ThinkDeeper Web Coach: A Literature-Informed Design Specification for Process-Visible Reasoning in Problem-Solving-Intensive Information Technology Courses

Rafli Arrasyid[1*], Nuur Wachid Abdul Majid[2]

[1,2]Department of Information System and Technology Education, Universitas Pendidikan Indonesia, Bandung, West Java, Indonesia
E-mail: [1]rafly111104@upi.edu, [2]nuurwachid@upi.edu

(Received: 30 May 2026, revised: 29 Jun 2026, accepted: 30 Jun 2026)

Abstract

Information technology education increasingly requires students to demonstrate how they reason while generative tools can produce plausible code, queries, and explanations with little visible evidence of the learner's own computational thinking. This study aimed to design ThinkDeeper Web Coach, a process-visible learning artifact for problem-solving-intensive IT courses that requires structured reasoning before direct solution support is accessed. A literature-informed design and development approach was applied to academic records indexed in Semantic Scholar and OpenAlex. The evidence corpus comprised 500 records published from 2023 to 2025, 130 records retained after abstract screening, 53 full texts assessed, and 51 studies included after full-text screening and extraction. The synthesis produced the Process-Visible Learning Design Framework, eight evidence-linked design requirements, and six integrated modules: Answer Delay Gate, Coach Chat, Evidence Drawer, Depth Analytics, Learning Circle, and Resource Recommendation. The revised specification defines applicable course contexts, operational dimensions of depth of thinking, a hybrid reference architecture, server-side gate and guardrail mechanics, trace data objects, and a staged evaluation protocol. ThinkDeeper is presented as a theoretically and empirically justified design specification that still requires expert review, usability testing, trace validation, and classroom pilot evaluation before effectiveness claims can be made.

Keywords: Depth of Thinking, Process-visible Learning, Information Technology Education, Learning Analytics, Scaffolding

I. INTRODUCTION

Information technology (IT) education requires learners to reason through ill-structured and semi-structured tasks rather than merely submit a correct final product. In programming and web development, students must decompose requirements, design algorithms, predict outputs, construct test cases, debug failures, and justify revisions. Comparable process evidence is required when students formulate SQL queries, interpret data-analysis results, or model system requirements. Yet instructors increasingly receive syntactically correct code, polished explanations, or complete design patterns that reveal little about whether students understood the problem, evaluated alternatives, or tested boundary conditions. Studies in IT-related education identify computational thinking, problem solving, self-regulated learning, cognitive engagement, and reflective thinking as central outcomes [1]-[5].

The educational phenomenon has become more consequential as generative systems make direct answers available outside the normal instructional sequence. Overreliance can lead to superficial code submission, copied solution structures, and weak explanation of assumptions or test logic. A final artifact may therefore look acceptable while the underlying reasoning remains unexamined. This creates an assessment problem: lecturers must distinguish productive assistance from substitution of student thinking, but conventional learning-management logs primarily record access, completion, and scores. Recent evidence suggests that intelligent support is more defensible when it preserves learner responsibility through decomposition, questioning, reflection, and explicit constraints on assistance [6]-[11].

Learning analytics offers one pathway for diagnosing this problem because discourse-aware analysis, dashboards, and trace-based representations can expose participation, cognitive presence, problem-solving provenance, revision behavior, and hint dependence [12]-[15]. However, an indicator is pedagogically meaningful only when a lecturer can inspect the reasoning evidence from which it was derived

DOI: 10.34148/teknika.v15i2.1504

and decide whether intervention is warranted [16]-[18]. Without that connection, analytics can reduce complex reasoning to an opaque score and may increase rather than reduce diagnostic uncertainty.

Collaborative orchestration and adaptive recommendation can extend this process when they are aligned with a reasoning task. Structured peer critique, socially shared regulation, and Think-Pair-Share activities can prompt elaboration and comparison [19]-[22], while recommendation systems can map a diagnosed gap to a relevant exercise or resource [23]-[26]. ThinkDeeper is therefore intended primarily for problem-solving-intensive IT courses and tasks, including algorithm design, programming, debugging, web development, SQL and database-query design, data-analysis workflows, and systems-analysis activities that produce inspectable assumptions, plans, tests, and explanations. It is less suitable for simple recall tasks or assessments in which no meaningful reasoning trace can be specified.

Operationally, depth of thinking refers to students' ability to make reasoning visible through observable evidence rather than through final answers alone. The construct is treated as a process-oriented capability comprising problem framing, decomposition, testing logic, explanatory justification, reflective revision, and self-regulation. These dimensions guide task configuration, trace capture, analytics, and later validation, as summarized in Table 1.

Table 1. Operational Dimensions of Depth of Thinking

Dimension	Observable Evidence
Problem framing	Assumptions, constraints, and task interpretation.
Decomposition	Plan, pseudocode, and subproblem breakdown.
Testing logic	Test cases, edge cases, and boundary conditions.
Explanatory justification	Expected output and reasoning justification.
Reflective revision	Revision notes and response to feedback.
Self-regulation	Hint dependence and resource use.

Three related gaps motivate the artifact. The pedagogical gap is that intelligent learning systems may provide answers or hints without requiring visible reasoning evidence before support is accessed. The design gap is that learning analytics, scaffolding, collaboration, and recommendation are frequently implemented as separate functions rather than as one sequence of evidence elicitation, guided revision, peer elaboration, remediation, and instructor intervention. The evaluation gap is that the proposed integration has not yet been validated through expert review, usability testing, trace validation, or a classroom pilot. These distinctions prevent the design contribution from being confused with an effectiveness claim.

To integrate the mechanisms, this article introduces the Process-Visible Learning Design Framework (PVLDF). The framework links evidence elicitation, scaffolded inquiry, traceable revision, interpretive analytics, gap-responsive support, and human-in-the-loop governance. The research question was: how can empirical evidence on these mechanisms be translated into a web-based learning coach that makes reasoning visible in problem-solving-intensive IT courses while constraining premature access to complete solutions?

