VL/HCC 2026 · Scoping Review

Exploring the Design Space of LLM-Based Programming Support in CS Education

A scoping review through the lens of assistance governance

Minsun Kim
minsunkim@vt.edu

S. Moonwara A. Monisha
Virginia Tech

Zihan Wu
University of Maine

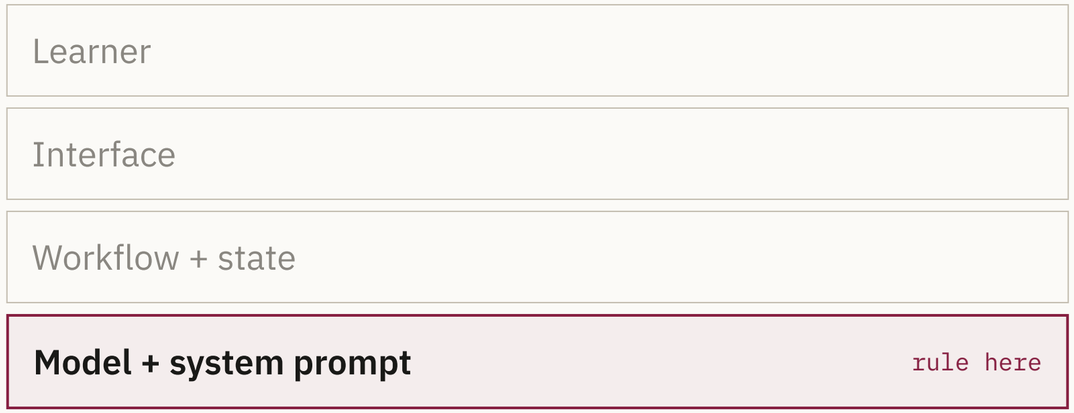
David H. Smith IV
Virginia Tech

Virginia Tech
ASCEND^3 Lab

The problem in one picture

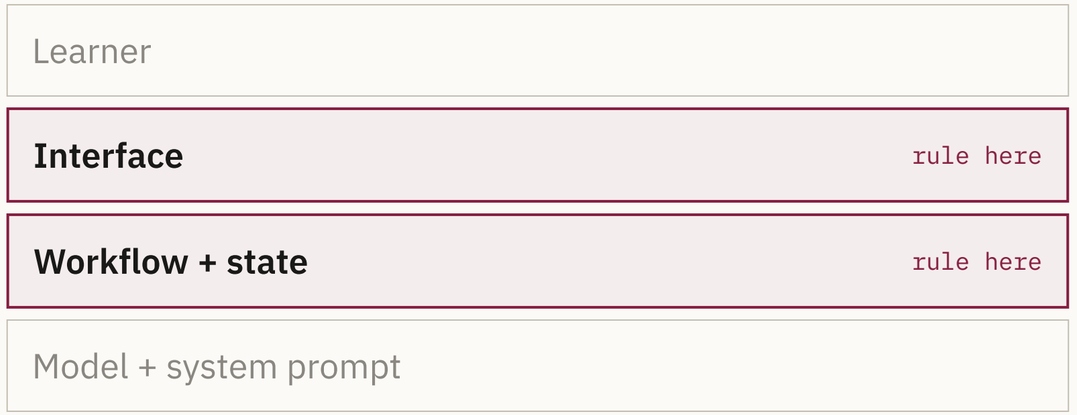
Same goal. Different place for the rule.

System A — prompt-governed



- **Revisable in an afternoon.**
- Also talk-around-able: the rule only holds while the model complies.

System B — workflow-governed



- **Hard to bypass.**
- Also hard to relax when a learner is genuinely stuck.

Where the literature stands

Prior reviews mapped where and why. Less on how assistance is bounded.

What we already have

- Broad reviews of GenAI in computing education map the tools and the tasks they support.
- They map teaching practices and educator responses.
- They catalogue risks — over-reliance, bias, misinformation, academic misconduct.

What is missing

- Their categories do not distinguish whether a system restricts solution-level help.
- Nor **how** that restriction is enforced.
- Nor whether learners and instructors can **modify it during use**.

The lens

Three questions: What · How · Who

Asistance governance:

how a learning support system defines, implements, and allocates control over learner-facing assistance — decisions every system already makes, just rarely stated as decisions.

What — is allowed

Policy

What help is permitted, delayed, or restricted.

Scaffolding · help-seeking · over-reliance · integrity

How — it is enacted

Enforcement

How those boundaries are enacted, in interaction and system behaviour.

Access control · human–AI interaction

Who — can change it

Authority

Who can configure or override them during use.

Control allocation · learner agency · orchestration

Research questions

RQ1 **What categories characterize assistance governance?**

What policy, enforcement, and authority categories appear across LLM-based programming support systems?

