

UDC 004.032.26(045)
DOI:10.18372/1990-5548.89.21471

Oleksandr Radovetskyi

A PRIORITY-ORDERED DECISION MODEL FOR EDUCATIONAL LLM AGENTS

Vadym Hetman Kyiv National Economic University, Kyiv, Ukraine
E-mail: radovetskyioleksandr@gmail.com, ORCID 0009-0006-2446-6480

Abstract—This work is devoted to the design of a decision-support architecture for artificial intelligence agents used in higher-education. The paper proposes a priority-ordered decision model that ranks three evaluation signals – divergence between self-assessment and teacher assessment, performance trend, and peer comparison – before any call to a language model, so that the same input data always yields the same intervention priority. The model is based on a deterministic progress-weighting service defined by equations (1) – (9), with a course-configurable divergence threshold ($\tau = 0.15$ in the reference configuration) and four weight coefficients that sum to 1. The paper develops a reference implementation with a typed evidence schema, a FastAPI backend, and a tool-augmented LLM agent restricted to four read-only tools. A worked example shows that a flagged divergence ($d = 0.25 > \tau$) overrides an improving trend ($L: 0.58 \rightarrow 0.64 \rightarrow 0.67$), correctly prioritizing a calibration alert over a routine update. The results can be used to design auditable, LLM-based recommendation systems for education and other domains where several evidence signals must be reconciled before a generative model is invoked.

Keywords—Adaptive learning systems, divergence detection, educational decision support, learning analytics, reference architecture, tool-augmented LLM agents.

I. INTRODUCTION

Artificial intelligence (AI) assistants are now used by a large and fast-growing share of students and teachers, and the way such systems are built directly affects learning outcomes at scale; this is why an architectural weakness in one widely-copied design pattern is not a narrow engineering concern but a problem with a real societal cost, in wasted study time, unnoticed learning gaps, and unequal support across student subgroups. Many existing proposals for AI-based learning support describe an architecture only in general terms. They stop at a diagram of layers, a governance framework, or a scaffolding theory, without giving the relational schema, the deterministic scoring service, or the tool contract that a working system actually needs. This matters because the things a reviewer or a teacher cares about – whether the system can be audited, whether it degrades safely when a signal is missing, whether the language model is constrained rather than simply trusted – are properties of an implementation, not of a diagram. This article addresses that gap for one concrete case: a relational evidence schema, a deterministic weighting / divergence / trend service, and a large language model (LLM)-based tool-augmented agent, combined into a working FastAPI backend.

This article makes those commitments explicit. It gives: (1) a relational schema in which test score, self-assessment, and teacher assessment are

independent, nullable fields tied to a course's module / topic hierarchy; (2) a deterministic service that turns an assessment record into a weighted progress score, a signed divergence flag, and a trend label, using formulas simple enough to check by hand; and (3) an agent layer in which the language model never sees raw scores in its prompt, and instead calls one of four scoped tools to get exactly the evidence it needs, under a system prompt that the instructor configures per course. The aim of this article is to describe this architecture precisely enough – schema, formulas, tool contract, and prompt-priority order – that the design decisions behind it, not only the motivation for them, can be examined and reused.

II. RELATED ARCHITECTURES AND THE IMPLEMENTATION GAP

Intelligent assistants and generative models are increasingly used across universities to widen access to learning materials, personalize learning paths, and reduce teacher workload [1]. The AI component in such systems is often described as a “third agent” next to the student and the teacher – a framing used both in the AIED literature [2] and in a large-scale classroom co-design study [3]. Dialogic large language model (LLM) assistants, in particular, have made real-time explanation far more accessible [4]. The risks of this adoption are just as well documented, and they are specifically implementation risks. A model that answers directly

from raw context, without any scoring or evidence-reconciliation step, reproduces the “performance paradox”: a learner gets a correct-looking result without the cognitive effort the task was meant to require [5]. The same shortcut leads to “metacognitive laziness” [6], and a 2026 policy report reaches a similar conclusion, warning of a gradual “cognitive atrophy” of unaided skills [7]. Checking whether a system avoids this failure is itself difficult, since an overall accuracy figure can hide worse outcomes for smaller subgroups unless the evaluation is broken down [8].

Existing systems largely inherit this risk because their data layer is behavioural, not evidentiary: personalization is built on top of learning management system (LMS) interaction logs, and there is no place in the schema for an independent teacher judgement to be checked against a learner's own self-report [9]. Where a governance layer is proposed at all, it usually stays at the level of a principle – “Teacher Framing” [2] or a scaffolding framework derived from Evidence-Centered Design [10] – [12] – without committing to the schema, thresholds, or tool contract that would let another team actually rebuild the system.