The article contributes four outputs. First, it defines the applicable course and task boundary and operationalizes depth of thinking as observable reasoning evidence. Second, it reports a transparent evidence-selection and coding procedure and translates recurring mechanisms into design requirements. Third, it specifies ThinkDeeper through the PVLDF, modules, workflows, guardrails, architecture, and analytic traces. Fourth, it proposes a measurable validation pathway. The contribution is therefore a design specification and evaluation protocol rather than a completed classroom effectiveness claim.

II. RESEARCH METHODOLOGY

A. Research Design

This study used a literature-informed design and development research approach because the primary output is an educational technology artifact and transferable design knowledge rather than a completed classroom effectiveness test. Educational design research supports iterative movement from problem analysis and evidence synthesis to requirement formulation, prototype specification, and validation planning [27]. In this study, the PVLDF served as the organizing design framework that connected pedagogical mechanisms to observable system functions and trace indicators.

The study was conducted at the design-specification and high-fidelity prototype stage. It included problem analysis, evidence synthesis, requirement derivation, module and workflow specification, a proposed hybrid reference architecture, and validation planning. Consequently, the article does not claim that ThinkDeeper has improved student outcomes. It claims that the artifact is sufficiently specified and evidence-informed to proceed to staged validation. The overall procedure is shown in Figure 1.

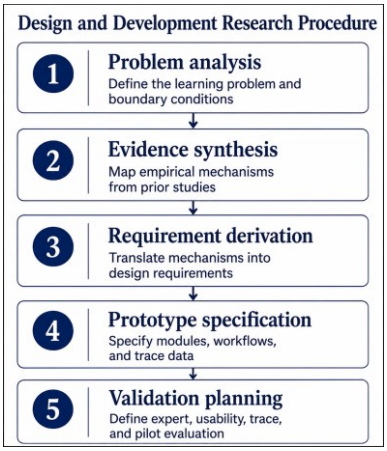

Figure 1. Literature-informed Design and Development Research Procedure Used in the Study.

DOI: 10.34148/teknika.v15i2.1504

B. Data Sources and Screening

The empirical foundation was constructed from academic records indexed in Semantic Scholar and OpenAlex. The search export used for analysis was generated on 24 December 2025 and covered publications from 2023 through 2025. The exact natural-language semantic query was: "ThinkDeeper Web Coach: Discourse-Aware Learning Analytics + Learning Circle Orchestration + Resource Recommendation to Increase Students' Depth of Thinking in IT Courses." No explicit language filter was applied. Journal articles, conference papers, and preprints were eligible when they provided empirical or design-oriented educational evidence. Duplicate candidates were flagged through normalized title, DOI, author, and venue metadata and were not treated as independent evidence during synthesis.

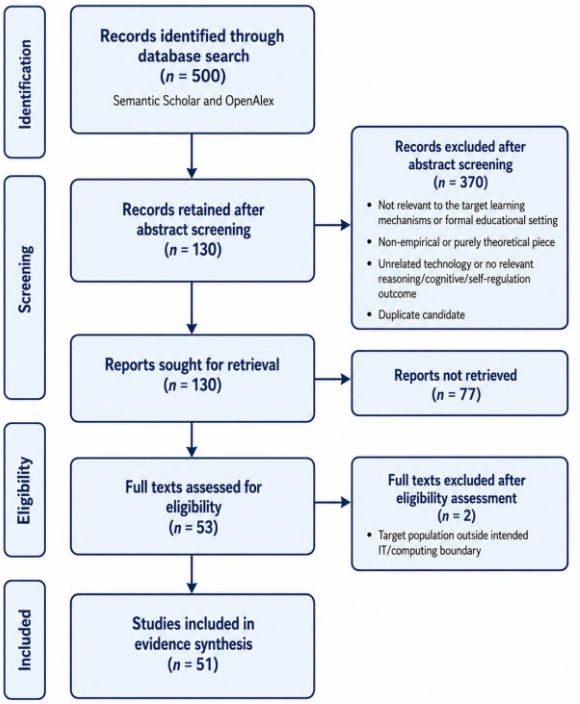

Figure 2. Evidence Identification and Screening Process Used to Derive the Artifact Requirements.

The initial corpus contained 500 records. Abstract screening excluded 370 records and retained 130 reports for retrieval. Full text could not be obtained for 77 reports; 53 full texts were assessed against the six criteria, two were excluded because the target population fell outside the intended IT/computing boundary, and 51 studies entered extraction and requirement synthesis. Screening considered target population, technology sophistication, cognitive or self-regulatory outcomes, empirical or design evidence, formal educational setting, and relevance to the artifact mechanism. Figure 2 and Table 2 report the reproducible flow and decision rules. Because the corpus is heterogeneous and includes adjacent disciplinary contexts, it is used to identify mechanisms and boundary conditions rather than to estimate pooled causal effects.

Criterion-level yes/maybe/no judgments and their rationales were recorded in structured worksheets. Uncertain abstract-stage records were retained when the holistic judgment favored inclusion and were resolved during full-text assessment. A structured model-assisted first pass was followed by consistency review; the process was not a blinded multi-reviewer screening, and no inter-rater agreement coefficient was calculated. Reports that could not be retrieved were recorded separately and were not counted as full-text eligibility exclusions.

