inquiry Direction, distinguished by who is entitled to ask questions in the collaboration. *Unidirectional* inquiry lets the Orchestrator ask questions of the AI, but the AI does not ask in return. *Latent bidirectional* inquiry introduces implicit questions, where the AI proposes options, alternatives, or tradeoffs that function as questions without being phrased as such. *Explicit bidirectional* inquiry adds direct clarifying questions, where the AI pauses before acting and asks what it needs to know.

The latent level reframes a familiar behavior. Mixed-initiative design and interface guidelines both advise an agent that cannot infer the user's intent to do less (Amershi

et al., 2019; Horvitz, 1999). That literature treats an option menu as the agent doing less, and D5 treats it as the agent asking. An option menu is a question the Orchestrator answers by choosing, but it does not signal uncertainty or ask about intent, so it lies between not asking and asking outright.

4.7 D6 — Task Allocation

Task Allocation describes how work is divided between the Orchestrator and the three AI roles. Tasks fall along a continuum with three zones. At the AI-leaning end are tasks governed by explicit rules, where syntactic accuracy and speed matter most. At the human-leaning end are tasks that demand contextual pedagogical judgment. Between them lies a negotiated zone of tasks the AI can draft but only the instructor can finalize, in which the instructor sets the intent, the AI produces candidates, and the instructor accepts or revises them. The boundaries between zones shift with the stakes of the material and with the instructor's familiarity with both the course content and the AI's current capabilities. Final Acceptance, the formal approval that an instructional material meets its requirements and is ready for students, always rests with the Orchestrator, whatever the artifact.

4.8 D7 — Input Curation

Input Curation describes how the source materials the AI agents work with are organized and scoped. We treat it as a dimension in its own right, since the Reader can work only with what is curated for it, and disorganized or out-of-scope inputs leave gaps the system cannot detect.

Curation has two parts. *Material organization* is the structure imposed on source materials before any interaction begins. In both cases, the instructor grouped inputs by deliverable, so that a single named path retrieves everything relevant to one material (Sections 5.2 and 5.3). Exam 3 drew on two parallel trees of prior exams and lecture decks. The Term Project drew on its own draft specification and successive revisions, a generated proposal template, and a lecture deck. *Per-interaction scoping* is the explicit naming of what the Reader should attend to in each session, and it narrows the organized collection to the files one task needs. Exam 3's opening prompts named the two prior-exam directories, then narrowed to the lecture decks and later to individual files (Section 5.2).

Scoping can also fail, and the repair shows what a scope must contain. When a Term Project session reviewed the wrong document, the instructor first told the agent which document it had been asked to review, and that instruction failed. The instructor then told the agent to read only that document and to compare the document with the other versions afterward, and that instruction succeeded. The successful version separated the scope from the comparison that followed, which shows that a scope must name both which materials the Reader may read and the order in which it reads them. Organization and scoping together determine what the rest of the framework operates on.

5 Case Studies

Exam 3 and the Term Project specification were developed by the same instructor, in the same course, with the same tool, and their development records are analyzed here through the seven dimensions.

5.1 Method

5.1.1 Research Design

Design science research treats the construction of an artifact, such as a framework, as the primary contribution in applied fields. Three of the criteria Hevner et al. set for such research govern how FORGE must be judged. The problem must be relevant and unsolved, the artifact's utility, quality, and efficacy must be demonstrated through well-executed evaluation, and both construction and evaluation must rest on rigorous methods (Hevner et al., 2004). The three conditions depend on one another.

A companion methodology supplies the process those criteria govern (Peffers et al., 2007). Design science sanctions case study as an evaluation method, the in-depth study of an artifact in its deployment environment (Hevner et al., 2004; Peffers et al., 2007). A case study demonstrates that a framework is sensitive to meaningful variation in its target domain while testing no hypothesis about population-level effects (Hevner et al., 2004; Peffers et al., 2007). Exam 3 (Section 5.2) and the Term Project (Section 5.3) serve this role, sharing an instructor, a course, and the architecture while differing in material type.

5.1.2 Data and Measures

The analysis rests on three definitions. A *session* is one Claude Code conversation rather than a contiguous sitting, since the instructor closed a session after one working block and resumed it days later. Session boundaries therefore mark changes of working context. *Active time* is the sum of intervals between consecutive prompts within a session. Any interval of 30 minutes or more is excluded as a break rather than truncated, so active-time totals are conservative lower bounds. *Substantive prompts* are those that remain after two screens. The first excludes interface commands, which carry no task content. The second attributes prompts to a case individually rather than by session, which excludes two trailing Term Project prompts that belong to a third deliverable. The screens leave 95 substantive instructor prompts for Exam 3 and 47 for the Term Project. The delivered materials serve as secondary sources. For Exam 3 they are the LaTeX source with its Version B and practice variants, and for the Term Project the specification, proposal template, and rubrics.

Each prompt was classified on four measures. *Generation versus response* codes a prompt as requesting generation if it asks for content that does not yet exist and as responding otherwise. A prompt directing a repair is therefore coded as responding, even though text is generated to satisfy it. *Path scoping* counts prompts naming an explicit filesystem path (D7). *Option selection* counts prompts that select from a menu the agent had offered, by a numbered or lettered option or a bare ordinal (D5). *Interaction style* classifies each prompt as conversational or directive (D4). The

first three measures were derived by pattern matching and verified by reading every matched prompt. Interaction style was coded by the instructor alone, so no inter-rater reliability is reported. The coding rule and the full coded set are in Online Resource 1.

A commercial LLM served as an analytic aid in preparing this analysis, recovering instructor prompts from the local prompt-history log, computing active-time totals, and producing the pattern-matched counts reported in the case studies. Every count and quotation was verified against the primary log. The same tool assisted revision of the manuscript, and the authors reviewed every change. All research questions, theoretical commitments, dimensions, guidelines, and interpretive claims originate with the authors.

5.2 Case Study 1: Exam 3 — Dense Refinement on a Convergent Material

5.2.1 Input Curation in Practice

Exam 3's inputs were organized before generation began and scoped at each interaction. The instructor staged two parallel directory trees, the first holding the two prior exams with their answer keys and a directory to receive the new exam, and the second holding the lecture decks each exam covered. Of the 95 instructor prompts, 15 (16%) named an explicit path, and six of these narrowed the scope to an individual file (Section 5.1). The second prompt of Session 1 set the initial scope by naming the course-material directory and telling the Reader to look at each exam's subdirectory, and later prompts narrowed it to single decks when a specific defect was in view. Scoping at this level made output traceable, as any generated question could be matched to the deck, code example, or prior item the Reader had read. It also limited the Reader to what the instructor had named.