The idea of separating layers goes back to intelligent tutoring systems (ITS), which used a domain model, a student model, and a pedagogical module for choosing a strategy [13]. That kind of separation is what this article returns to, but classical ITS paid for it with heavy, subject-specific engineering, and without access to a learner's metacognitive state, such systems could mistake a lucky correct answer for real mastery [13]. Contemporary LLM-based assistants removed the scalability problem, since natural-language interaction needs no hand-built domain model, but in doing so they usually also removed that layered decision structure. Assistants reviewed in the AIED literature are commonly connected directly to LMS platforms and use behavioural traces as their main input [1], [4], so nothing in the pipeline can tell superficial task completion apart from real understanding – there is no schema slot for an independent judgement to be checked against the behavioural trace [1].

Where a more disciplined pipeline has been proposed, it usually stays at the level of governance rather than a full specification. “Teacher Framing” keeps the instructor inside the decision loop [2], and Evidence-Centered Design (ECD) has been adapted into evidence-decision-feedback formulations that favour scaffolding over direct answers [10], [11]. These are useful principles, but none of them

commits to a schema, a scoring formula, or a tool contract precise enough for someone else to rebuild the system [14]. A related line of work treats the mismatch between self-assessment and independent evidence as a diagnostic variable – a “calibration discrepancy” that predicts later strategy choice [15] and correlates with hidden academic disadvantage [16], [17] – but this is usually computed after the fact, for a human reader, rather than produced by a running service that a downstream agent can consume.

Taken together, this body of work points to something the literature does not yet provide: a specification detailed enough to build from – a concrete evidence schema, a deterministic scoring and divergence service with named thresholds, and an explicit rule for what a generative model is and is not allowed to see. The rest of this article gives exactly that.

III. PROBLEM STATEMENT

Formally, the problem this article addresses is as follows. Given an assessment record with three evidence values – a test score $O \in [0, 100]$, a self-confidence rating $S \in [0, 10]$, and an optional teacher score $E \in [0, 100]$ – and a course configuration consisting of four weight coefficients $w_o, w_p, w_e, w_s \geq 0$ with $w_o + w_p + w_e + w_s = 1$, and a divergence threshold $\tau \in [0, 1]$, the task is to compute, deterministically and before any call to a generative language model: (1) a weighted progress score $L \in [0, 1]$; (2) a divergence flag and, when set, a signed direction; (3) a trend label over the student's last three weighted scores; and (4) a single, priority-ranked intervention action, such that the same $(O, S, E, \text{configuration})$ always yields the same action regardless of how a downstream prompt happens to be phrased.

The quality of a solution to this problem is judged against three metrics used throughout the rest of this article: auditability – whether every intermediate value (L , divergence, trend) can be inspected and independently recomputed by a teacher, rather than hidden inside a single opaque model call; safe degradation – whether the output stays well-defined when a signal (E or S) is missing, instead of silently treating a missing value as zero; and determinism – whether identical inputs always produce an identical priority ranking, independent of any language-model call or prompt wording.

Sections IV–VIII specify, in turn, a three-part architecture (Section IV), the deterministic scoring formulas that satisfy the auditability and safe-degradation metrics (Section V), the priority rule that satisfies the determinism metric (Section VI),

the tool-augmented agent layer that consumes the result (Section VII), and a reference implementation of all of the above (Section VIII). Section IX then traces a concrete example through the full pipeline to show the three metrics holding in practice, before Section X concludes.

IV. SYSTEM ARCHITECTURE

A. Three-part decomposition

The reference implementation is organized around three parts with a narrow interface between each, so that the language model at the end of the pipeline never has to be trusted with more than it is explicitly given.

- *Evidence schema (data layer)*. A relational schema, served by a FastAPI backend over SQLAlchemy, models a course as a Course $\rightarrow$ Module $\rightarrow$ Topic hierarchy and a learner as a User with a role (student or teacher). An Assessment row is scoped to exactly one Topic or one Module (enforced by a database check constraint) and carries three independently nullable evidence fields: test_score (0–100), self_confidence (0–10), and teacher_score (0–100). Nullability is deliberate: the schema does not force a self-assessment or a teacher score to exist before a test score can be scored.

- *Progress-weighting service*. A deterministic service consumes an Assessment row and a course-level WeightConfig record and produces three outputs – a weighted progress score, a divergence flag with a direction, and a trend label – which are persisted as a ProgressSnapshot. This service contains no LLM calls; it is ordinary, testable application code (Section V).

- *Tool-augmented agent layer*. An LLM-based agent consumes the persisted evidence indirectly, through four scoped tools, and drafts a natural-language recommendation under a role-specific, per-course-configurable system prompt (Section VII). Recommendations are themselves persisted as Recommendation rows, targeted at either the student or the teacher.

B. Configuration Surfaces

Two records make the pipeline configurable per course without touching code. WeightConfig holds four weight coefficients (for test score, historical performance, teacher score, self-assessment) and a divergence_threshold, editable by the instructor through the Evaluation Policy panel of the web application; the application programming interface (API) layer rejects an update whose four weights do not sum to 1. CoursePolicy holds two numeric thresholds (struggling/proficient) plus three free-

text fields – course context, student-facing guidance, and teacher-facing guidance – that are injected verbatim into the agent's system prompt (Section VII). Both records are ordinary rows in the schema, not hard-coded constants, which is what lets the same backend serve courses with materially different grading philosophies.