Table 2. Literature Search, Screening, and Extraction Protocol

Component	Description
Data sources	Academic records indexed in Semantic Scholar and OpenAlex.
Search protocol	Export generated 24 December 2025. Exact semantic query reported verbatim in Section II-B.
Scope and restrictions	Publication years 2023-2025; no explicit language filter; journal articles, conference papers, and preprints.
Screening stages	500 identified; 370 excluded at abstract screening; 130 sought; 77 not retrieved and recorded separately; 53 assessed; 2 eligibility exclusions; 51 included.
Decision procedure	Criterion-level yes/maybe/no judgments with rationales; uncertain abstract records retained when the holistic judgment favored inclusion; no formal inter-rater coefficient.
Inclusion criteria	Relevant IT, computing, or technical-learning mechanism; intelligent/adaptive technology; cognitive, reasoning, engagement, or self-regulated learning outcome; empirical or design evidence.
Exclusion criteria	Non-educational context; purely theoretical piece; unrelated technology; no relevant outcome; target population outside the intended boundary.
Extraction and coding	Eight primary fields plus supporting quotations/table references; structured model-assisted first pass; consistency, stored-evidence, inclusion, and mechanism checks.
Final corpus	51 studies used for mechanism, boundary-condition, and requirement synthesis; not for meta-analysis or causal effectiveness claims. Complete coding matrix prepared as Supplementary Table S1.

C. Instruments and Data Items

The coding sheet contained eight primary fields: technology system, thinking-depth measures, effects on thinking, learning-analytics features, interaction mechanisms, course context, student characteristics, and study design. Supporting quotation and table-reference fields were retained

DOI: 10.34148/teknika.v15i2.1504

for each primary field so that interpretations could be traced back to source evidence. A structured language-model-assisted pass populated the initial extraction sheet; the resulting entries were then checked for internal consistency, duplicated concepts, alignment between finding and mechanism, and compatibility with the inclusion criteria. No pooled statistic or artifact requirement was accepted solely from an unsupported generated summary. The evidence chains selected for Table 4 were cross-checked against the stored supporting quotation and table-reference fields, and the numerical distributions in Table 3 were reconciled with the screening and extraction worksheets.

D. Data Analysis and Requirement Derivation

Analysis proceeded in four steps. First, records were coded into course-context, technology-type, cognitive-construct, interaction-mechanism, analytic-feature, and study-design categories. Second, recurring mechanisms were organized into five evidence streams: discourse-aware analytics, question-driven scaffolding, collaborative orchestration, personalized recommendation, and instructor-mediated intervention. Third, each stream was translated into a design principle within the PVLDF. Fourth, each principle was operationalized as one or more requirements, trace objects, and evaluation indicators.

A requirement was retained when it appeared across more than one evidence stream or when a highly relevant study described a sufficiently explicit mechanism that could be represented as an observable artifact function. Positive results were not treated as universally transferable. Ambiguous, null, or context-dependent findings were used to define boundary conditions, such as the need for calibrated assistance, course-specific gate rules, lecturer review, and cautious interpretation of analytics. Table 3 summarizes the distribution of the included studies by course context and technology type, while Table 4 presents representative study-level evidence-to-requirement chains, the complete 51-study coding matrix, including course context, technology type, study design, evidence stream, and supporting evidence fields, is prepared as Supplementary Table S1.

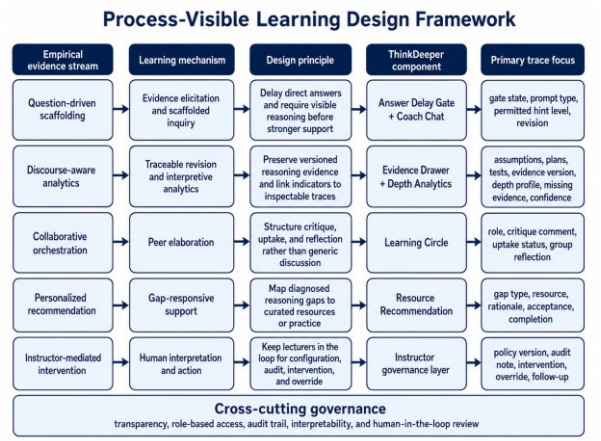

Figure 3. Process-Visible Learning Design Framework used to translate empirical mechanisms into artifact decisions.

III. RESULTS AND DISCUSSION

A. Evidence Base Characteristics and Evidence-to-Design Synthesis

The 51 included studies were heterogeneous rather than exclusively programming-focused. Fifteen studies concerned programming, software, or web-development contexts; eleven addressed general computing, IT, or informatics; five concerned data, analytics, or quantitative computing; and twenty studies came from adjacent STEM, engineering, general education, or mixed contexts. Technology categories were dominated by learning analytics, dashboards, and predictive systems (20 studies), followed by generative assistants or pedagogical agents (13), adaptive recommenders or intelligent tutoring systems (8), collaboration or digital-learning platforms (5), other instructional technologies (2), and studies without a specific system (3). Table 3 makes these boundary conditions explicit.

Across this diverse base, the most consistent mechanism was not a particular product but the preservation of learner activity. Discourse and trace studies supported inspectable process evidence [12]-[16]; question-driven systems supported decomposition, reflection, and learner-led inquiry [6]-[11]; collaborative studies supported structured critique rather than unstructured social interaction [19]-[22], [28]; and recommendation studies were strongest when resources responded to identified learning gaps [23]-[26]. These convergent mechanisms justify the PVLDF while the disciplinary heterogeneity cautions against claiming applicability to every IT course or every form of assessment.

Table 3. Distribution of the 51 Included Studies

Dimension	Distribution, n (%)
Course context	Programming/software/web 15 (29.4); computing/IT/informatics 11 (21.6); data/analytics/quantitative 5 (9.8); adjacent STEM/engineering 11 (21.6); general/non-IT 7 (13.7); mixed/unspecified 2 (3.9).
Technology type	Analytics/dashboard/predictive 20 (39.2); GenAI/chatbot/pedagogical agent 13 (25.5); adaptive/recommender/ITS 8 (15.7); collaboration/digital platform 5 (9.8); other instructional system 2 (3.9); no specific system 3 (5.9).

B. Process-Visible Learning Framework, Architecture, and Design Requirements

ThinkDeeper Web Coach is specified as a process-visible learning artifact for problem-solving-intensive IT coursework. The PVLDF in Figure 3 organizes five evidence-to-design pathways: eliciting evidence, scaffolding inquiry, preserving revision, interpreting traces, and responding to gaps. Human-in-the-loop governance cuts across the sequence. The artifact operationalizes the framework through six integrated modules: Answer Delay Gate, Coach Chat, Evidence Drawer, Depth Analytics, Learning Circle, and

DOI: 10.34148/teknika.v15i2.1504

Resource Recommendation. The instructor dashboard is treated as a governance and intervention layer rather than as a seventh student-learning module.