5.2.2 Process Execution

Exam 3 was produced across five sessions between November 7 and 16, 2025, in 9 hours 28 minutes of active time and 95 prompts (Figure 3). Coding each prompt by what it asks the agent to do separates generation from response. Six of the 95 prompts request new artifact content, comprising the first draft and its immediate continuation, the practice exam, and three requests for Version B. Together they account for 32 minutes, or 5.6%, of active time. The remaining 89 prompts, which make up 94% of prompts and 94% of active time, respond to material that already existed, by identifying a defect, judging difficulty, selecting among proposed repairs, or directing a fix. Response work exceeds generation by a factor of 17.

The split between generation and response cuts across session boundaries, so it is reported per prompt rather than per session. Even Session 1, labeled "Initial Draft," spent 15 of 17 prompts responding to a draft that existed after the first exchange. On a convergent material the Writer produces a plausible draft at once, and nearly all remaining work is detecting and repairing defects the Writer cannot see in its own output.

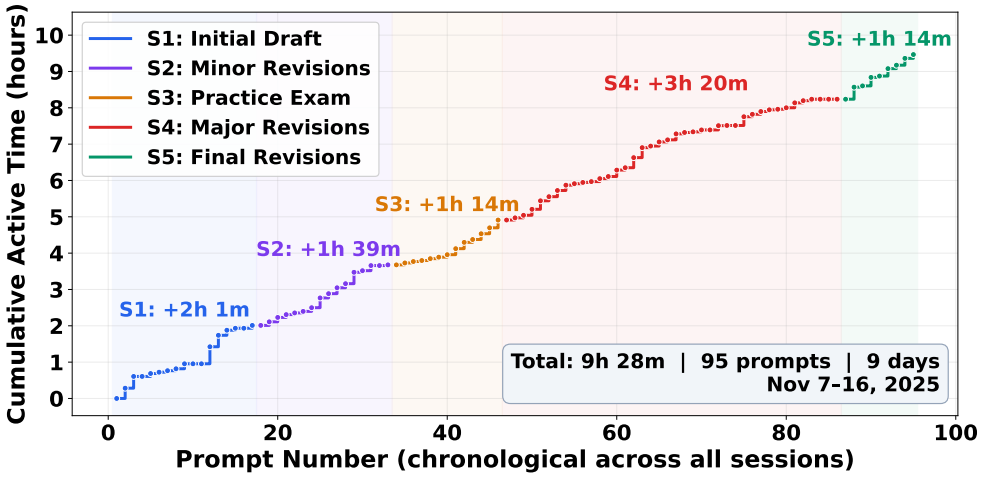

Fig. 3 Cumulative active time across Exam 3 development, grouped by the five sessions (S1–S5). Each point is one instructor prompt. The step height is the interval since the previous prompt, with intervals of 30 minutes or more excluded as breaks rather than truncated. Session segments are drawn in distinct colors and labeled with their active-time contribution. The slope is the instantaneous interaction density. The flat shaded regions between sessions show disengagement between work blocks. Session 4's steep slope reflects the verification-dominant phase of development.

5.2.3 Interaction Dynamics: The Pet-API Refinement

FORGE's interaction dynamics are clearest in a 23-minute episode from Session 4, in which one item, Question 1.3b, referred to here as the Pet-API item, passed through seven instructor turns. The initial AI-generated item showed students a complete FastAPI endpoint and asked which status code was returned when the pet was not found, while the code visibly contained `status_code=404`. The final item removed the exception handling from the code and asked students to choose the line that should be added (Figure 4).

Five of the seven turns show a named behavior, and the other two show the instructor provisionally accepting a flawed item. **Turn 1 (integration judgment):** The instructor identified the construct-irrelevant easiness, a defect only someone who knew what the item was meant to measure could see, noting that the code contained the 404 status and asking how the item could be improved. **Turn 2 (latent inquiry):** The Writer proposed three remediation options and the instructor chose one. This is the latent form of inquiry, in which ambiguity is presented as a menu. **Turn 3 (structure boundary):** The revised draft was syntactically correct but still leaked the answer through the `detail="Pet not found"` string, which the instructor detected.

The remaining turns alternated between stopping and reopening. **Turn 4:** The instructor stopped, writing "Lets keep it as is. Maybe we can remove the detail?" and provisionally accepting a question just judged defective. **Turn 5 (directive reopening):** The instructor reopened it with pedagogical intent rather than a surface fix, asking how the difficulty could be raised while keeping the content the same. **Turn 6:** The instructor stopped again, repeating "Lets keep it as is." **Turn 7 (integration judgment):** The instructor proposed removing the exception handling entirely and

Initial

```python
@app.get("/pets/{id}")
async def get_pet(
    id: int,
    session: Session = Depends(get_session)
) -> PetRead:
    pet = session.get(Pet, id)
    if pet is None:
        raise HTTPException(
            status_code=404,
            detail="Pet not found")
    return pet
```

Q: What status code is returned if the pet is not found?

<Short Answer>

7 turns → **23 minutes**

Final

```python
@app.get("/pets/{id}")
async def get_pet(
    id: int,
    session: Session = Depends(get_session)
) -> PetRead:
    pet = session.get(Pet, id)
    # Missing error handling here
    return pet
```

Q: What code should be added after `session.get(Pet, id)` to properly handle cases where the pet ID does not exist?

A) `if not pet: return None`
B) `if pet is None: raise HTTPException(status_code=404)`
C) `if pet == False: raise Exception("Not found")`
D) `if pet is None: return {"error": "Pet not found"}`

Fig. 4 Pet-API refinement worked example from Session 4 of Exam 3 development. The initial item (left) presents a short-answer question whose answer is visible in the accompanying code (`status_code=404`). Seven instructor turns over 23 minutes produced the final item (right), in which the exception handling is elided and the student must select the correct handler from four options. Option B (highlighted) is the correct answer. The instructor's integration judgment opened and closed the loop. Between those two judgments came a latent inquiry through an option menu, a leak that survived a syntactically correct repair, and a directive reopening. At turns 4 and 6 the instructor provisionally accepted the flawed item before reopening it.

asking students to supply it from a multiple-choice list. The Writer regenerated the item, and the instructor accepted it. The defect survived two decisions to retain the item, so integration-level judgment operated only intermittently against a draft that already appeared adequate.

An explicit question at the handoff would have shortened this loop. A Critic reviewing the initial draft against its code could have asked, "These items reference visible code. Do you want the answers visible, or should I hide them?", and surfaced the defect before the instructor reviewed the draft.

5.3 Case Study 2: Term Project — Incremental Generation on a Divergent Material

5.3.1 Input Curation in Practice

Exam 3 drew on a filesystem staged in full before generation began. The Term Project instead accumulated its inputs as the work proceeded, beginning in Session 1 from the instructor's own draft specification and admitting each later input when a question demanded it. Accumulating inputs had two consequences. First, several of the most consequential inputs were outputs of earlier sessions, so curation on a divergent material depends on its own earlier outputs. The proposal template generated in Session 5

became a scoping target in Sessions 6 and 8, and the revised specification became the standing reference thereafter. Second, version labeling became necessary as versions multiplied, so the Reader had to be told which version was authoritative at each turn.

