

Instructional Governance by Design

A Framework for AI in Computing Education

Ethan Dickey | Purdue University, Computer Science | dickeye@purdue.edu

A practical reference: 11 tool profiles and design lessons.

Governance fit matches AI authority, learner work and human responsibility to the instructional purpose. Use these comparisons to choose mechanisms and evaluation priorities, not to rank tools by effectiveness. [1]

Seven portfolio cases

Six dimensions: 1 Pedagogical grounding | 2 AI instructional authority | 3 Human accountability and control | 4 Learner agency and cognitive engagement | 5 Context specificity and boundary setting | 6 Evaluation visibility and revision

Tool / activity	Grounding + context Dimensions 1 + 5	AI authority + human control Dimensions 2 + 3	Learner work + visibility Dimensions 4 + 6
GLOW GTA rehearsal	Office-hour conversation rubric. Instructor-authored scenarios and student personas for GTA preparation.	AI: Simulates students and gives rubric-aligned formative feedback. Humans: Staff interpret analytics and provide coaching.	Work: GTAs rehearse difficult conversations. Evidence: Rubric-domain dashboards, readiness and coaching needs.
Owlgorithm Post-submission reflection	Self-regulated learning and Bloom-aligned questions. Prompts use the submission outcome and student code.	AI: Asks metacognitive questions after accepted, partial or unsuccessful submissions. Humans: Students retain reasoning and debugging.	Work: Explain, debug, compare strategies and revise. Evidence: Student ratings and TA feedback on prompt quality and usefulness.
BoilerTAI Course-forum support	Course norms for response quality, tone and conceptual explanation.	AI: Drafts candidate replies. Humans: Staff review, edit, reprompt and approve publication.	Work: Student reasoning task not specified in this profile. Evidence: Response time, staff edits, student satisfaction and TA workload.
BoilerSketch Conceptual diagrams	Instructor- and TA-vetted CS representations. Graph and diagram conventions guide conceptual explanations.	AI: Generates Mermaid-style diagrams and explanatory support. Humans: TAs supervise and vet outputs before instructional use.	Work: Student reasoning task not specified in this profile. Evidence: Staff judgments of representational clarity and diagram quality.
CodeStylist Code-style coaching	Local style guides, readability expectations and assignment-level course conventions.	AI: Suggests style changes and explanations. Humans: Instructors configure the norms. Students choose revisions.	Work: Revise one's own code. Evidence: Style consistency, uptake, clarity and fit to course expectations are evaluation targets.
AI-Lab Responsible-use onboarding	Structured activities on prompting, critique, debugging, limitations and mindful AI use.	AI: A tool to use and an object to critique. Humans: Instructors frame appropriate uses and connect activities to course goals.	Work: Assess outputs and reflect on skill development. Evidence: Surveys and focus groups on perceptions and usage patterns.
GAIDE Instructor content development	Constructivist and TPACK-informed content-development workflow connects pedagogy, content and technology.	AI: Brainstorms, drafts, varies and refines materials. Humans: Instructors judge pedagogical quality and adoption.	Work: Instructor design and revision, not a direct student task. Evidence: Reproducible prompts, examples and materials support review and revision.

Reading the evidence column. Entries identify traces or evaluation targets, not measured benefits. “Not specified” describes the paper’s profile, not proof that a tool lacks learner activity. GTA = graduate teaching assistant. TPACK = technological pedagogical content knowledge.

What the comparison shows. Human control can mean interpreting analytics, approving replies, vetting diagrams, configuring norms or choosing materials. Specify who does that work and when, rather than relying on “human in the loop.” [1, §3]

Full profiles and original studies: arxiv.org/abs/2609.26098 | External systems and design guide on the reverse.