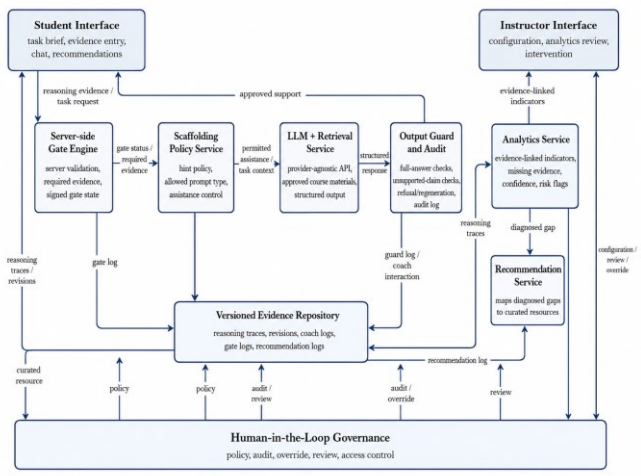

Figure 4. Proposed Hybrid Reference Architecture of ThinkDeeper Web Coach.

The design requirements preserve an auditable chain from selected evidence to evidence stream, learning mechanism, design principle, artifact behavior, and trace indicator. Table 4 makes this chain explicit. The proposed architecture in Figure 4 is hybrid: deterministic server-side rules control gate status and assistance permissions, while model-based services generate constrained questions, analyze selected traces, and recommend curated resources. This separation prevents a conversational model from unilaterally deciding whether a student has met the evidence requirement. Operationally, versioned evidence is read by the analytics service, diagnosed gaps are passed to the recommendation service, and curated resources are returned to the student interface. Gate state is passed to the policy service, which constrains the model, and only outputs accepted by the guard are delivered to the learner.

Table 4. Representative Study-to-Requirement Audit Trail

Study/context	Finding, evidence stream, design principle, and associated requirement/trace
Ma et al. [8]; algorithmic programming	Learner-led decomposition improved cognitive engagement and critical thinking. This finding supports the question-driven scaffolding stream and the principle of requiring learner-authored steps before deeper assistance (DR1-DR2). Trace indicators: gate state, hint level, and revision.
Park et al. [11]; graduate HCI	Informed inquiry generated significantly more reasoning and deeper interaction. This finding supports question-first scaffolding and the principle of asking rather than answering (DR2). Trace indicators: prompt type, response, and revision.
Dornauer et al. [13]; online discussion	The classification of discussion traces identified cognitive-presence phases. This finding supports discourse-aware analytics and the preservation of inspectable, interpretable evidence (DR3-DR4). Trace indicators: coded phase, evidence source, and confidence.
Utamachant et al. [17]; programming	Evidence-based analytics supported timely instructional intervention. This finding supports instructor-mediated analytics and the principle of linking indicators to audit and follow-up activities (DR4, DR7). Trace indicators: indicator, audit note, and follow-up.
Lobo-Quintero [20]; systems engineering	AI-enhanced Think-Pair-Share increased productivity and thematic diversity, with the strongest pattern at moderate AI similarity. This finding supports collaborative orchestration and calibrated peer elaboration (DR5, DR8). Trace indicators: role, contribution, AI similarity, and uptake.
Pham-Duc [25]; introductory programming	Personalized recommendation increased exercise completion by 73.8% and improved performance. This finding supports gap-responsive remediation and the principle of recommending curated practice for diagnosed gaps (DR6). Trace indicators: gap, resource, acceptance, and completion.
Guo et al. [16]; intelligent tutoring	Visual analytics exposed learner problem-solving traces for responsive pedagogy. This finding supports human interpretation and evidence-auditable oversight (DR4, DR7-DR8). Trace indicators: source event, lecturer decision, and review trail.
Uzun et al. [18]; postgraduate EdTech	Engagement with analytics varied according to self-regulated learning and did not independently improve performance. This boundary condition supports the principle that analytics should inform, rather than automate, judgment (DR4, DR7-DR8). Trace indicators: engagement, confidence, and lecturer review.

C. Student and Instructor Workflows

The student workflow begins with a lecturer-defined task brief and an evidence schema appropriate to the course (Figure 5). A debugging task may require a fault hypothesis and failing test; a SQL task may require data assumptions, a query plan, and expected rows; a systems-analysis task may require stakeholder assumptions, alternative models, and validation criteria. Before a complete solution or exemplar

DOI: 10.34148/teknika.v15i2.1504

can be revealed, the learner must submit the configured minimum evidence. Coach Chat then provides clarifying questions, counterexamples, or partial hints permitted by the current gate state and may not return a complete solution before the gate is satisfied.

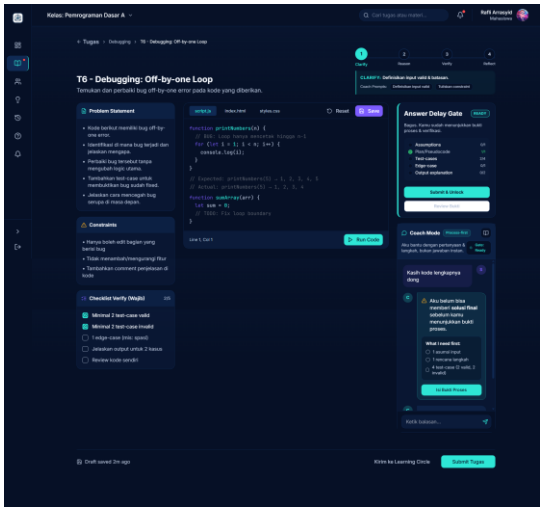

Figure 5. Student Workspace Mock-up Interface.