Re-scoping across versions left a measurable trace. Of the 47 Term Project prompts, 12 (26%) named an explicit filesystem path, compared to 16% for Exam 3 (path scoping, Section 5.1). Every one of the 12 named a specific file rather than a directory. A convergent material can be scoped once by directory, with files named only as defects surface. A divergent one must be re-scoped by file as its own intermediate products multiply.

5.3.2 Process Execution

The work unfolded across 10 sessions between November 3 and November 29, 2025, totaling 5 hours 45 minutes of active time and 47 instructor prompts. The sessions fall into four work packages, each organized around one deliverable (Figure 5). Research covered sessions 1–2 (42 min, 7 prompts), Requirements sessions 3–6 (2 h 45 min, 22 prompts), Revisions sessions 7–8 (54 min, 11 prompts), and Slides & Rubrics sessions 9–10 (1 h 24 min, 7 prompts). Against the phases of D2, Research and Requirements together form Requirements Capture, Revisions is predominantly V&V, and Slides & Rubrics is a further cycle of Design and Generation on derivative materials. Requirements Capture therefore consumed 207 of 345 active minutes, or 60% of the total, inverting Exam 3's V&V-dominant distribution. On a divergent material, scoping is the central difficulty, which here meant deciding what APIs to offer, what SQL query types are feasible across them, and how strict the frontend requirement should be.

On the same coding, 11 of the 47 prompts request new content and account for 95 minutes, or 27% of active time. The remaining 36 prompts, which make up 77% of prompts and 73% of active time, respond to material that already existed. Response work exceeds generation by a factor of 2.6 here, compared to 17 for Exam 3. Generation also accounts for roughly four times as large a share of prompts on the divergent material as on the convergent one.

The 26-day span overstates the spread, as 40 of the 47 prompts fall within the first four days and the rest occupy two later sessions separated from that block and from each other by four and 19 days. Calendar time therefore does not distinguish the modes. The gaps instead show that a divergent specification can be released in pieces.

5.3.3 Interaction Dynamics: The SQL Requirements Arc

The divergent-material dynamics are clearest in a 52-minute arc from Session 4 in which the specification's SQL requirements were commissioned and then withdrawn across six instructor turns. **Turn 1 (incremental release):** The instructor asked the Writer to draft SQL query requirements tailored to each of the five API options and to present them for approval before any change to the specification file. **Turn 2 (latent inquiry):** Twenty-six minutes later the instructor replied with a bare ordinal, selecting from a menu the record does not preserve (Section 3). **Turn 3 (conversational reopening):** Rather than direct a revision, the instructor reopened scope, saying the options might need reducing to make grading easier and asking which three or four to keep. **Turn 4 (scoping):** The instructor settled on three options and named both

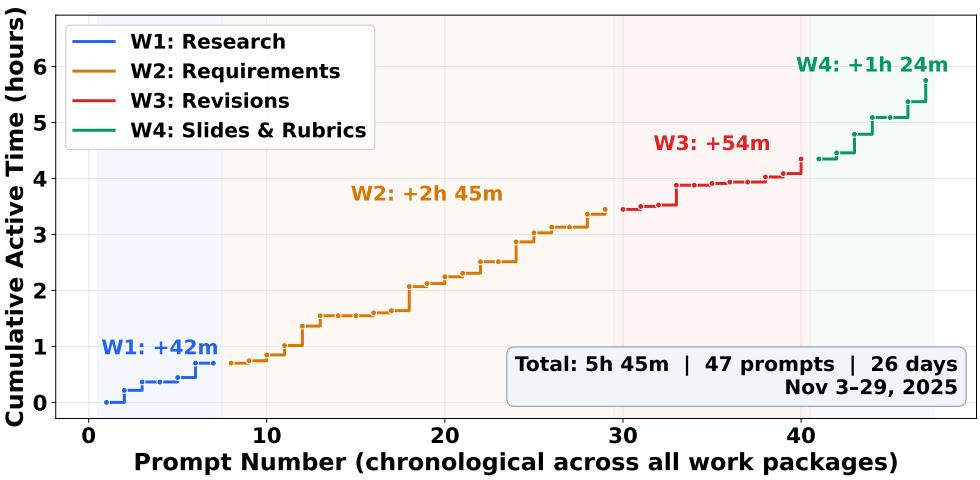

Fig. 5 Cumulative active time across Term Project development, grouped by the four work packages (W1–W4). Each point is one instructor prompt. The step height is the interval since the previous prompt, with intervals of 30 minutes or more excluded as breaks rather than truncated. Of the 47 prompts, 40 fall in the first four days. The flat region before W4 is a four-day gap, and the one inside W4 is the 19-day gap preceding the rubric session. W1 and W2 together constitute Requirements Capture under D2, which consumed 60% of active time.

the source file and the destination file. **Turn 5 (conversational reopening):** The instructor reopened the project's premise, asking whether having students draw on a public API to build their own API made sense. **Turn 6 (integration judgment):** The instructor cut what Turn 1 had commissioned, asking that the optional features under each option and the SQL query requirements section be removed or shortened.

The arc has no counterpart in Exam 3, whose criteria were fixed before the session opened. The Term Project's criteria moved, as five API options became three and a section commissioned 52 minutes earlier was removed once its cost in length became visible. On a divergent material, generated content stays provisional until scope settles.

5.3.4 Interaction Dynamics Against Exam 3

The two cases differ first in how densely the instructor worked within a session. Exam 3 averaged roughly 19 instructor prompts per session, while the Term Project averaged 4.7. Each Term Project session closed when a question was answered rather than when a draft converged, since a dense refinement loop has nothing to refine while requirements are still being defined.

The cases differ more sharply in interaction style but not in inquiry direction. Conversational prompts, which seek judgment, made up 4 of 95 prompts (4%) for Exam 3 and 12 of 47 (26%) for the Term Project, a factor of six. Prompts that selected from an option menu the AI had offered, the latent form of inquiry, were about as common in each case, 16 of 95 (17%) for Exam 3 and 7 of 47 (15%) for the Term Project. The timing of the conversational prompts is more informative than their count. The Term Project's 12 cluster in Sessions 3 through 8, while scope was contested, and none occur in Sessions 9 and 10, where slides and rubrics were produced

against a specification the instructor had by then settled. Style shifted toward directive prompts once scope had settled.