Four published systems beyond the portfolio

The paper applies the same six dimensions to four published systems selected for contrasting mechanisms. These are analytical classifications, not claims that the systems were built using the framework. [1, §4]

Tool / activity	Grounding + context Dimensions 1 + 5	AI authority + human control Dimensions 2 + 3	Learner work + visibility Dimensions 4 + 6
CodeHelp [2] Guarded programming help	Programming-help norms distinguish productive guidance from direct solution disclosure. Instructors judge course fit.	AI: Explains, hints and guides within response boundaries. Humans: Instructors decide whether those boundaries suit the course.	Work: Students construct and debug their own solutions. Evidence: Helpfulness, solution leakage, reliance and educator trust are evaluation targets.
CodeAid [3] Balanced classroom assistance	Large-course support balances over-assistance, correctness, transparency and student control.	AI: Provides programming help through a classroom interface. Humans: Control is expressed through constraints, transparency and course expectations.	Work: Preserve engagement in solution construction. Evidence: Inspect request and response boundaries, reliance and educator visibility.
CodeTailor [4] Personalized Parsons puzzles	Parsons puzzles scaffold code ordering and reasoning. The generated activity uses student context.	AI: Transforms context into a structured practice artifact. Humans: Choose this curricular form for assistance.	Work: Assemble, reason about and test a solution. Evidence: The generated puzzle is inspectable for its suitability as a scaffold.
BISCUIT [5] Interface-mediated notebook coding	Computational-notebook work connects natural-language intent, generated code, execution and interpretation.	AI: Generates code through an intermediate interface. Humans: Users inspect, steer and structure generation before acceptance.	Work: Interpret and control generated code. Evidence: The temporary interface makes choices visible before code enters the notebook.

Choose a mechanism by the work it needs to support

Use the comparisons to connect an instructional need to a design choice and relevant evidence. [1]

Instructional need	Mechanism to consider	Evidence to inspect
Feedback needs instructional criteria.	Use rubric anchoring, local style standards or a documented content workflow (GLOW, CodeStylist, GAIDE).	Alignment with learning goals or course standards, clarity and pedagogical relevance.
Outputs become course guidance.	Place staff approval or representational review before publication or use (BoilerTAI, BoilerSketch).	Final response or diagram quality, staff edits and review workload.
Help could replace student reasoning.	Use reflection, guarded help or scaffolded task transformation (Owlgorithm, CodeHelp, CodeAid, CodeTailor).	Reflection and debugging, solution leakage, and whether the scaffold preserves active construction.
Generated code needs user interpretation.	Stage inspection and control before code acceptance (BISCUIT).	Whether users can inspect, steer and interpret generation before incorporation.
Students need judgment about AI use.	Build structured prompting, critique and reflection activities (AI-Lab).	Accounts of appropriate use, reflection on skill development, perceptions and usage patterns.

Combine mechanisms when the task calls for it. The paper proposes CodeAid-style guardrails with CodeTailor-style tasks, BoilerTAI-style approval with GAIDE-style workflow documentation, or BISCUIT-style interfaces with CodeStylist-style course constraints. These are proposed combinations. [1, §4]

Evidence scope. The framework comes from abductive design analysis of seven cases and analytical application to four external systems. The paper does not test the proposed combinations or whether framework-guided design improves outcomes.

Paper, original studies and more

[1] Ethan Dickey (2026). Instructional Governance by Design: A Framework for AI in Computing Education. arxiv.org/abs/2609.26098 | www.cs.purdue.edu/homes/dickeye/
[2] CodeHelp (Liffiton et al.). [doi:10.1145/3631802.3631830](https://doi.org/10.1145/3631802.3631830) [3] CodeAid (Kazemitabaar et al.). [doi:10.1145/3613904.3642773](https://doi.org/10.1145/3613904.3642773)
[4] CodeTailor (Hou et al.). [doi:10.1145/3657604.3662032](https://doi.org/10.1145/3657604.3662032) [5] BISCUIT (Cheng et al.). [doi:10.1109/VL/HCC60511.2024.00012](https://doi.org/10.1109/VL/HCC60511.2024.00012)