The instructor workflow begins with task configuration and continues with evidence-linked monitoring (Figure 6). Lecturers define the required evidence, permitted hint levels, curated resources, peer roles, and conditions for override. The dashboard identifies possible missing assumptions, weak test coverage, low explanation specificity, repeated hint dependence, or limited peer contribution, but it does not assign a final grade automatically. Every indicator must open the underlying student evidence and its revision history so that the lecturer can confirm, reject, or contextualize the system's interpretation.

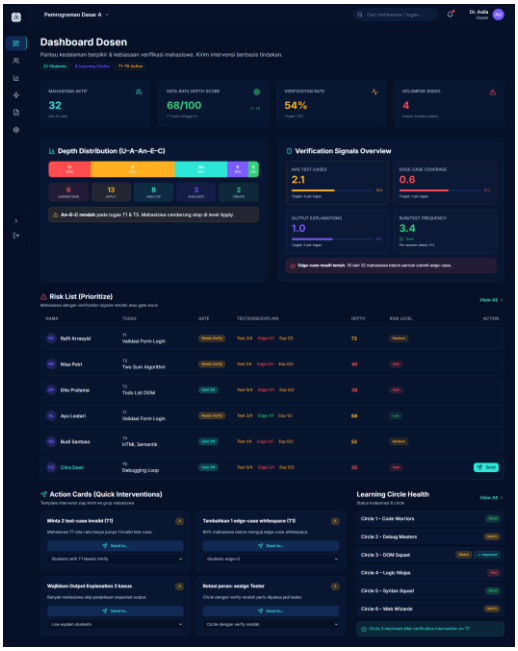

Figure 6. Instructor Dashboard Mock-up Interface.

Table 5 summarizes the six modules and the cross-cutting instructor-governance layer. The distinction is important: student-facing modules structure learning activity, whereas governance functions configure rules, inspect evidence, authorize overrides, and record intervention decisions.

Table 5. Core Modules, Governance Layer, and Trace Data

Module/layer	Trace focus
Answer Delay Gate	Gate state, policy version, required evidence, specificity check, bypass attempt, override.
Coach Chat	Allowed prompt type, hint level, retrieval source, response, guard result, revision link.
Evidence Drawer	Assumptions, plans, tests, explanations, timestamps, versions, peer comments.
Depth Analytics	Evidence-linked depth profile, confidence, missing-evidence flags, rule/model version.
Learning Circle	Role participation, critique, uptake, revision, and group reflection.
Resource Recommendation	Gap type, resource, source rationale, acceptance, and completion.
Instructor governance layer	Task configuration, audit sampling, intervention, override, follow-up, and review trail.

D. Technical Flow, Guardrails, Trace Data Model, and Use Case Specification

Figure 7 presents the proposed server-side Answer Delay Gate and guardrail workflow rather than a completed production implementation. A concrete future stack can use a React or Next.js client, a FastAPI service layer, PostgreSQL for versioned reasoning evidence and event logs, role-based authentication, and a provider-agnostic large-language-model API behind policy, retrieval, and output-guard services. Curated course materials form the retrieval source. Deterministic analytics calculate evidence completeness and trace features, while explainable classifiers may be evaluated later for higher-level indicators. These choices preserve portability and prevent the model provider from becoming the system's source of truth.

The Answer Delay Gate operates on the server rather than in the browser. It validates required fields, minimum specificity rules, version and timestamp information, and a signed gate state. If evidence is incomplete, a policy service selects the allowed assistance level and sends the model only the necessary task context and approved course resources. The model must return structured output such as a Socratic question, counterexample, reflection prompt, or partial hint. An output guard checks for complete code, final-answer leakage, unsupported claims, and policy violations. Failed checks trigger refusal or regeneration; bypass attempts, refusals, prompt versions, and lecturer overrides are logged. Retrieval citations, an abstention option, audit sampling, and instructor review reduce the consequences of hallucinated or inappropriate responses. In this specification, minimum

DOI: 10.34148/teknika.v15i2.1504

specificity denotes a lecturer-defined threshold for non-empty, task-relevant evidence; signed gate state is a server-verifiable record of the active rule and completion result; a bypass attempt is a request for disallowed assistance before gate completion; and guard result records whether an output was allowed, refused, or regenerated.

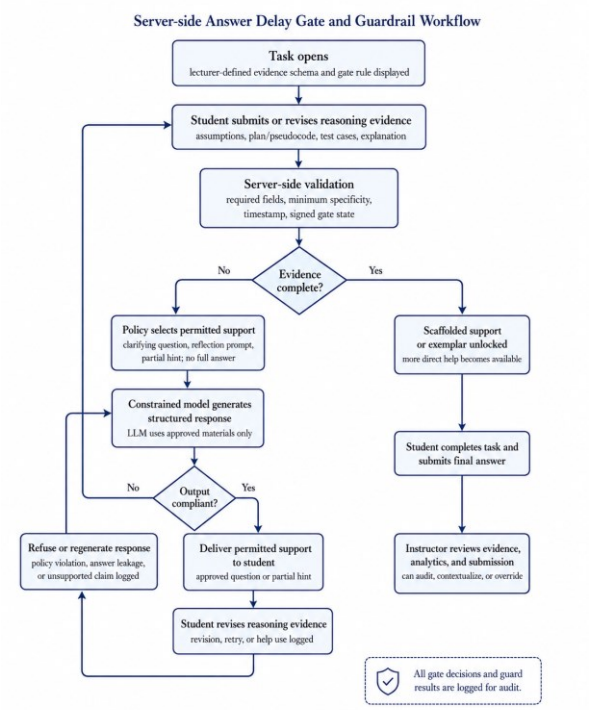

Figure 7. Server-side Answer Delay Gate and Guardrail Workflow.