V. THE PROGRESS-WEIGHTING SERVICE: SCORE, DIVERGENCE AND TREND

A. Normalization

The service first maps the three raw evidence fields of an assessment onto a common $[0, 1]$ scale, since test score and teacher score are recorded on a 0–100 scale while self-confidence is recorded on a 0–10 scale:

$$O_n = \frac{O}{100}, \quad S_n = \frac{S}{10}, \quad E_n = \frac{E}{100}. \quad (1)$$

All subsequent formulas in this section operate on the normalized values O_n, S_n, E_n rather than on the raw fields.

B. Historical Performance Average

Let $\{O_{n,1}, \dots, O_{n,n}\}$ be the student's normalized test scores from prior assessments in the same course ($n \geq 0$). The historical performance term P is:

$$P = \frac{1}{n} \sum_{i=1}^n O_{n,i} \quad (n > 0), \quad (2)$$

$$P = O_n \quad (n = 0),$$

that is, the mean of prior normalized test scores in the same course, falling back to the current score itself on a student's first attempt.

C. Weighted Progress Score

Given the four normalized terms O_n, P, E_n, S_n and the course's weight coefficients w_O, w_P, w_E, w_S , the weighted progress score is:

$$L = \frac{w_O O_n + w_P P + w_E E_n + w_S S_n}{w_O + w_P + w_E + w_S}. \quad (3)$$

The denominator here is not a fixed constant equal to 1 – it only reaches 1 in the special case where all four terms are active, by the constraint in equation (4). Whenever a term is dropped because its field is null (Section V-A), that term's weight is removed from the denominator too, so the remaining weights are renormalized rather than left summing to less than 1. In other words, equation (3) is written in its general form so that it stays correct in the incomplete-data case; it only simplifies to a plain weighted sum with

denominator 1 when a student has submitted a test score, a teacher score, and a self-assessment together – the renormalization does real work precisely when data is missing, which is the case a fixed-weight average would otherwise get wrong.

Subject to the constraint enforced by the API layer whenever an instructor edits the course's WeightConfig:

$$w_O + w_P + w_E + w_S = 1. \quad (4)$$

The E_n and S_n terms, together with their weights, are dropped from both the numerator and the denominator of equation (3) whenever the corresponding field is null; the denominator is therefore the sum of only the active weights, not a constant. This renormalization over active weights, rather than treating a missing signal as zero, is what lets the schema's nullability (Section IV-A) behave sensibly: a student who has not yet submitted a self-assessment is not penalized as if they had reported zero confidence.

D. Divergence Flag and Direction

When both E_n and S_n are defined for the same assessment, the service computes the divergence magnitude:

$$d = |E_n - S_n|, \quad (5)$$

and, if d exceeds the course's `divergence_threshold` τ , attaches a signed direction rather than only a flag:

$$\text{direction}(E_n, S_n) = \text{OVERCONFIDENT} \text{ if } S_n > E_n, \quad (6)$$

with the symmetric case (UNDERCONFIDENT if $S_n \leq E_n$) completing the rule, defined only when $d > \tau$; the assessment's `divergence_flag` is set to true

exactly when this condition holds. Carrying the sign forward, rather than only the magnitude d , is what allows the agent layer (Section VII) to give qualitatively different advice for the two cases.

E. Trend Label

Writing the weighted scores of the two most recent prior ProgressSnapshot rows and the current one, in chronological order, as L_1, L_2, L_3 (Section V-C), the trend label is the piecewise rule:

$$\text{trend}(L_1, L_2, L_3) = \text{IMPROVING} \text{ if } L_1 < L_2 < L_3, \quad (7)$$

with the symmetric case (DECLINING if $L_1 > L_2 > L_3$) and the default case (STABLE otherwise, including whenever fewer than two prior snapshots exist) completing the rule. This intentionally simple monotonic condition is cheap enough to recompute on every submission and transparent enough that a teacher can verify it from the three numbers alone.

F. Snapshot Pipeline

On every new Assessment, the service executes the following steps: (1) resolve the owning course from the assessment's topic or module; (2) collect the student's prior test scores in that course to compute P via equation (2); (3) load the course's WeightConfig and compute L via equation (3); (4) if both E_n and S_n are present, compute the divergence magnitude and direction via equations (5), (6); (5) compute the trend label via equation (7) against the two most recent snapshots; (6) persist the result as a new ProgressSnapshot row. Only the persisted snapshot and the flagged assessment – not the raw computation – are visible to the agent layer described next. Algorithm 1 gives the same six steps as pseudocode.