Exam 3 complicates the pattern. Its four conversational prompts all occur in the final two sessions, once a complete draft existed, among them a question whether the number of multiple-choice items should be reduced. Read against calendar position, this finding reverses the Term Project's pattern. Read against the decision at stake, it does not, since these prompts reopen what the exam should measure rather than correct a defect in its wording. Style tracks whether the instructor is settling scope or repairing an artifact, and scope can be reopened even after a complete draft exists.

5.4 Outcomes

Both materials reached students in the forms the sessions produced. Exam 3 was delivered in two versions with a matched practice exam, including the Pet-API item as its seventh turn left it (Figure 4). The Term Project was delivered as a specification, a proposal template, two rubrics, and a slide outline, each released as it settled. No comparable measurement exists for conventional development of either material, although the instructor recalls Exams 1 and 2 as having taken between 15 and 25 hours each. In both cases, the allocation of tasks held to the FORGE continuum. One Term Project decision, whether to require version control of students who might never have used it, depended on a judgment about the student population that no agent could make. Both cases show that the framework can be executed through all three phases, with its prescribed division of labor applied to materials that reached students.

5.5 Comparative Summary

Both cases exercise the full FORGE architecture (D1), depend on curated per-interaction-scoped inputs (D7), and share the same inquiry direction (D5). They nevertheless produce different profiles on the procedural, interactional, and allocative dimensions (Figure 6). Exam 3 spent most of its active time on V&V, while the Term Project spent most of its time on Requirements Capture (D2). Exam 3 ran as Dense Refinement and the Term Project as Incremental Generation (D3). Conversational prompts were about six times as common in the Term Project as in Exam 3 (D4). The one departure from the shared architecture was an ad-hoc Researcher the Term Project added for API discovery in each of its opening two sessions.

On D6, the tasks of both cases sort onto the continuum defined in Section 4.7, with formatting and Version B shuffling at the AI-leaning end and difficulty calibration and hint detection at the human-leaning end. The negotiated boundary lay at different points, at pedagogical calibration for the exam and at scope-and-content choice for the specification (Figure 7).

6 Guidelines

The seven FORGE guidelines (G1–G7) answer the questions the dimensions raise (Figure 2). They fall into five groups, with G1 foundational, G2 and G3 concerned

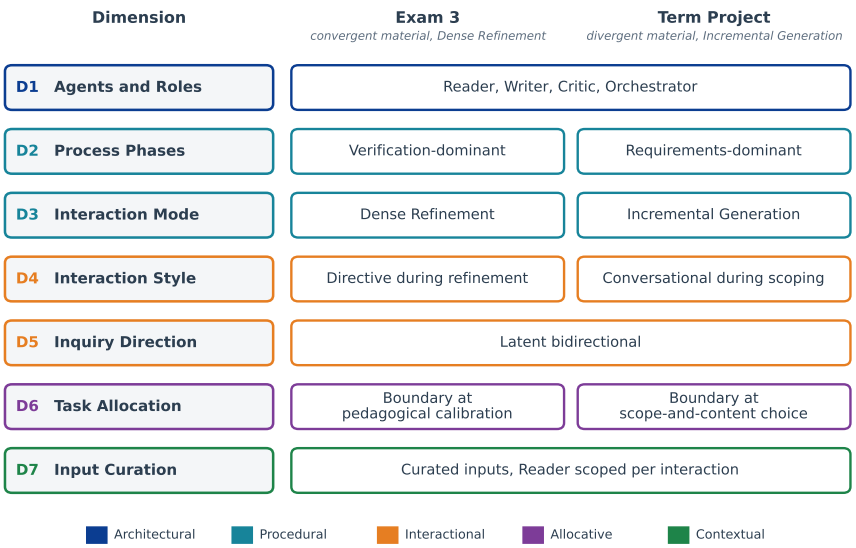

Fig. 6 Both cases mapped onto the seven FORGE dimensions. D1, D5, and D7 hold across the two cases, while D2, D3, D4, and D6 take different values with the material type.

with allocation, G4 with mode, G5 and G6 with interaction, and G7 with integration. Each is stated as a rule and grounded in the mechanism and literature that support it.

6.1 G1 — Curate and Scope Input Materials

Organize the source material the AI will read into a consistent, versioned filesystem, and scope the Reader to a named subset of it at each prompt. The curated material is a component of the system that performs the work rather than preparation for it (Section 2.5). Exam 3 staged its inputs in full before generation and scoped by directory, naming files only when a specific defect was in view. The Term Project could not. Its inputs accumulated as questions arose, and its own intermediate versions became what later sessions had to be scoped against. Path scoping therefore rose from 16% to 26% of prompts, and every scope narrowed to a file (Sections 5.2 and 5.3). Staging inputs in full is thus available only when the deliverable's shape is known in advance, and on divergent material the achievable target is that each session names which version is authoritative.

6.2 G2 — Separate Structure from Substance

Allocate work governed by explicit rules to the AI and retain pedagogical judgment for the human, so that each component carries what it is best positioned to carry (Section 2.5). The Pet-API item shows where the line falls, since a syntactically correct draft still leaked the answer through a string that only the instructor could read as a leak (Section 5.2.3). The same line appears in an experiment outside education, where

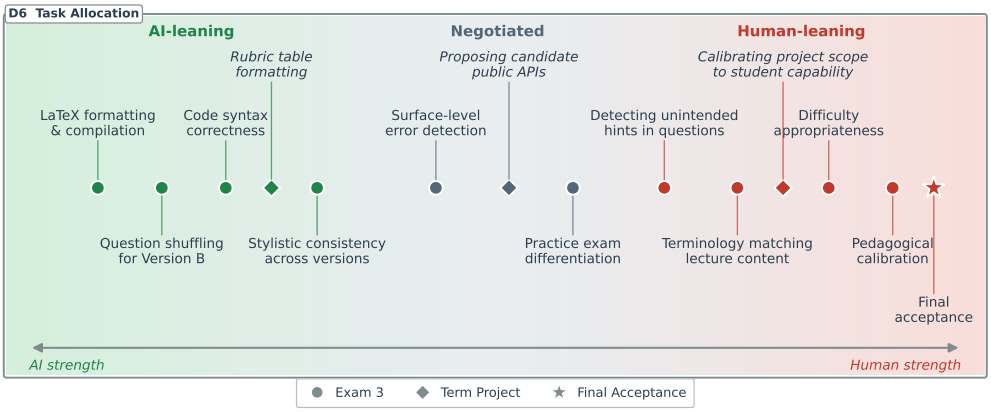

Fig. 7 The FORGE task allocation continuum (D6). Tasks observed during Exam 3 development (circles) and Term Project development (diamonds) are positioned along a continuum from AI strength to human strength. AI-leaning tasks (green) are governed by explicit rules, where syntactic accuracy and speed matter most. Human-leaning tasks (red) demand contextual pedagogical judgment, and tasks in the negotiated zone (slate) can be drafted by the AI but finalized only by the instructor. Final Acceptance (star) is the Orchestrator's formal approval of the finished material, common to both cases.

checkers without domain knowledge caught 82% of spelling and digit errors in AI-generated output but only 31% of errors that depended on the task's rules (Beck et al., 2026).