The trace data model in Table 6 specifies the minimum objects required for process visibility, policy enforcement, and later validation. In addition to reasoning versions and learning interactions, the revised model stores policy version, gate signature, source reference, guard status, and override history. A future database schema and API must implement data minimization, retention periods, and role-based access because reasoning traces may reveal student misconceptions and assistance-seeking behavior.

Table 6. Trace Data Model for Process-Visible Learning

Entity	Key trace attributes
Task	task_id, course_context, expected_evidence, gate_rule, policy_version, due_date
Reasoning Evidence	assumption, plan, test_case, edge_case, explanation, specificity, version, timestamp
Coach Interaction	prompt_type, hint_level, retrieval_source, student_response, guard_status, revision_link
Peer Interaction	role, critique_comment, uptake_status, linked_revision, group_reflection
Recommendation	gap_type, resource_id, source_reference, rationale, acceptance, completion_status
Instructor Intervention	evidence_reference, audit_note, feedback_type, override_reason, follow_up_status, reviewer
System Audit	gate_signature, policy_version, prompt_hash, response_hash, bypass_attempt, refusal, timestamp

One Task may contain many Reasoning Evidence versions. One evidence version may be linked to multiple Coach Interactions, each governed by the gate state and hint policy active at that time. Peer Interaction records whether critique is incorporated into later work. Recommendation links a diagnosed gap to a curated resource and stores the selection rationale. Instructor Intervention must reference specific evidence or a risk indicator. System Audit records preserve prompt and policy versions, output-guard decisions, bypass attempts, and authorized overrides so that disputed recommendations can be reconstructed.

In use-case terms, the student accesses a task, submits reasoning evidence, requests permitted scaffolding, participates in a Learning Circle, uses recommended resources, and submits the final answer. The instructor configures the task and gate, reviews analytics and source evidence, samples coach interactions, sends feedback, and authorizes exceptions. Figure 8 shows these actors within the system boundary; operational deployment should add administrator functions for access control, retention, and model-policy configuration. After gate completion, final submission remains a student action, whereas task configuration, evidence audit, override authorization, and intervention feedback are instructor-only actions. Access-control, retention, and model-policy configuration are reserved for an administrator role in operational deployment.

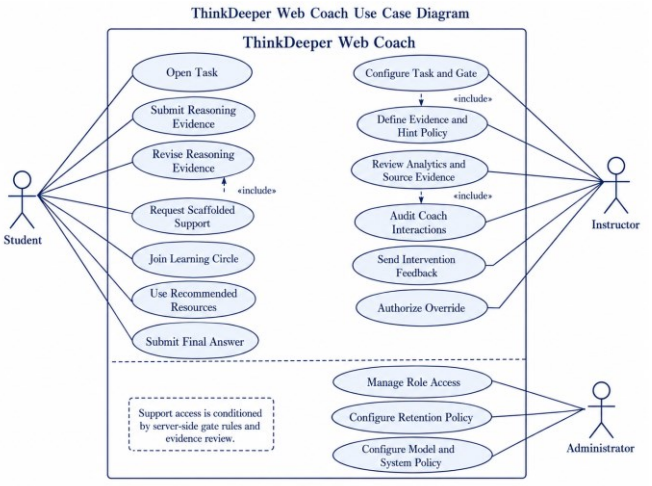

Figure 8. Use Case Diagram of ThinkDeeper Web Coach.

DOI: 10.34148/teknika.v15i2.1504

E. Evaluation Plan

Because this article reports design and development rather than completed classroom implementation, evaluation must proceed in stages (Table 7). Expert review examines construct coverage, requirement relevance, prompt quality, architecture, safety controls, and dashboard interpretability. A cognitive walkthrough and usability study then test whether students and lecturers can complete core tasks. Trace validation compares system indicators with human-coded reasoning, and a classroom pilot evaluates feasibility and preliminary learning outcomes without treating the prototype as already effective.

Primary outcomes should extend beyond final grades. Suitable measures include rubric-rated reasoning depth, quality and coverage of test cases, explanation specificity, revision after feedback, appropriate help use, self-regulated learning behavior, and instructor-rated readiness for independent problem solving. Process measures should include gate completion, time to revision, hint escalation, bypass attempts, peer-critique uptake, resource completion, and lecturer overrides.

Table 7. Staged Evaluation Plan

Stage	Evidence
Expert review	CVI/CVR, architecture and safety comments, requirement-level revision matrix.
Cognitive walkthrough	Student and lecturer task success, error points, policy comprehension, walkthrough notes.
Usability test	SUS, UEQ, time-on-task, errors, think-aloud evidence, perceived workload.
Trace validation	Human rubric scores, inter-rater agreement, indicator agreement, false-positive/negative analysis.
Safety and security test	Gate-bypass attempts, answer leakage, unsupported recommendations, guard and override logs.
Classroom pilot	Pre/post reasoning rubric, test-case quality, logs, lecturer workload, interviews, feasibility.

Expert review should involve at least three to five specialists in educational technology, IT education, learning analytics, and software architecture. Content validity may be summarized using Formula 1:

$$\text{CVR} = \frac{n_e - \frac{N}{2}}{\frac{N}{2}} \text{ and I-CVI} = \frac{n_e}{N} \quad (1)$$

Where n_e is the number of experts rating an item as essential or relevant and N is the total number of experts. The review should produce a requirement-level revision matrix rather than only an overall acceptability score.

Usability testing should use task-based scenarios for both actors and report SUS, UEQ, completion time, errors, and think-aloud findings. Trace validation should compare evidence-linked indicators with independently rated reasoning rubrics using inter-rater agreement, descriptive analysis, and correlation where assumptions are satisfied. A classroom pilot should triangulate pre/post reasoning evidence, logs, rubric scores, and interviews. Security and safety testing should additionally examine unauthorized gate bypass, answer leakage, unsupported recommendations, and the reliability of lecturer overrides.