Algorithm 1. Compute_Progress_Snapshot (Assessment, Config)

```

Input:  assessment (O, S may be null, E may be null)
        config (w_O, w_P, w_E, w_S, divergence_threshold  $\tau$ )
Output: ProgressSnapshot (L, divergence_flag, direction, trend)

1:  course_id  $\leftarrow$  resolve_course(assessment)
2:  prior  $\leftarrow$  get_prior_test_scores(assessment.student_id, course_id)
3:  P  $\leftarrow$  eq.(2) applied to prior
4:  terms  $\leftarrow$  { (O/100, w_O), (P, w_P) }
5:  if E  $\neq$  null: terms  $\leftarrow$  terms  $\cup$  { (E/100, w_E) }
6:  if S  $\neq$  null: terms  $\leftarrow$  terms  $\cup$  { (S/10, w_S) }
7:  W  $\leftarrow$   $\Sigma$  weight for (value, weight) in terms
8:  if W = 0: return L  $\leftarrow$  0
9:  L  $\leftarrow$   $\Sigma$  (value  $\cdot$  weight) for (value, weight) in terms, divided by W // eq. (3)
10: flag  $\leftarrow$  false; direction  $\leftarrow$  null
11: if E  $\neq$  null and S  $\neq$  null:
12:     d  $\leftarrow$  |E/100 - S/10| // eq. (5)
13:     if d >  $\tau$ :
14:         flag  $\leftarrow$  true
15:         direction  $\leftarrow$  eq. (6)
16: prior_snapshots  $\leftarrow$  last 2 ProgressSnapshot rows for (student, scope)
17: trend  $\leftarrow$  eq. (7) applied to (prior_snapshots, L)
18: snapshot  $\leftarrow$  persist ProgressSnapshot(L, flag, direction, trend)
19: return snapshot

```

VI. SIGNAL PRIORITY AND SEVERITY ORDERING

The weighted score, divergence flag and trend label computed in Section V are not combined into a single risk index; instead, the agent's system prompt (Section VII) imposes an explicit priority ordering over them when several fire at once, so that a teacher reading a recommendation can tell which signal drove it. Before that ordering can be stated, one further signal needs a formal definition: how a student compares to their class.

A. Peer-comparison Indicator

Let μ be the class average test score for the current scope, as returned by `get_class_statistics` (Table III), and let $\Delta = O - \mu$ be the student's raw deviation from it. The peer-comparison indicator used by the agent's prompt is:

$$\text{peer}(\Delta) = \text{SUPPORT_CASE} \quad \text{if } \Delta < -15, \quad (8)$$

with **ENRICHMENT** if $\Delta > 15$, and **NONE** otherwise, completing the rule – the same ± 15 -point

band the student-facing and teacher-facing system prompts state explicitly (Section VII-B).

B. Priority Decision Rule

Table I gives that ordering as implemented in the teacher-facing prompt; the student-facing prompt uses the same ordering but phrases the required action supportively rather than as a directive. Written as a single rule over the three boolean-or-categorical signals – the divergence flag from equation (6), the trend label from equation (7), and the peer indicator from equation (8) – the agent's required action is the first branch of:

$$\text{action} = \text{divergence-action} \quad \text{if flag} = \text{true}, \quad (9)$$

with the remaining branches, in strictly decreasing priority, being trend-action if flag = false and trend $\in \{\text{IMPROVING}, \text{DECLINING}\}$; peer-action if flag = false, trend = STABLE, and peer(Δ) $\neq$ NONE; and a default light-touch action otherwise. Table I spells out the concrete action text for each branch.

TABLE I. SIGNAL-PRIORITY MATRIX ENFORCED BY THE AGENT'S SYSTEM PROMPTS

Priority	Signal	Condition	Required agent action
1	Divergence flag	$ E/100 - S/10 > \text{divergence_threshold}$	Name overconfidence or underconfidence explicitly; for the teacher prompt, suggest an oral check or a calibration exercise
2	Trend	declining / improving (Section V-C)	Declining: recommend a 1-on-1 check-in and cite a concrete study strategy; improving: reinforce the likely cause and set a next target
3	Peer comparison	>15 points below or above the class average (from <code>get_class_statistics</code>)	Below: flag as a support case; above: consider an enrichment assignment
4	No urgent signal	none of the above triggered	Confirm the student is on track; suggest a light-touch action

Divergence is placed above trend in equation (9) deliberately: a student whose self-confidence and teacher assessment disagree by more than the configured threshold is treated as the higher-priority case even if their trend is merely stable, because an unflagged calibration gap is the failure mode most likely to go unnoticed by a purely score-based dashboard [15]. Peer comparison is placed last because, unlike divergence and trend, it says nothing about this specific student's trajectory on its own — it is only informative once the higher-priority, individual-level signals have been checked.