6.3 G3 — Budget for Verification, Not Generation

Allocate time and effort to verification rather than to initial generation (Section 2.5). In Exam 3, nearly all active time went to detecting and repairing defects the Writer could not see in its own output (Section 5.2, Figure 3). Pedagogical calibration also took place during verification, since difficulty tuning, terminology alignment, and validity checks all require a draft to examine. Verification dominated the Term Project as well, where response work exceeded generation by a factor of 2.6 (Section 5.3).

The dominance of verification over generation extends beyond this course and predates LLMs. In an undergraduate operating systems course, most LLM-generated questions failed the instructor's validation, and for five questions no acceptable counterpart emerged after many attempts (Witsken et al., 2025). In AIG, review remains the most time-consuming step once generation volume grows (Circi et al., 2023). Template-based AIG places much of this review effort before generation, in the design of the item templates (Gierl et al., 2012). Unstructured prompting, as used in both cases, leaves the review effort until after a draft exists. Verification omitted during development cannot be recovered after an exam is administered, since item-writing defects introduce systematic error that internal-consistency reliability does not detect (Downing, 2005). The verification effort G3 recommends nonetheless has an upper limit. In systems engineering, process effort improves program outcomes up to an optimum near 14% of program cost and degrades them beyond it (INCOSE, 2015).

6.4 G4 — Match Interaction Mode to Material Type

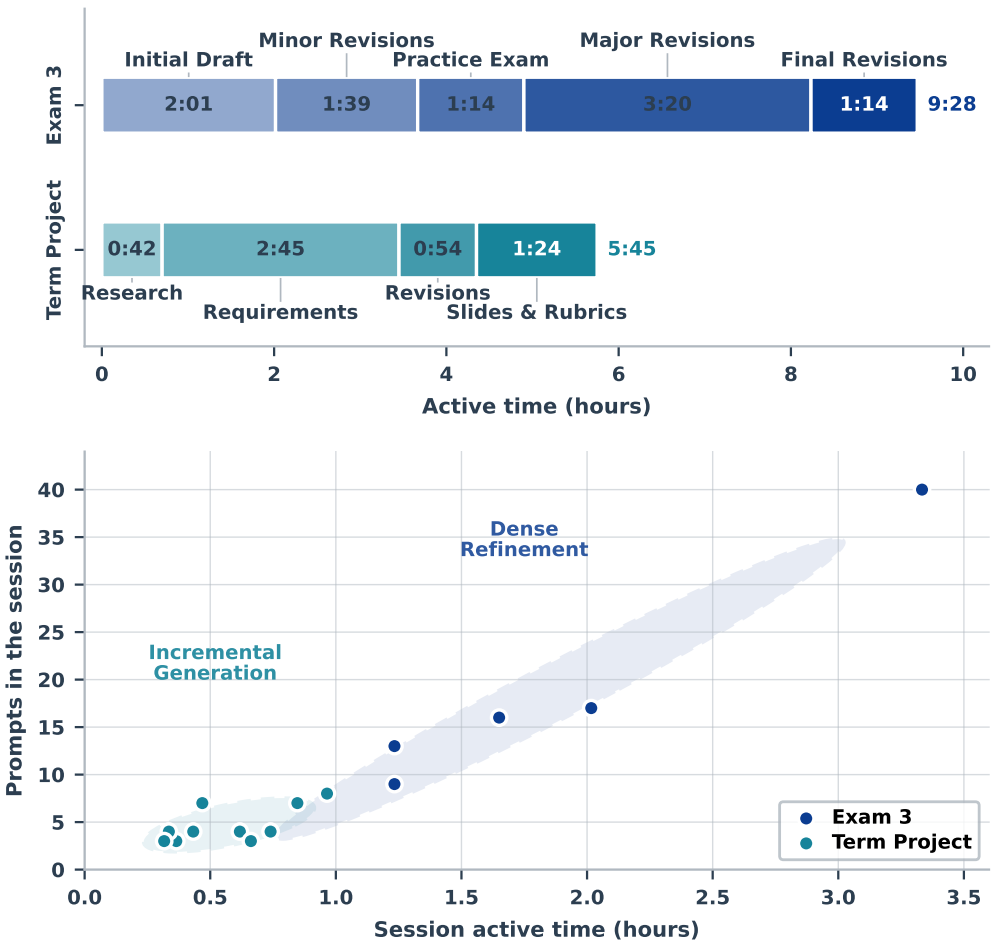

Fig. 8 Interaction mode in the two cases. Top: active time by session for Exam 3 and by work package for the Term Project. Bottom: each session by its active time and instructor prompt count, with Dense Refinement and Incremental Generation as the two clusters.

Choose the interaction mode (D3) to match whether the material is convergent or divergent. Dense Refinement suits a convergent material, whose completion state is known in advance. Incremental Generation suits a divergent material, whose scope is settled one piece at a time. The two cases show the two modes as distinct patterns of work, with Exam 3 developed in five long sessions of many prompts and the Term Project in ten short sessions of few prompts (Figure 8). A mismatch between mode and material can fail in two ways. Dense Refinement on a divergent material risks premature closure, in which the scope is fixed before its pieces are understood. Incremental Generation on a convergent material risks under-refinement, in which

items are released before they are fully refined. Premature closure has a mechanism in distributed cognition, where tightly coupled groups converge faster and then resist disconfirming evidence (Hutchins, 1995). The risk of under-refinement has a parallel in a study that deliberately withheld iteration when generating exam questions. Any conversation whose output failed review was abandoned, and the authors named refinement as the likely way to align generated items with the instructor's own phrasing (Witsken et al., 2025).

6.5 G5 — Shift Interaction Style with the Decision at Stake

Shift interaction style (D4) with the kind of decision at stake. Conversational prompts suit decisions about scope, and directive prompts suit the repair of a draft whose scope is settled. In both cases, interaction style followed the decision at stake, and calendar position did not predict it. The Term Project's 12 conversational prompts fell in Sessions 3 through 8, while scope was contested. Exam 3's four conversational prompts came only after a complete draft existed, and each reopened what the exam should measure (Section 5.3). An instructor who reopens scope late in development should therefore return to conversational prompts.

6.6 G6 — Make Inquiry Bidirectional at Phase Gates

Configure the AI to ask the instructor an explicit question at each phase gate, where Requirements Capture hands off to Design and Generation and where Design and Generation hands off to V&V. Both cases showed only the latent form of inquiry, in which the AI offered option menus after a draft existed. In the Pet-API refinement, one explicit question from the Critic at handoff could have revealed the leaked answer before the instructor reviewed the draft (Section 5.2.3).