F. Critical Discussion, Practical Implementation, and Limitations

The theoretical contribution is the PVLDF, which treats depth of thinking as an evidence-producing process rather than an opaque score. Unlike a general conversational assistant, ThinkDeeper makes assistance conditional on visible reasoning and limits the form of permitted support. Unlike a conventional intelligent tutor, it emphasizes student-authored assumptions, plans, tests, and explanations across open-ended IT tasks. Unlike a learning-analytics dashboard, it does not begin with retrospective visualization; it first structures the activity that generates interpretable traces. The practical value is therefore the coupling of pedagogy, trace provenance, and instructor judgment.

Implementation in a real course should begin with one recurring task type rather than an entire curriculum. The lecturer defines the evidence schema and rubric, configures three or four hint levels, uploads approved resources, assigns peer roles, and pilots the gate with a small group. Weekly audit sampling can reveal whether students submit superficial text merely to unlock support, whether guardrails over-block legitimate help, and whether indicators create excessive review work. The main trade-offs are productive struggle versus frustration, assistance versus cognitive substitution, diagnostic visibility versus surveillance, and automation versus lecturer workload. Analytics should therefore support decisions, not become automatic grades.

The study has several limitations. First, it establishes design plausibility rather than classroom effectiveness. Second, the corpus is recent, heterogeneous, and only partly centered on IT; adjacent-disciplinary studies support mechanisms but do not prove transfer to every IT course. Third, 77 reports could not be retrieved, which may introduce availability bias. Fourth, the structured extraction process included model-assisted coding and therefore requires source checking before future quantitative synthesis. Fifth, required evidence fields may privilege written fluency, encourage performative compliance, or oversimplify nonlinear reasoning. Future work must evaluate construct validity, accessibility, privacy, teacher workload, guardrail reliability, and learning impact in authentic courses. Sixth, screening was not conducted as a blinded multi-reviewer process and no inter-rater agreement coefficient was calculated; later evidence syntheses should add independent screening and adjudication.

DOI: 10.34148/teknika.v15i2.1504

IV. CONCLUSION

This study designed ThinkDeeper Web Coach as a literature-informed specification for process-visible reasoning in problem-solving-intensive IT courses. The evidence base comprised 51 studies retained from 500 records indexed in Semantic Scholar and OpenAlex. The revised synthesis distinguishes pedagogical, design, and evaluation gaps and organizes the evidence through the Process-Visible Learning Design Framework. The framework indicates that depth-oriented support is most defensible when it elicits explicit reasoning, uses question-driven scaffolding, preserves revision traces, structures peer elaboration, recommends resources in response to diagnosed gaps, and keeps lecturers in the decision loop.

The resulting specification comprises six student-learning modules and a cross-cutting instructor-governance layer. It clarifies applicable course contexts, evidence-to-requirement derivation, a hybrid technical architecture, server-side gate rules, constrained model assistance, output guardrails, auditable trace data, and a staged evaluation protocol. These additions make the artifact more reproducible and technically explicit while preserving the central limitation: ThinkDeeper has not yet demonstrated classroom effectiveness. Expert validation, usability and safety testing, trace validation, and a classroom pilot are required before claims about improved reasoning can be made.

REFERENCES

[1] L. T. Ameen, M. R. Yousif, N. A. J. Alnoori, and B. H. Majeed, "The Impact of Artificial Intelligence on Computational Thinking in Education at University," *Int. J. Eng. Pedagogy*, vol. 14, no. 5, pp. 192–203, 2024, doi: 10.3991/ijep.v14i5.49995.

[2] H.-L. Liu, C.-F. Lai, and H.-C. K. Lin, "Effects of Facial Recognition and Text Semantic Recognition on Affective Tutoring System," *J. Internet Technol.*, vol. 25, no. 6, pp. 807–814, 2024, doi: 10.70003/160792642024112506001.

[3] M. Lu and Z. Hu, "Leveraging Multimodal Information for Web Front-End Development Instruction: Analyzing Effects on Cognitive Behavior, Interaction, and Persistent Learning," *Information*, vol. 16, no. 9, p. 734, 2025, doi: 10.3390/info16090734.

[4] R. H. Sakti *et al.*, "Diving into the Future: Unravelling the Impact of Flowgorithm and Discord Fusion on Algorithm and Programming Courses and Fostering Computational Thinking," *Int. J. Learn. Teach. Educ. Res.*, vol. 23, no. 7, pp. 347–367, 2024, doi: 10.26803/ijlter.23.7.18.

[5] Y. Xu, J. Zhu, M. Wang, F. Qian, Y. Yang, and J. Zhang, "The Impact of a Digital Game-Based AI Chatbot on Students' Academic Performance, Higher-Order Thinking, and Behavioral Patterns in an Information Technology Curriculum," *Appl. Sci.*, vol. 14, no. 15, p. 6418, 2024, doi: 10.3390/app14156418.

[6] C.-C. Lee and M. Y. H. Low, "Using GenAI in Education: The Case for Critical Thinking," *Front. Artif. Intell.*, vol. 7, p. 1452131, 2024, doi: 10.3389/frai.2024.1452131.

[7] Q. Liu and C.-C. Tu, "Improving Critical Thinking Through AI-Supported Socio-Scientific Issues Instruction," *J. Logist. Inform. Serv. Sci.*, vol. 11, no. 3, pp. 52–65, 2024, doi: 10.33168/JLISS.2024.0304.

[8] S. Ma, J. Wang, Y. Zhang, X. Ma, and A. Y. Wang, "DBox: Scaffolding Algorithmic Programming Learning Through Learner-LLM Co-Decomposition," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, in CHI '25. New York, NY, USA: Association for Computing Machinery, 2025. doi: 10.1145/3706598.3713748.

[9] P. Maiti and A. K. Goel, "How Do Students Interact with an LLM-Powered Virtual Teaching Assistant in Different Educational Settings?" 2024. doi: 10.48550/arXiv.2407.17429.

[10] Q. Pan *et al.*, "AMQuestioner: Training Critical Thinking with Question-Driven Interactive Argument Maps in Online Discussion," *Proc. ACM Hum.-Comput. Interact.*, vol. 9, no. CSCW, 2025, doi: 10.1145/3757551.