RQ2 **How are they distributed and combined?**

How do these categories co-occur across systems, and what underexplored configurations emerge from the resulting design space?

90

peer-reviewed, learner-facing systems

3

databases: ACM DL, IEEE Xplore, Scopus

2023–26

January 2023 through March 2026

01 · Method — search and screening

From database records to 90 systems

A scoping review to build the corpus, then a qualitative synthesis to code governance inside it.

- Structured search across **ACM DL, IEEE Xplore, and Scopus**, run 10 March 2026, covering January 2023 onward.
- Four concept groups combined: LLM terms × CS/programming education terms × learner-facing system terms × design or development terms.
- Title and abstract screening by one reviewer; a second reviewer independently screened a random subset of **1,010 papers**, with disagreements resolved by consensus.
- Full-text screening by three reviewers after a 20-paper calibration exercise.

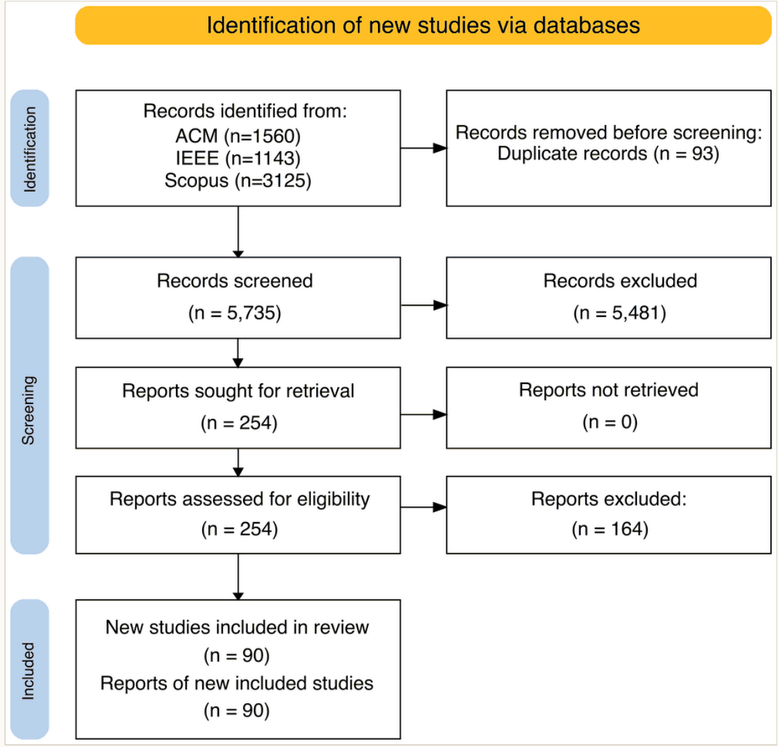


Fig. 1. PRISMA-ScR flow of study selection. 5,735 records screened after de-duplication, 254 full texts assessed, 90 studies included.

Scope

What counted as a governable system

Included

- A learner-facing tool, system, or prototype for CS or programming education.
- At least one **meaningful adaptation** beyond a general-purpose LLM interface or unmodified API.
- Enough reported detail to support governance extraction.

Excluded

- Curriculum and instructional planning, or learning analytics.
- Literature reviews, benchmarks, and surveys.
- Perception- or interview-only studies.
- Direct ChatGPT or unmodified API use.
- Papers under five pages.
- Papers without a user study.

Codebook development & reliability

An LLM in the loop — but not in the coder's seat

- **Provisional candidates only.** After a 20-paper pilot, an LLM processed the full corpus to generate candidate policy–enforcement pairs and evidence excerpts.
- **Every label human-reviewed.** No LLM-generated label was accepted without review; the model was never treated as a coder or a rater.
- **Team consolidation.** The team merged overlapping categories and refined definitions into a finalized codebook — dimensions deductive, subcategories inductive.
- Two human reviewers then independently coded the same **20 papers** with the finalized codebook; unweighted Cohen's κ per label, macro-averaged. No LLM–human reliability is reported, since the LLM was never positioned as an independent coder.

.705

macro-averaged Cohen's κ — Policy

.679

macro-averaged Cohen's κ — Enforcement

Coding Authority

A deliberately strict bar for runtime control

Counted only where a paper reported an **explicit runtime feature**.