C. Worked Example (Synthetic Data)

The following synthetic trace (Table II) illustrates equations (1) – (9) end to end for one assessment, using round numbers rather than data from the seeded demonstration course. Take a course with weights

$w_O = 0.40$, $w_P = 0.20$, $w_E = 0.25$, $w_S = 0.15$ (summing to 1, equation (4)) and divergence_threshold $\tau = 0.15$. A student submits an assessment with test score $O = 72$, teacher score $E = 55$, and self-confidence $S = 8$; their two prior test scores in the course were 60 and 66, and their two most recent prior weighted scores were 0.58 and 0.64.

Because the divergence flag is set (Priority 1 in Table I), the required action is the overconfidence branch of equation (9) regardless of the trend: the teacher-facing recommendation opens with an overconfidence alert, even though the IMPROVING trend and unremarkable peer comparison alone would have suggested a routine update. This is exactly the ordering Table I specifies: a system that surfaced trend or peer comparison first would under-report a calibration gap that equation (9) is designed to catch first.

TABLE II. WORKED-EXAMPLE TRACE THROUGH EQUATIONS (1) – (9) FOR ONE SYNTHETIC ASSESSMENT

Quantity	Value	Equation
Weights (w_O, w_P, w_E, w_S)	0.40, 0.20, 0.25, 0.15	eq. (4)
Normalized (O_n, P, E_n, S_n)	0.72, 0.63, 0.55, 0.80	eq. (1)–(2)
Weighted score L	$0.6715 \approx 0.67$	eq. (3)
Divergence d , flag, direction	$d = 0.25 > \tau (0.15) \rightarrow$ flagged, OVERCONFIDENT	eq. (5)–(6)
Trend (L_1, L_2, L_3)	0.58, 0.64, 0.67 $\rightarrow$ IMPROVING	eq. (7)
Peer comparison (μ, Δ)	$\mu = 65, \Delta = +7 \rightarrow$ NONE	eq. (8)
Priority decision	flag = true $\rightarrow$ divergence-action (Table I, Priority 1)	eq. (9)

VII. TOOL-AUGMENTED AGENT LAYER

The agent layer is the only part of the system that calls a large language model, and it is built so that the model's inputs are bounded and its unavailability is non-fatal. Its design follows the now-common tool-calling pattern in which a model interleaves reasoning with calls to external functions and conditions its next step on the result [18], extended with the earlier observation that a model can be trained or prompted to decide for itself which tool to call, when, and how to use the result, rather than following a fixed script [19].

A. Tool-calling Loop

A recommendation request supplies the agent with a short initial context – student name and id,

scope, the latest assessment's raw fields, the divergence flag if set, and the current weighted score and trend – together with the four tools in Table III. The agent runs in a loop: it may call any subset of the tools, in any order, before producing its final text; the backend executes each requested tool against the database and returns the result as a `tool_result` block, and the loop continues until the model emits an `end_turn` response. No tool grants write access; all four are read-only queries scoped to the current student and topic or module. This single, composable loop – rather than a multi-agent framework with separate planner and executor roles – follows the practical guidance that the most reliable agent deployments favour simple, inspectable patterns over additional architectural complexity [20].

TABLE III. TOOLS EXPOSED TO THE AGENT INSIDE THE TOOL-CALLING LOOP

Tool	Purpose	Typical trigger
<code>get_scope_name</code>	Resolve the human-readable name of the topic or module under discussion	Always called first, to anchor the response
<code>get_assessment_history</code>	Retrieve the student's last 1–10 assessments in the current scope	Detecting a pattern across attempts, not just the latest one
<code>get_class_statistics</code>	Return count, average, min and max test score for the current scope across all students	Deciding whether the student is an outlier or part of a class-wide pattern
<code>get_learning_resources</code>	Fetch curated materials filtered by difficulty (foundational or advanced) for the current scope	Selecting what to recommend once the difficulty tier is decided

B. Role-specific, Policy-conditioned Prompts

Two system prompts exist, selected by the target of the request (student or teacher), and both are templated against the course's `CoursePolicy` at request time rather than hard-coded. The `struggling_threshold` and `proficient_threshold` values from the policy are interpolated directly into the prompt text, and the policy's free-text `course_context`, `student_guidance` and `teacher_guidance` fields are appended verbatim when non-empty. This is the mechanism by which an instructor's own grading philosophy – for example, a department rule that a student on a declining trend across more than two consecutive assessments must be referred to academic support – becomes part of what the agent is instructed to do, without any code change.

The student-facing prompt asks for three to five sentences addressed to the learner by name, forbids

giving direct answers to academic questions, and requires that any resource returned by `get_learning_resources` be connected explicitly to something observed in the data rather than listed as a bare link. The teacher-facing prompt asks for professional advisory language, opens with a one-sentence status summary, and follows the priority ordering of Table I explicitly in its instructions.

C. Deterministic Fallback

When no LLM API credentials are configured, the agent layer does not fail closed; it returns a deterministic, template-based response assembled directly from the weighted score, trend label and divergence direction, using the same qualitative branches (declining / improving / stable; overconfident / underconfident) that the LLM prompts describe in prose. This keeps the rest of the pipeline – schema, weighting service, API – testable

and demoable without any dependency on external model availability, and it gives the system a defined, inspectable behaviour in the absence of the LLM rather than an undefined one.