Mixed-initiative design already advises an agent unsure of the user's intent to ask, provided the question is worth the interruption (Horvitz, 1999). A question also differs from an explanation, which raised acceptance of AI recommendations whether or not they were correct (Bansal et al., 2021). Interrupting the user before AI output could be accepted reduced overreliance on incorrect output, even though users disliked the interruption (Romeo & Conti, 2026). G6 adds a fixed point at which the AI asks. Mixed-initiative design leaves that point to the agent's estimate of its own uncertainty, which an LLM does not reliably maintain. G6 places the question at the phase gates instead, where an unasked question costs the most, since each phase builds on the output of the one before it.

6.7 G7 — Preserve the Human as Systems Integrator

Preserve human authority over cross-cutting, integration-level decisions, and treat Final Acceptance as non-delegable. Integration-level decisions govern the properties that no single AI role owns, namely material coherence, equivalence across versions, and pedagogical alignment (Section 2.5). In the Pet-API refinement, only the instructor could see that a visible status code reduced the item to pattern matching. Only the instructor could then specify the intent that let the regenerated item test exception handling (Section 5.2.3).

Better models are unlikely to remove the need for a human integrator, since integration-level decisions are difficult for human experts as well. Plausible distractors are difficult to write, even for professional item writers (Section 2.4). Three widely used concept inventories built their distractors on known misconceptions, yet none could show which misconceptions a student held (Jorion et al., 2015). Guidance for multi-agent frameworks also advises keeping a human in the loop, even when the intended end state is more autonomous (Wu et al., 2023). Each guideline places a specific demand on the instructor and answers one dimension of the design space (Figure 9).

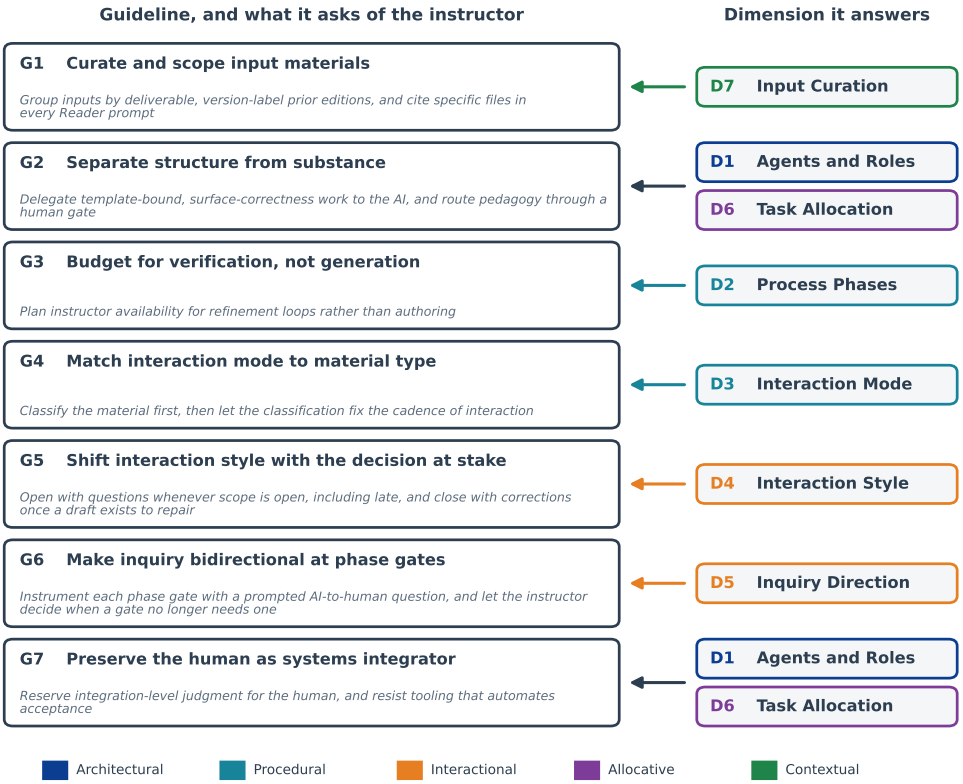

Fig. 9 The seven FORGE guidelines, what each asks of the instructor in practice, and the dimension each answers. Color marks the five categories.

7 Discussion

7.1 What FORGE Adds

FORGE makes four contributions. The first is a descriptive vocabulary for the human-AI exchange. Studies that treat AI assistance as a one-shot generation task reduce that exchange to a single variable, "using ChatGPT" or "prompting." Such studies

therefore lack the vocabulary to distinguish a productive exchange from an unproductive one (Section 2.1). Interaction Style (D4) and Inquiry Direction (D5) supply two of the missing parameters. An instructor whose progress stalls during V&V can then identify a mismatch between prompts that remain directive and a decision that has reopened scope. The second is a prescriptive set of guidelines, G1–G7, stated in terms of materials, phases, and judgment. The guidelines carry no course-specific content and may therefore apply wherever instructional materials are developed. The third is alignment with systems engineering. FORGE's three phases correspond to the distinct requirements, design, and V&V processes of systems engineering. G3 bounds verification effort at the optimum that systems engineering reports, and G7 assigns the instructor the role of systems integrator (INCOSE, 2015). FORGE therefore brings an established process model to a task previously left to improvisation. The fourth is a low cost of adoption. Both cases realized the Reader, Writer, and Critic within one shared LLM session through role-scoped prompting. Role-scoped prompting depends on no tool-specific feature and therefore reduces what an instructor must adopt.

7.2 Limitations

The demonstration's setting is narrow. FORGE was applied by a single instructor in a single undergraduate information systems engineering course (IEE 305, Fall 2025). Variation in how other instructors and other disciplines would apply the framework remains unobserved. The two case studies meet the design science standard of showing utility in representative settings. They are not a statistical sample, however, and support no population-level estimate.

The evaluation carries its own limits. Surface-level checking was delegated to the Critic, and deeper pedagogical verification stayed with the instructor. This division records where the instructor set the boundary and does not measure what the Critic could have done. No student-facing measurement was collected, so the study reports how the materials were produced and not how they performed once administered. The study therefore cannot determine whether AI-assisted materials teach or assess differently from conventionally developed ones. In a comparable undergraduate course, students as a group could not identify which questions had been written with AI, yet scored about nine percentage points lower on those questions (Witsken et al., 2025). The presence of a similar effect in Exam 3's items is unknown. The analysis is retrospective and rests on instructor prompts alone, so no claim about agent behavior can be checked against the record (Section 3).