Counted as authority

Hint-level selection · solution access toggle · feedback type · scaffolding settings · override

Did not count

Design-time configurability · researcher-side parameters · stated intent with no shipped feature

02 · Corpus — learners and learning contexts

Mostly novices, mostly in class — but not only

69%
targeted introductory programming learners (62 studies)

28%
addressed non-introductory contexts (25 studies)

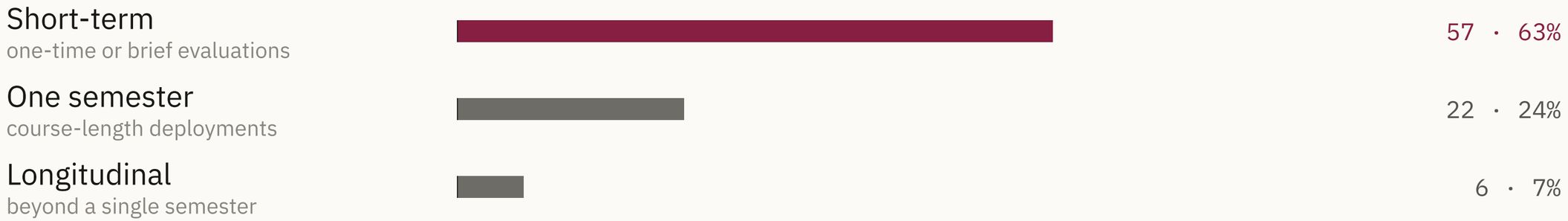
48%
deployed in in-class research settings (43 studies)

18%
home or field contexts (16 studies)

External researcher-managed settings accounted for a further 22% (20 studies), and four studies spanned two environments. LLM-based programming support is being explored across **structurally different educational arrangements** — not within one canonical setting.

Study design and methodology

Short, quantitative, and thin on learner experience



Over half carry no qualitative component

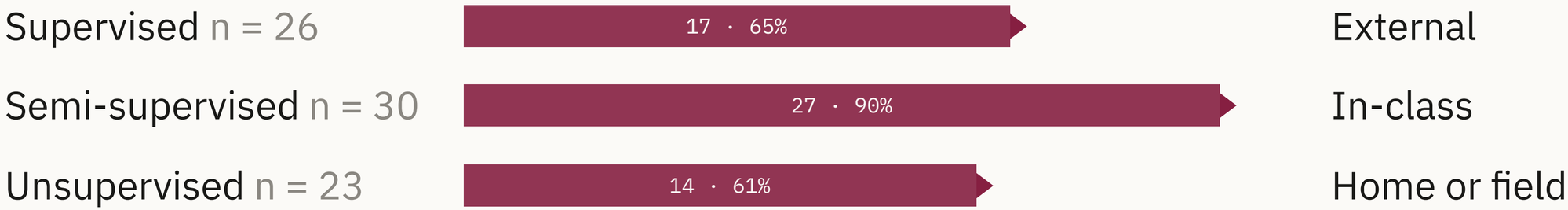
49 studies (54%) included no qualitative method, limiting insight into learner experience and process. Across all 90, 41 studies (45.6%) used at least one.

But practice is broadening

Qualitative methods appeared in 23 of 41 studies from 2025 (56%), up from 13 of 34 in 2024 (38%). True experimental designs rose from 15% to 27% over the same span.

Supervision × study environment

Supervision is inherited from the setting



Where direct oversight is limited, **system enforcement and learner authority** carry more of the load.