VIII. IMPLEMENTATION

The reference implementation is a FastAPI backend with SQLAlchemy models, organized so that each architectural part in Sections IV–VII maps onto its own code module: a models module for the schema of Section IV-A (Course, Module, Topic, User, Assessment, WeightConfig, CoursePolicy, ProgressSnapshot, Recommendation, LearningResource); a services module for the deterministic progress-weighting logic of Section V; an agent module for the tool-calling loop and system prompts of Section VII; and an api module that exposes both as a representational state transfer (REST) interface. This separation keeps the deterministic scoring code testable on its own, without a running language model, and keeps the agent module the only part of the codebase that has to import an LLM client at all.

A. API Surface

Three groups of endpoints cover the pipeline end to end. Assessment endpoints accept a new test_score, self_confidence, or teacher_score for a student and trigger the progress-weighting service of Section V, which computes and persists a ProgressSnapshot. Configuration endpoints let an instructor read and update a course's WeightConfig and CoursePolicy records; the API layer rejects a WeightConfig update whose four weights do not sum to 1 (equation (4)), which is the one validation rule enforced outside the weighting service itself. Recommendation endpoints trigger the agent layer of Section VII for a given student and scope, and return the persisted Recommendation together with the trend and divergence values it was conditioned on, so a caller can check why a particular recommendation was produced.

B. Data Footprint

The schema in Section IV-A stores only the evidence fields needed by the weighting service – test_score, self_confidence, teacher_score, and the module / topic scope – and does not retain page-view or clickstream logs, which keeps the personal-data footprint close to the minimum the algorithm needs. This also narrows, though does not by itself resolve, the system's exposure under privacy regimes such as the Family Educational Rights and Privacy Act (FERPA) in the United States or the General Data Protection Regulation (GDPR) in the European Union: the only student-linked fields the agent's tools can return (see Table III) are the same records an institution already treats as protected educational data. The current implementation does not yet include per-tool-call audit logging or a retention policy for Recommendation rows, which a production deployment would need to add; this is flagged as a gap rather than a compliance claim.

IX. EXAMPLE OF SYSTEM OPERATION

This section traces one request through the full pipeline of Sections IV–VII, using the worked-example values already computed in Table II, to show the three metrics of Section III – auditability, safe degradation, and determinism – holding on a concrete case rather than only in the abstract.

The client submits an assessment with $O = 72$, $E = 55$, $S = 8$ for a student whose two most recent weighted scores were 0.58 and 0.64. The progress-weighting service (Algorithm 1) computes $L = 0.6715$, flags a divergence of 0.25 (above $\tau = 0.15$) in the OVERCONFIDENT direction, and labels the trend IMPROVING (see Fig. 1); all three values, together with the raw inputs, are persisted as a single ProgressSnapshot row and are individually queryable – this is the auditability metric in practice, since a teacher can recompute any of the three from the stored raw fields without re-running the service.

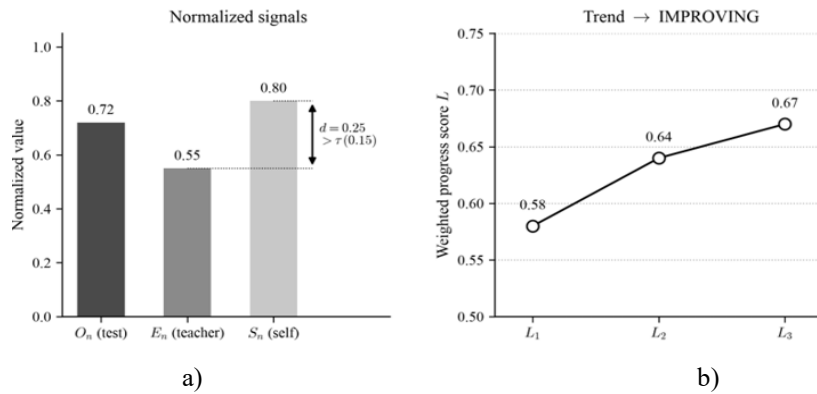


Fig. 1. Worked-example signals: (a) normalized test, teacher, and self-confidence scores, showing the divergence gap; (b) the weighted progress score trend across three assessments (see Table II)

The recommendation endpoint then invokes the agent layer with this snapshot. Applying the priority rule of equation (9), the divergence flag outranks the improving trend, so the agent's teacher-facing system prompt is instructed to open with an overconfidence alert rather than a routine progress note. With no LLM credentials configured, the deterministic fallback of Section VII-C assembles the following output directly from the persisted fields, without any language-model call:

“Status: needs attention. This student's self-assessment (8/10) is markedly higher than the teacher score recorded for this topic (55/100), a gap of 0.25 on the normalized scale, above the course threshold of 0.15. This is flagged as overconfidence. Although the student's weighted progress score has improved over the last three assessments (0.58 → 0.64 → 0.67), the calibration gap takes priority: consider a short oral check or a calibration exercise before assuming the improving trend reflects genuine mastery.”