Three design choices in FORGE remain untested. The G6 prescription is grounded in the Exam 3 refinement loop but has not been tested against a control. Both cases used one commercial LLM tool at one pricing tier, so portability across model families and pricing tiers is assumed on architectural grounds. Role scoping within one session leaves the three AI roles sharing a context, so the Critic evaluates a draft it has already seen produced. However, shared context also carries criteria forward from one instruction to later work (Hashem et al., 2024). Separating the roles is therefore a tradeoff, and its effect on defect detection is unknown.

The role set and the order of work carry two further limits. The four-role set may be incomplete, since the Term Project twice invoked a Researcher agent for a discovery task the Reader is not defined to perform (Section 5.5). Exam 3's workflow also presented the Writer's complete draft before the instructor had formed an independent judgment. That order risks anchoring, in which the first draft seen sets the reference for every later judgment and is accepted more readily than its quality warrants (Bansal et al., 2021). The size of that risk is contested (Beck et al., 2026; Romeo & Conti, 2026). Experiments outside education reduced it by withholding the AI's output until the human had committed to a judgment (Romeo & Conti, 2026). None of the limits in this section affects the prompt counts and timings reported for both cases (Section 5).

8 Conclusion

FORGE was demonstrated in two case studies in one undergraduate course. Exam 3 was developed by Dense Refinement on a convergent material, and the Term Project specification by Incremental Generation on a divergent one. Of the seven dimensions that define the design space, Interaction Style and Inquiry Direction are the two that current practice leaves underspecified. Under the same architecture, the two materials differed on the procedural, interactional, and allocative dimensions, while inquiry direction and input curation remained constant. Conversational prompts, which seek judgment, made up 26% of Term Project prompts, compared to 4% of Exam 3 prompts. Seven guidelines follow from that variation, and one set describes both materials. The guidelines align with the lifecycle and the integrator role of systems engineering.

Four lines of future work follow. FORGE's architectural grounding suggests that it transfers across disciplines, and comparative evaluation in other departments would test that expectation. G6's prescription of an explicit AI question at each phase gate can be falsified by a controlled comparison against unidirectional and latent bidirectional inquiry. An AI Orchestrator whose only human gate is Final Acceptance would test whether G7's allocation is necessary or only sufficient. Finally, wider evaluation should extend to cohorts of instructors, to program-level coordination, and to direct measurement of student learning outcomes (Section 7.2).

Generative AI adds refinement steps to open-ended design that instructors would not otherwise take (Kim, 2026), and the FORGE guidelines turn those steps into decisions an instructor can plan for. The first decisions are to classify the next deliverable as convergent or divergent (G4), curate and scope its inputs (G1), and budget V&V effort to match (G3). Coordinators and researchers can use the dimensions as a vocabulary for comparing sections, semesters, and studies, provided session records retain agent responses alongside instructor prompts.

Instructors already develop course materials with generative AI, largely without a shared method for the collaboration. FORGE gives that collaboration a vocabulary for describing it, guidelines for conducting it, and a grounding in the process models of systems engineering. With FORGE, an instructor can describe how a material was developed with AI, keep what worked, and carry it to the next deliverable.

Supplementary information. The instructor prompt records, session timing data, interaction-style coding, and figure-generation scripts supporting this article are provided as Online Resource 1, one archive of eight files.

Acknowledgments. We thank the Fall 2025 students of Information Systems Engineering (IEE 305) in the School of Computing and Augmented Intelligence, Ira A. Fulton Schools of Engineering, Arizona State University, for whom the materials analyzed here were developed. We also thank attendees at the 2026 Conference on Systems Engineering Research for questions and discussion that sharpened the framework. In particular, we thank Dr. Jon Wade, whose questions about conversational versus directive styles of human-AI interaction directly motivated D4 (Interaction Style) and G5 (Shift Interaction Style with the Decision at Stake).

Declarations

Funding. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

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

Ethics approval and consent to participate. This study analyzes the instructor's own records of instructional material development. No student data, coursework, or performance information was collected or analyzed, and no human subjects research was conducted, so institutional review board approval was not required.

Consent for publication. Not applicable.

Data availability. The instructor prompt records, session timing data, interaction-style coding, and figure-generation scripts supporting this article are provided as Online Resource 1. Agent responses were not retained beyond the sessions in which they occurred and are therefore not available.

Code availability. The figure-generation scripts are included in Online Resource 1.

Author contributions. Emmanuel M. Gonzalez: conceptualization, methodology, formal analysis, investigation, data curation, visualization, writing (original draft), writing (review and editing). Paul T. Grogan: supervision, writing (review and editing). Both authors read and approved the final manuscript.

References

Albluwi, I. (2019). Plagiarism in programming assessments: A systematic review. *ACM Transactions on Computing Education*, *20*(1), 6, https://doi.org/10.1145/3371156

Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P.N., Inkpen, K., Teevan, J., Kikin-Gil, R., Horvitz, E. (2019).

Guidelines for human-AI interaction. *Proceedings of the 2019 CHI conference on human factors in computing systems (CHI '19)* (pp. 1–13). https://doi.org/10.1145/3290605.3300233

Banihashem, S.K., Noroozi, O., van Ginkel, S., Macfadyen, L.P., Biemans, H.J.A. (2022). A systematic review of the role of learning analytics in enhancing feedback practices in higher education. *Educational Research Review*, *37*, 100489, https://doi.org/10.1016/j.edurev.2022.100489

Bansal, G., Wu, T., Zhou, J., Fok, R., Nushi, B., Kamar, E., Ribeiro, M.T., Weld, D.S. (2021). Does the whole exceed its parts? The effect of AI explanations on complementary team performance. *Proceedings of the 2021 CHI conference on human factors in computing systems.* https://doi.org/10.1145/3411764.3445717

Barnes, M., & Tour, E. (2026). Teachers' use of generative AI: A 'dirty little secret'? *Language and Education*, *40*(1), 24–39, https://doi.org/10.1080/09500782.2025.2485935

Beck, J., Eckman, S., Kern, C., Kreuter, F. (2026). Bias in the loop: How humans evaluate AI-generated suggestions. *Harvard Data Science Review*, *8*(2), https://doi.org/10.1162/99608f92.0e98898d

Bond, M., Khosravi, H., De Laat, M., Bergdahl, N., Negrea, V., Oxley, E., Pham, P., Chong, S.W., Siemens, G. (2024). A meta systematic review of artificial intelligence in higher education: A call for increased ethics, collaboration, and rigour. *International Journal of Educational Technology in Higher Education*, *21*, 4, https://doi.org/10.1186/s41239-023-00436-z