03 · Findings — RQ1–2 · Policy, 12 categories in 6 themes

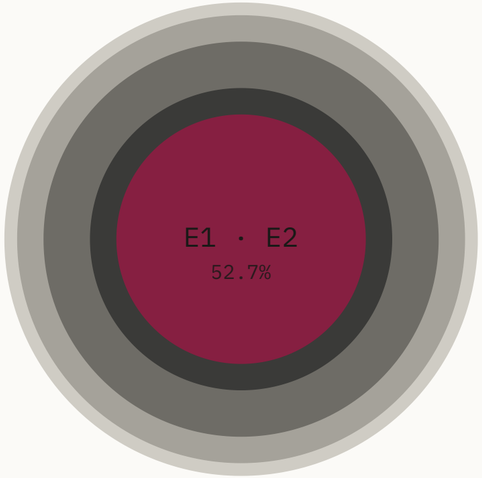
Policy is a bundle, not a switch

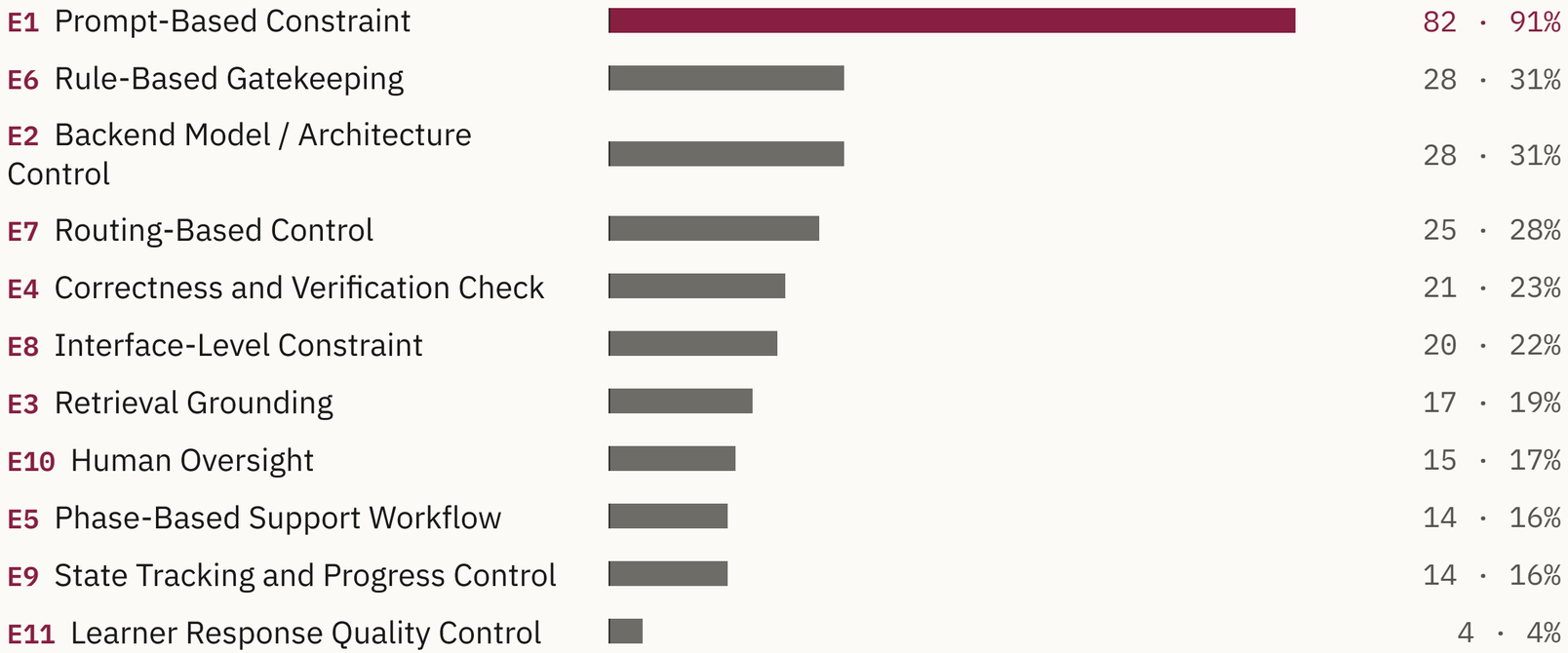
- **Response Constraint** — **P1** Scope Restriction, · **P2** Direct-Solution Prohibition, · **P3** Scaffolding-Oriented Support
- **Cognitive Scaffolding** — **P4** Independent Problem Solving, · **P5** Pedagogical Alignment, · **P6** Demonstrating Understanding
- **Participation Support** — **P7** Collaborative Learning Support, · **P8** Reward-Based Engagement
- **Learner Adaptation** — **P9** Learning-Based Personalization, · **P10** Engagement-Based Personalization
- **AI Literacy** — **P11** Responsible AI Use Instruction
- **Content Generation** — **P12** Learning Material Generation

Collaboration, motivation, AI literacy, and material generation trail far behind — a gap that returns later in the design space.