Two properties of this output are worth making explicit. First, safe degradation: had the self_confidence field been null instead of 8, the divergence branch of the pipeline would simply not have executed (Algorithm 1, line 11), and the agent would have fallen through to the trend branch of equation (9) without any special-casing in the agent layer itself. Second, determinism: re-submitting the identical (O , E , S) values, with the identical WeightConfig and CoursePolicy, reproduces the identical L, divergence, trend, and — in the fallback path — the identical output text, because every step from Algorithm 1 through equation (9) is a pure function of its inputs; only the LLM-authored phrasing (when credentials are configured) can vary between runs, and even then the priority it must express, per Table I, does not.

X. CONCLUSIONS

This article described a reference implementation for an AI-mediated progress-monitoring system in higher education, built from a typed evidence schema, a deterministic weighting/divergence/trend service, and a tool-augmented, policy-conditioned agent layer. Against the three quality metrics of Section III, the architecture is auditable — every intermediate value can be inspected and independently recomputed by a teacher — degrades safely when a signal is missing, and is deterministic regardless of prompt wording; the worked example of Section IX demonstrated all three directly, on a concrete case, rather than only in principle.

The same pattern is not specific to progress monitoring. A deterministic, auditable scoring layer that constrains what a language model is allowed to see, followed by a small, tool-augmented agent, can be applied wherever several evidence signals — a test result, a self-report, an instructor's judgement, a peer comparison — must be reconciled before a generative model drafts a recommendation. Beyond higher education, this includes clinical decision support, coaching and performance-review tools, and other domains where over-trusting a single model call carries a real cost.

REFERENCES

- [1] R. Sajja, Y. Sermet, M. Cikmaz, D. Cwiertny, and I. Demir, “Artificial intelligence-enabled intelligent assistant for personalized and adaptive learning in higher education,” *Information*, vol. 15, no. 10, Article 596, 2024. <https://doi.org/10.3390/info15100596>
- [2] R. K. Jangale, S. B. Wankhade, S. N. Raut, and Y. S. Kute, “The impact of AI assistants on students' academic performance and learning behavior,” *International Journal on Science and Technology*, 2025. URL: <https://www.ijstat.org/papers/2025/4/9222.pdf>
- [3] Amplify Learn. AI, AI as a teaching partner: project report, Colleague AI Classroom. 2025. URL: https://www.amplifylearn.ai/wp-content/uploads/2025/12/AI_as_teaching_partner.pdf
- [4] R. Sajja, Y. Sermet, B. Fodale, and I. Demir, “Evaluating AI-powered learning assistants in engineering higher education: student engagement, ethical challenges, and policy implications,” arXiv. 2025. <https://doi.org/10.48550/arXiv.2506.05699>
- [5] Z. Xu, F. H. Choudhury, S. Ma, T. P. Murphrey, and K. E. Dooley, “Leveraging learning analytics to model student engagement in graduate statistics: a problem-based learning approach in agricultural education,” *Behavioral Sciences*, vol. 15, no. 10, Article 1360, 2025. <https://doi.org/10.3390/bs15101360>
- [6] H. Xue and Y. Niu, “Multi-output based hybrid integrated models for student performance prediction,” *Applied Sciences*, vol. 13, no. 9, Article 5384, 2023. <https://doi.org/10.3390/app13095384>
- [7] L. Loble and J. Lodge, “Artificial intelligence, cognitive offloading and implications for education,” *University of Technology Sydney / Australian Network for Quality Digital Education*. 2026. URL: <https://www.uts.edu.au/news/2026/03/experts-warn-unstructured-ai-use-in-schools-risks-cognitive-atrophy>