Cheung, B.H.H., Lau, G.K.K., Wong, G.T.C., Lee, E.Y.P., Kulkarni, D., Seow, C.S., Wong, R., Co, M.T.-H. (2023). ChatGPT versus human in generating medical graduate exam multiple choice questions—a multinational prospective study (Hong Kong S.A.R., Singapore, Ireland, and the United Kingdom). *PLOS ONE*, *18*(8), e0290691, https://doi.org/10.1371/journal.pone.0290691

Chowa, S.S., Alvi, R., Rahman, S.S., Rahman, M.A., Raiaan, M.A.K., Islam, M.R., Hussain, M., Azam, S. (2025). *From language to action: A review of large language models as autonomous agents and tool users.* arXiv preprint 2508.17281. https://doi.org/10.48550/arXiv.2508.17281

Circi, R., Hicks, J., Sikali, E. (2023). Automatic item generation: Foundations and machine learning-based approaches for assessments. *Frontiers in Education*, *8*, 858273, https://doi.org/10.3389/feduc.2023.858273

del Gobbo, E., Guarino, A., Cafarelli, B., Grilli, L., Limone, P. (2023). Automatic evaluation of open-ended questions for online learning. A systematic mapping. *Studies in Educational Evaluation*, *77*, 101258, https://doi.org/10.1016/j.stueduc.2023.101258

Downing, S.M. (2005). The effects of violating standard item writing principles on tests and students: The consequences of using flawed test items on achievement examinations in medical education. *Advances in Health Sciences Education*, *10*(2), 133–143, https://doi.org/10.1007/s10459-004-4019-5

Gierl, M.J., Lai, H., Turner, S.R. (2012). Using automatic item generation to create multiple-choice test items. *Medical Education*, *46*(8), 757–765, https://doi.org/10.1111/j.1365-2923.2012.04289.x

Haladyna, T.M., Downing, S.M., Rodriguez, M.C. (2002). A review of multiple-choice item-writing guidelines for classroom assessment. *Applied Measurement in Education*, *15*(3), 309–334, https://doi.org/10.1207/s15324818ame1503_5

Hashem, R., Ali, N., El Zein, F., Fidalgo, P., Abu Khurma, O. (2024). AI to the rescue: Exploring the potential of ChatGPT as a teacher ally for workload relief and burnout prevention. *Research and Practice in Technology Enhanced Learning*, *19*, 23, https://doi.org/10.58459/rptel.2024.19023

Hevner, A.R., March, S.T., Park, J., Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105, https://doi.org/10.2307/25148625

Hollan, J., Hutchins, E., Kirsh, D. (2000). Distributed cognition: Toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction*, *7*(2), 174–196, https://doi.org/10.1145/353485.353487

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. *Proceedings of the SIGCHI conference on human factors in computing systems (CHI '99)* (pp. 159–166). https://doi.org/10.1145/302979.303030

Hutchins, E. (1995). *Cognition in the wild*. MIT Press. https://doi.org/10.7551/mitpress/1881.001.0001

Ibrahim, L., Collins, K.M., Kim, S.S.Y., Reuel, A., Lamparth, M., Feng, S., Ahmad, Z., Soni, N., El Kattan, A., Stein, A., Swaroop, S., Padmakumar, V., Sucholutsky, I., Strait, M., Yang, Q., Liao, Q.V., Bhatt, U. (2026). *Measuring and mitigating overreliance to build human-compatible AI.* arXiv preprint 2509.08010. https://doi.org/10.48550/arXiv.2509.08010

INCOSE (2015). *Systems engineering handbook: A guide for system life cycle processes and activities* (4th ed.). Wiley. (INCOSE-TP-2003-002-04)

Jorion, N., Gane, B.D., James, K., Schroeder, L., DiBello, L.V., Pellegrino, J.W. (2015). An analytic framework for evaluating the validity of concept inventory claims. *Journal of Engineering Education*, *104*(4), 454–496, https://doi.org/10.1002/jee.20104

Kim, W.J. (2026). Teachers' use of generative artificial intelligence for designing science lessons in support of environmental science agency. *Research in Science Education*, *56*(1), 203–222, https://doi.org/10.1007/s11165-025-10262-0

Lo, C.K. (2023). What is the impact of ChatGPT on education? A rapid review of the literature. *Education Sciences*, *13*(4), 410, https://doi.org/10.3390/educsci13040410

Megahed, F.M., Chen, Y.-J., Ferris, J.A., Knoth, S., Jones-Farmer, L.A. (2024). How generative AI models such as ChatGPT can be (mis)used in SPC practice, education, and research? An exploratory study. *Quality Engineering*, *36*(2), 287–315, https://doi.org/10.1080/08982112.2023.2206479

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

Mhlanga, D. (2023). *Open AI in education, the responsible and ethical use of ChatGPT towards lifelong learning.* SSRN preprint 4354422. https://doi.org/10.2139/ssrn.4354422

Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77, https://doi.org/10.2753/MIS0742-1222240302

Romeo, G., & Conti, D. (2026). Exploring automation bias in human–AI collaboration: A review and implications for explainable AI. *AI & SOCIETY*, *41*(1), 259–278, https://doi.org/10.1007/s00146-025-02422-7

Sallam, M. (2023). ChatGPT utility in healthcare education, research, and practice: Systematic review on the promising perspectives and valid concerns. *Healthcare*, *11*(6), 887, https://doi.org/10.3390/healthcare11060887

Shi, Y., Yu, K., Dong, Y., Chen, F. (2026). Large language models in education: A systematic review of empirical applications, benefits, and challenges. *Computers and Education: Artificial Intelligence*, *10*, 100529, https://doi.org/10.1016/j.caeai.2025.100529

Streveler, R.A., Litzinger, T.A., Miller, R.L., Steif, P.S. (2008). Learning conceptual knowledge in the engineering sciences: Overview and future research directions. *Journal of Engineering Education*, *97*(3), 279–294, https://doi.org/10.1002/j.2168-9830.2008.tb00979.x

Wang, J., Xiao, R., Hou, X., Stamper, J. (2025). *Enabling multi-agent systems as learning designers: Applying learning sciences to AI instructional design.* arXiv preprint 2508.16659. https://doi.org/10.48550/arXiv.2508.16659

Wang, Z., Liu, M., Islam, A.Y.M.A. (2026). Reimagining teacher-AI co-design in learning task design: Trends and perspectives. *Humanities and Social Sciences Communications*, *13*, 757, https://doi.org/10.1057/s41599-026-06981-y

Witsken, G., Crk, I., Gultepe, E. (2025). *LLMs in the classroom: Outcomes and perceptions of questions written with the aid of AI.* arXiv preprint 2503.18995. https://doi.org/10.48550/arXiv.2503.18995

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A.H., White, R.W., Burger, D., Wang, C. (2023). *AutoGen: Enabling next-gen LLM applications via multi-agent conversation.* arXiv preprint 2308.08155. https://doi.org/10.48550/arXiv.2308.08155

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ESM1.zip