03 · Findings — RQ1–2 · Enforcement, 11 categories in 5 themes

A prompt monoculture, close to the model

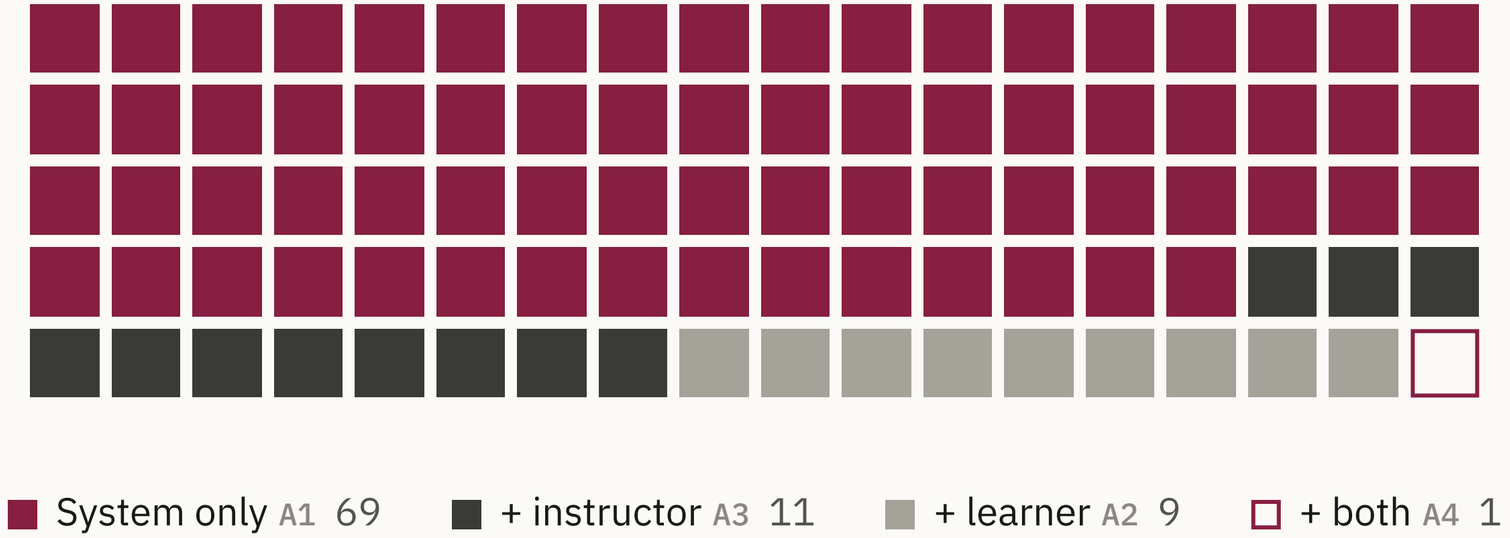




Rings read outward from the model — Prompt & Model Control (E1–E2) · Grounding (E3–E4) · Workflow & Access Control (E5–E7) · Interface & Interaction (E8–E9) · Pedagogical Oversight (E10–E11), by share of 423 pairings. Systems averaged 2.98 mechanisms, but 91% use E1 alone — enforcement is strongly centered on the instruction layer.

RQ2 · Authority — one cell per reviewed system

Control stays in the system



Policy and enforcement spread widely.

Authority does not — **one** system in ninety let a learner and an instructor both move the boundary.

RQ2 · Policy × Enforcement

Prompts carry almost every policy goal

132 possible pairings. **82 observed**, 50 never. The top six all run through **E1**:

- **P3** Scaffolding-Oriented Support — **38**
- **P2** Direct-Solution Prohibition — **31**
- **P9** Learning-Based Personalization — **27**
- **P5** Pedagogical Alignment — **20**
- **P6** Demonstrating Understanding — **18**
- **P1** Scope Restriction — **18**

43.5%

of all 423 pair instances are **E1** alone (184 of them)

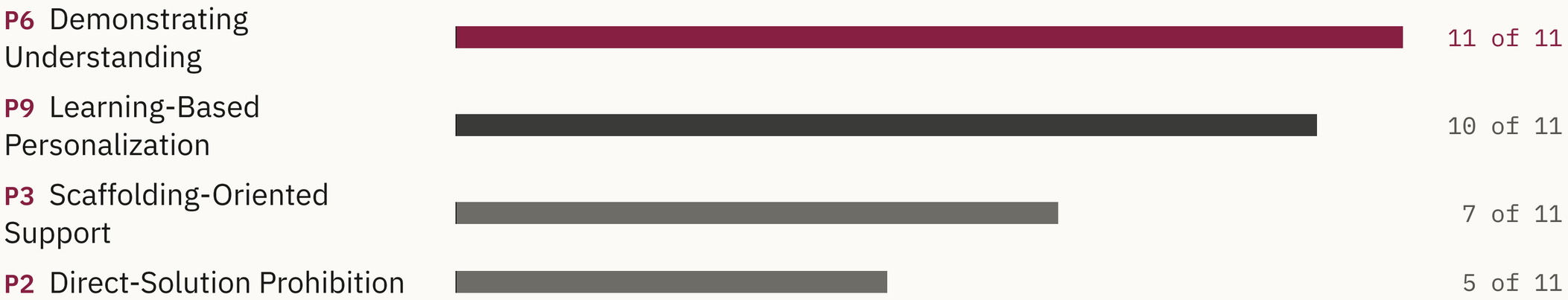
ρ = .114

policy count vs. enforcement count, n.s. (p = .283). More policies does not mean more mechanisms.