- [8] C. Herlihy, K. Truong, A. Chouldechova, and M. Dudík, "A structured regression approach for evaluating model performance across intersectional subgroups," arXiv. 2024. <https://doi.org/10.48550/arXiv.2401.14893>
- [9] E.-M. Schön, M. Neumann, C. Hofmann-Stölting, R. Baeza-Yates, and M. Rauschenberger, "How are AI assistants changing higher education?" *Frontiers in Computer Science*, vol. 5, Article 1208550. 2023. <https://doi.org/10.3389/fcomp.2023.1208550>
- [10] C. Cohn, S. Rayala, N. Srivastava, J. H. Fonteles, S. Jain, et al., "A theory of adaptive scaffolding for LLM-based pedagogical agents," arXiv. 2025. <https://doi.org/10.48550/arXiv.2508.01503>
- [11] C. Cohn, S. Guo, S. Rayala, H. D. Wang, N. Mohammed, et al., Evidence-decision-feedback: theory-driven adaptive scaffolding for LLM agents. arXiv. 2026. <https://doi.org/10.48550/arXiv.2602.01415>
- [12] Y. Fan, L. Tang, H. Le, K. Shen, S. Tan, et al., Beware of metacognitive laziness: effects of generative artificial intelligence on learning motivation, processes, and performance, *British Journal of Educational Technology*. 2024. <https://doi.org/10.1111/bjet.13544>
- [13] A. C. Graesser, S. D'Mello, and W. Cade, "Instruction based on tutoring. Handbook of research on learning and instruction," ed. by R. E. Mayer, P. A. Alexander. Routledge, 2011, pp. 408–426.
- [14] A. Mavroudi and G. Torgersen, "A large language model implementing an AI assistant in a higher education setting," *CEUR Workshop Proceedings*, vol. 3685, 2024. URL: <https://ceur-ws.org/Vol-3685/short13.pdf>
- [15] H. Lee and N. Bosch, "Calibration discrepancy predicts students' subsequent metacognitive strategy use in computer-based learning environments," *International Journal of Artificial Intelligence in Education*, vol. 35, no. 6, pp. 3746–3779. 2025. <https://doi.org/10.1007/s40593-025-00514-5>
- [16] M. D. Johnson, Learners' self-assessment and metacognition when using an open learner model with drill down: PhD thesis. University of Birmingham, 2018. URL: <https://etheses.bham.ac.uk/id/eprint/8590/>
- [17] C. Milton, V. Lokesh, and G. Thiruvengadam, "Examining the impact of AI-powered writing tools on independent writing skills of health science graduates," *Advanced Education*, vol. 12, no. 25, pp. 143–161, 2024. <https://doi.org/10.20535/2410-8286.315068>
- [18] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, et al., "ReAct: synergizing reasoning and acting in language models," *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)*, 2023. <https://doi.org/10.48550/arXiv.2210.03629>
- [19] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, et al., "Toolformer: language models can teach themselves to use tools," *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 68539–68551. <https://doi.org/10.52202/075280-2997>
- [20] E. Schluntz and B. Zhang, *Building effective agents. Anthropic Engineering Blog*. 2024. URL: <https://www.anthropic.com/engineering/building-effective-agents>.

Received: July 10, 2026

Accepted: August 12, 2026

Published: September 18, 2026

Radovetskyi Oleksandr. ORCID 0009-0006-2446-6480. Master's degree.

Department of Information Systems in Economics, Vadym Hetman Kyiv National Economic University, Kyiv, Ukraine. Research direction: applied AI systems and software architecture for education technology.

Education: Vadym Hetman Kyiv National Economic University, (2025).

Publications: 1.

E-mail: radovetskyioleksandr@gmail.com.

О. О. Радовецький. Модель ухвалення рішень з пріоритетизацією сигналів для освітніх LLM-агентів

Статтю присвячено розробці архітектури підтримки прийняття рішень для агентів штучного інтелекту, які використовуються у вищій освіті. У роботі запропоновано модель ухвалення рішень з пріоритетизацією, яка ранжує три сигнали оцінювання – розбіжність між самооцінкою та оцінкою викладача, тренд успішності та порівняння з однокурсниками – ще до будь-якого звернення до мовної моделі, завдяки чому ті самі вхідні дані завжди дають однаковий пріоритет втручання. Модель базується на детермінованому сервісі зважування прогресу, описаному формулами (1) – (9), з порогом розбіжності, що налаштовується для курсу ($\tau = 0,15$ у референсній конфігурації), та чотирма ваговими коефіцієнтами, сума яких дорівнює 1. У роботі розроблено референсну реалізацію з типізованою схемою даних, бекендом на FastAPI та агентом на основі LLM, обмеженим чотирма інструментами лише для читання. Розрахунковий приклад демонструє, що позначена розбіжність ($d = 0,25 > \tau$) перебиває тренд, що покращується ($L: 0,58 \rightarrow 0,64 \rightarrow 0,67$), правильно пріоритетизуючи сигнал калібрування над рутинним оновленням. Результати роботи можуть бути використані для проєктування прозорих LLM-систем рекомендацій в освіті та інших галузях, де потрібно узгоджувати кілька сигналів свідчень перед зверненням до генеративної моделі.

Ключові слова: адаптивні навчальні системи, аналітика навчання, детекція розбіжностей, LLM-агенти з інструментальним доповненням, освітня підтримка прийняття рішень, референсна архітектура.

Радовецький Олександр Олександрович. ORCID 0009-0006-2446-6480. Магістр.

Кафедра інформаційних систем в економіці, Київський національний економічний університет імені Вадима Гетьмана, Київ, Україна.

Освіта: Київський національний економічний університет імені Вадима Гетьмана, (2025).

Напрямок наукової діяльності: прикладні системи штучного інтелекту та архітектура програмного забезпечення для освітніх технологій.

Кількість публікацій: 1.

E-mail: radovetskyioleksandr@gmail.com