[11] S. Park, H. Subramonyam, and C. Kulkarni, "Thinking Assistants: LLM-Based Conversational Assistants That Help Users Think by Asking Rather Than Answering." 2023. doi: 10.48550/arXiv.2312.06024.

[12] C. Borchers, J. Zhang, R. S. Baker, and V. Aleven, "Using Think-Aloud Data to Understand Relations between Self-Regulation Cycle Characteristics and Student Performance in Intelligent Tutoring Systems," in *LAK '24: Proceedings of the 14th Learning Analytics and Knowledge Conference*, New York, NY, USA: Association for Computing Machinery, 2024, pp. 529–539. doi: 10.1145/3636555.3636911.

[13] V. Dornauer, M. Netzer, É. Kaczkó, L.-M. Norz, and E. Ammenwerth, "Automatic Classification of Online Discussions and Other Learning Traces to Detect Cognitive Presence," *Int. J. Artif. Intell. Educ.*, vol. 34, pp. 395–415, 2024, doi: 10.1007/s40593-023-00335-4.

[14] G. Ramaswami, T. Sušnjak, and A. Mathrani, "Effectiveness of a Learning Analytics Dashboard for Increasing Student Engagement Levels," *J. Learn. Anal.*, vol. 10, no. 3, pp. 115–134, 2023, doi: 10.18608/jla.2023.7935.

[15] V. Serrano, J. Cuadros, L. Fernández-Ruano, J. García-Zubía, U. Hernández-Jayo, and F. Lluch, "Learning Analytics Dashboards for Assessing Remote Labs Users' Work: A Case Study with VISIR-DB," *Technol. Knowl. Learn.*, vol. 30, no. 1, pp. 263–290, 2025, doi: 10.1007/s10758-024-09752-3.

[16] G. Guo, A. M. S. Kumar, A. Gupta, A. J. Coscia, C. J. MacLellan, and A. Endert, "Visualizing Intelligent Tutor Interactions for Responsive Pedagogy," in *Proceedings of the 2024 International Conference on Advanced Visual Interfaces*, in AVI '24. New York,

NY, USA: Association for Computing Machinery, 2024. doi: 10.1145/3656650.3656667.

[17] P. Utamachant, C. Anutariya, and S. Pongnumkul, "i-Ntervene: Applying an Evidence-Based Learning Analytics Intervention to Support Computer Programming Instruction," *Smart Learn. Environ.*, vol. 10, no. 1, 2023, doi: 10.1186/s40561-023-00257-7.

[18] Y. Uzun, W. Suraworachet, Q. Zhou, A. Gauthier, and M. Cukurova, "Engagement with Analytics Feedback and Its Relationship to Self-Regulated Learning Competence and Course Performance," *Int. J. Educ. Technol. High. Educ.*, vol. 22, no. 1, 2025, doi: 10.1186/s41239-025-00515-3.

[19] G. L. Akinyi, R. O. Oboko, and L. Muchemi, "Learning Analytics Intervention Using Prompts and Feedback for Measurement of e-Learners' Socially-Shared Regulated Learning," *Electron. J. E-Learn.*, vol. 22, no. 5, pp. 103–116, 2024, doi: 10.34190/ejel.22.5.3253.

[20] R. A. Lobo-Quintero, "AI-Enhanced Think-Pair-Share: A Learning Analytics Approach to Foster Linguistic Creative Thinking and Collaborative Learning," *J. Learn. Anal.*, vol. 12, no. 2, pp. 19–34, 2025, doi: 10.18608/jla.2025.8807.

[21] M. Mielikäinen and E. Viippola, "ICT Engineering Students' Perceptions on Project-Based Online Learning in Community of Inquiry (CoI)," *SAGE Open*, vol. 13, no. 3, 2023, doi: 10.1177/21582440231180602.

[22] G. Psathas, S. Tegos, S. N. Demetriadis, and T. Tsiatsos, "Exploring the Impact of Chat-Based Collaborative Activities and SRL-Focused Interventions on Students' Self-Regulation Profiles, Participation in Collaborative Activities, Retention, and Learning in MOOCs," *Int. J. Comput.-Support. Collab. Learn.*, vol. 18, pp. 329–351, 2023, doi: 10.1007/s11412-023-09394-0.

[23] T. T. Dien, T.-H. Nguyen, and T.-N. Nguyen, "Novel Approaches for Searching and Recommending Learning Resources," *Cybern. Inf. Technol.*, vol. 23, no. 2, pp. 151–169, 2023, doi: 10.2478/cait-2023-0019.

[24] M. Domino and C. A. Shaffer, "Using a Wider Digital Ecosystem to Improve Self-Regulated Learning," *Front. Educ.*, vol. 10, p. 1487344, 2025, doi: 10.3389/feduc.2025.1487344.

[25] T. Pham-Duc, "PERS: A Personalized Recommender System for Student-Generated Questions in Programming Courses," in *Proceedings of the International Conference on Computers in Education*, 2024. doi: 10.58459/icce.2024.4913.

[26] J.-W. Tzeng, N.-F. Huang, A.-C. Chuang, T.-W. Huang, and H.-Y. Chang, "Massive Open Online Course Recommendation System Based on a Reinforcement Learning Algorithm," *Neural Comput. Appl.*, vol. 37, pp. 11607–11618, 2023, doi: 10.1007/s00521-023-08686-8.

[27] S. McKenney and T. C. Reeves, "Educational Design Research: Portraying, Conducting, and Enhancing Productive Scholarship," *Med. Educ.*, vol. 55, no. 1, pp. 82–92, 2021, doi: 10.1111/medu.14280.

[28] N. Norouzi and A. Mazaheri, "Context-Aware Analysis of Group Submissions for Group Anomaly Detection and Performance Prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, pp. 15938–15946. doi: 10.1609/aaai.v37i13.26892.