RQ2 · Enforcement breadth

Some goals need more than an instruction

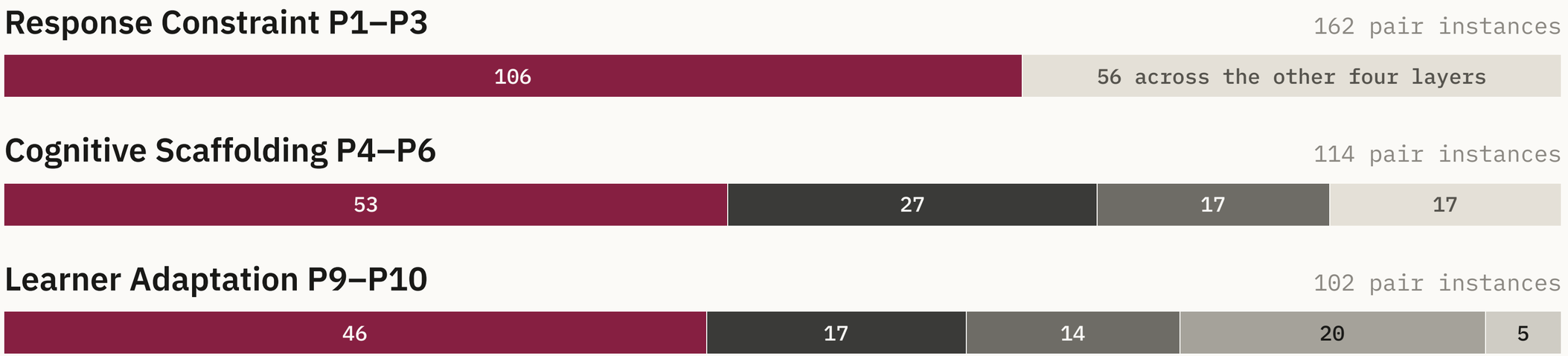
Policy categories differ sharply in how many distinct mechanisms have been used to realize them.



Prohibiting solutions is a **response-level** constraint — a prompt or a gate will do. Asking a learner to demonstrate understanding, or adapting to their progress, reaches into **workflow, state, interface, and people**.

RQ2 · Theme-level comparison

Different policy goals reach for different layers



Across all 423 pair instances: Prompt & Model 52.7% · Workflow & Access 19.6% · Knowledge & Grounding 11.1% · Interface 11.1% · Pedagogical Oversight 5.4%. Policy and enforcement themes are not independent, $\chi^2(20) = 60.77$, $p < .001$, Cramer's $V = .190$ — read descriptively, since pair instances from one paper are not fully independent.

RQ2 · Concentration across dimensions

Diverse in what and how. Centralized in who.

.114

Policy — lowest concentration; boundaries are expressed in many different ways

.149

Enforcement — dispersed across categories, yet still prompt-centered in practice

.613

Authority — substantially concentrated in system-only control

Herfindahl–Hirschman Index, the sum of squared category shares. The connection between the three dimensions is **uneven**: a broad set of policy goals, a narrower and prompt-centered set of enforcement practices, and a highly system-centered distribution of control.

04 · Discussion — two design tensions

Flexibility vs. stability, directness vs. effort

Flexibility vs. stability

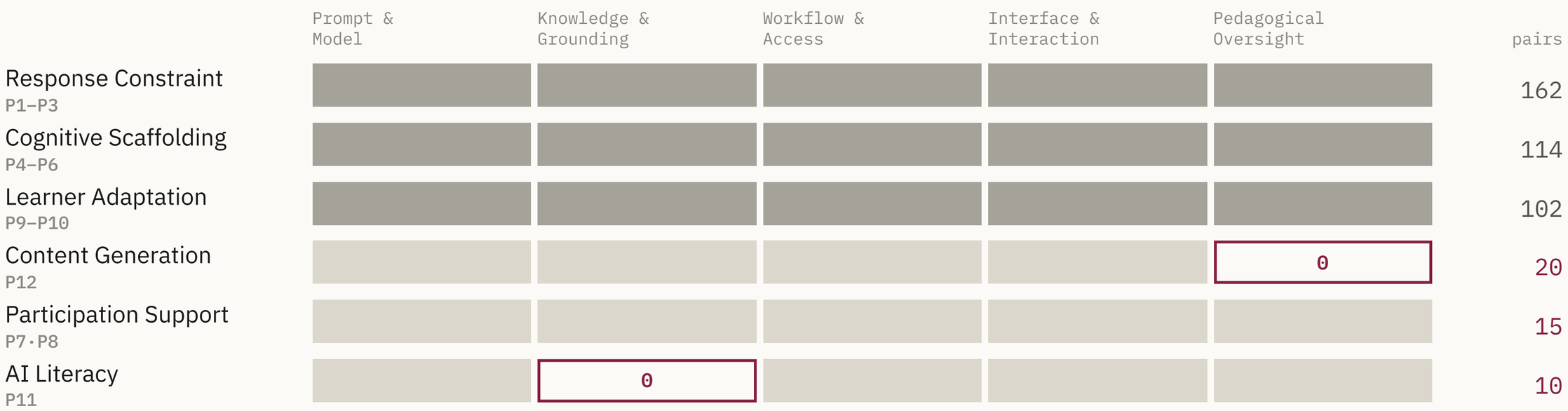
E1 Prompt-based constraint is cheap to author and revise — no touching interface or architecture.
But: unstable and invisible, holding only while the model complies. Structural mechanisms (gates, phases, verification, locks, oversight) make the boundary explicit and enforceable — **but** narrow open-ended help seeking.
91% of the corpus sits in the prompt column; that concentration is a design fact, not a verdict.

Directness vs. learner effort

P2 Restricting the response is usually enforced by an · **E1** prompt or · **E6** gate — directness as a property of one output. Preserves productive struggle, but slower when a learner is genuinely blocked.
P9 Personalization instead needs routing, state tracking, grounding, verification — a model of the learner over time, not a restriction on one output.
The question moves from **whether** to restrict, to **how** and **who** can adjust it.

Policy theme × enforcement theme — what the corpus never built

Where the map is blank



PEA as a design vocabulary

A walkthrough: the tutor that won’t hand over the answer

1 · Define the Policy

P2 Direct-Solution Prohibition

P3 Scaffolding-Oriented Support

Questions and hints before usable code.

2 · Select the Enforcement

E1 Prompt constraint restricts complete solutions

E5 Phase-based workflow raises hint specificity over time

E6 Rule-based gate delays examples until an attempt is made

3 · Allocate the Authority

A1 Fixed by the system

A3 Configured by the instructor

A2 Learner chooses, within instructor-set limits

From analysis to design

Three configurations to build — and three limits on this map

01 **Collaborative help routing**

Simple questions → AI. Conceptual → peers. Repeated difficulty → instructor. Fills the Participation Support gap.

02 **Grounded AI literacy**

Compare a generated explanation against course material, test the code, justify the fix. Fills the AI Literacy × Grounding gap.

03 **Governed content generation**

Instructor sets objectives, difficulty, prerequisites, and a review gate before material reaches learners. Fills the Content Generation × Oversight gap.

01 **Bounded by the published record**

Three databases, one time window. Preprints, workshops, industry tools, and local deployments are not represented.

02 **Inclusion criteria shape the space**

Sparse cells are **underexplored here** — not infeasible.

03 **Screening audit trail**

Full-text exclusion reasons were logged only in aggregate, not as exclusive categories.

Conclusion

Not what these systems can generate — but how their assistance is governed

Shared goals, divergent mechanisms

Similar intentions, different mechanisms. Governance is a **configuration**, not a switch.

Enforcement is prompt-centered

E1 in 91% of systems, 43.5% of all pairings — flexibility bought with stability and visibility.

Authority is the frontier

77% fix the boundary in system logic. Shared runtime control is largely unbuilt.

PEA makes the choices sayable: **what help is available, how the boundary is implemented, who can move it**.

Thank you

Questions?

Presenter

Minsun Kim minsunkim@vt.edu

Lab

ascend3.cs.vt.edu

A

Appendix

Two systems from the corpus, read back through PEA — and the corpus overview figure.

Appendix A · Worked example 1 — corpus paper [21]

CodeHelp: guardrails, verification, and an instructor in the loop

Liffiton, Sheese, Savelka & Denny — **Using Large Language Models with Guardrails for Scalable Support in Programming Classes**, Koli Calling 2023.

Policy what help is available	**P2** Direct-Solution Prohibition — no complete, directly usable code **P1** Scope Restriction — help bounded to the course and its tasks **P6** Demonstrating Understanding — the learner states the problem before help arrives **P11** Responsible AI Use Instruction
Enforcement where the boundary lives	**E1** Prompt-Based Constraint — carries both P2 and P1 **E6** Rule-Based Gatekeeping **E4** Correctness and Verification Check **E10** Human Oversight — the rarest theme in the corpus, at 5.4% of pairings
Authority who can adjust it	**A3** System + instructor — one of the 11 studies (12.2%) giving instructors runtime control

Coded pairs, verbatim: P2–E1, P1–E1, P6–E6, P11–E10, P2–E4 · Authority A3. Supplemental Tables I and II.

Appendix A · Worked example 2 — corpus paper [19]

CodeAid: the same prohibition, spent on personalisation

Kazemitabaar, Ye, Wang, Henley, Denny, Craig & Grossman — **Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs**, CHI 2024.

Policy what help is available	**P2** Direct-Solution Prohibition — the same opening move as CodeHelp **P1** Scope Restriction **P9** Learning-Based Personalization — coded three times, against three different mechanisms **P10** Engagement-Based Personalization
Enforcement where the boundary lives	**E1** Prompt-Based Constraint — carries P2 and P1, as almost everywhere **E7** Routing-Based Control — requests go to different treatments by question type **E8** Interface-Level Constraint — coded twice **E3** Retrieval Grounding
Authority who can adjust it	**A2** System + learner — one of the 9 studies (10.0%) giving learners runtime control

Coded pairs, verbatim: P2–E1, P9–E7, P10–E8, P9–E8, P1–E1, P9–E3 · Authority A2. Supplemental Tables I and II.

Appendix B · Reference figure

Corpus overview

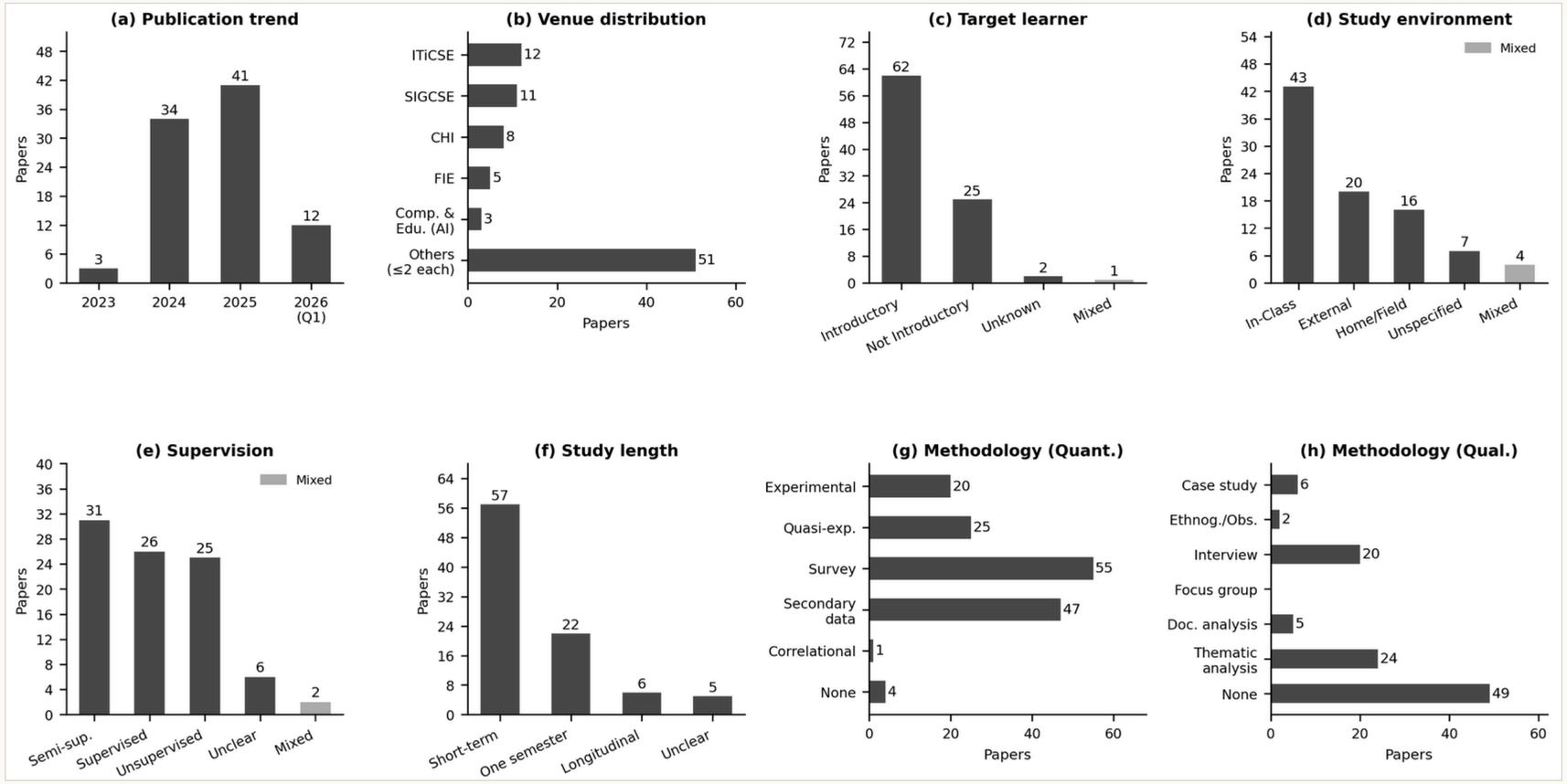


Fig. 2. Descriptive characteristics of the 90 reviewed studies: (a) publication trend, (b) venue distribution, (c) target learner, (d) study environment, (e) supervision, (f) study length, (g) quantitative methodology, (h) qualitative methodology.