НАЦІОНАЛЬНИЙ АЕРОКОСМІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АЕРОКОСМІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова праця
на правах рукопису

Євдокимов Олександр Олегович

УДК 004.588:004.8:37.016:51

ДИСЕРТАЦІЯ
МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПІДТРИМКИ НАВЧАННЯ ТОЧНИМ НАУКАМ

122 Комп’ютерні науки
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ О. О. Євдокимов

Науковий керівник Чухрай Андрій Григорович, д.т.н., професор

Харків – 2026

АНОТАЦІЯ

Євдокимов Олександр Олегович. Моделі, методи та інформаційна технологія підтримки навчання точним наукам – кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки», галузь знань 12 «Інформаційні технології». Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, 2026.

Дисертаційну роботу присвячено розробленню та науковому обґрунтуванню комплексу математичних моделей, алгоритмічних методів і інформаційної технології підтримки навчання точним наукам, що забезпечують формалізовану діагностику знань, адаптивний вибір навчальних задач, алгоритмічну генерацію навчального контенту, інтеграцію формальної верифікації доведень і алгоритмів, а також мотиваційне регулювання навчальної діяльності.

Об'єктом дослідження є процес навчання точним наукам із використанням комп'ютерних та інтелектуальних навчальних систем.

Предметом дослідження є математичні моделі, алгоритмічні методи та інформаційна технологія підтримки навчання точним наукам, що забезпечують діагностику знань, адаптивний вибір задач, формальну верифікацію теоретичних положень і алгоритмів, генерацію навчального контенту та програмну реалізацію відповідних механізмів підтримки навчання.

У дисертаційній роботі виконано систематизований аналіз сучасного стану інтелектуальних навчальних систем, підходів knowledge tracing, адаптивних зовнішніх і внутрішніх циклів навчання, цифрових платформ підтримки вивчення математики та суміжних точних дисциплін, а також можливостей формальної верифікації у навчальному процесі. Показано, що наявні цифрові платформи ефективно підтримують масову перевірку відповідей і параметричну варіативність завдань, однак недостатньо забезпечують діагностику причин

помилок, підтримку доведень, виведення формул, побудову алгоритмів і контроль логічної строгості міркування.

Виявлено науково-прикладну суперечність між потребою в масштабованих індивідуалізованих засобах навчання точним наукам і обмеженістю наявних систем, що переважно орієнтовані на перевірку кінцевого результату без достатньої підтримки теоретичного мислення, формальної коректності та адаптивної діагностики. Для подолання цієї суперечності запропоновано інформаційну технологію, у якій поєднано моделі стану знань, діагностичні моделі помилок, алгоритмічну генерацію задач, формально верифіковані навчальні артефакти та засоби мотиваційного підкріплення.

Розроблено системну модель підтримки навчання точним наукам, що включає вербальний, структурний, функціональний, математичний та машинний рівні опису. У межах структурної моделі визначено компоненти інтелектуальної навчальної системи: студентський інтерфейс, модель студента, діагностичне ядро, генератор задач, модуль формальної верифікації, мотиваційний модуль і сховище навчальних даних. Функціональна модель подає навчання як замкнений цикл, у якому вибір або генерація задачі, розв’язання, перевірка, діагностика, оновлення стану знань, мотиваційний сигнал і добір наступного навчального впливу утворюють єдиний керований процес.

Формалізовано навчальні задачі точних наук як об’єкти, що містять множину понять і правил, можливі постановки, вхідні дані, допустимі відповіді та процедури отримання або перевірки результату. Запропоновано розмежування простих задач, орієнтованих на застосування відомого алгоритму, і складних задач, пов’язаних із побудовою алгоритмів, виведенням формул та доведенням тверджень. Для складних задач введено поняття формалізованості, що дає змогу визначати, який рівень автоматичної або напівавтоматичної перевірки може бути застосований у конкретному навчальному сценарії.

У роботі розроблено модель архітектури інтелектуальної навчальної системи для індивідуалізованого навчання точним наукам та алгоритмічних

дисциплін, яка інтегрує модель предметної області, модель студента з байєсівським оновленням стану знань, параметричну генерацію задач, діагностичні моделі, формальну верифікацію, LLM-сервіси, мотиваційні механізми та фіксацію навчальних дій студента;

Удосконалено метод параметричної генерації навчальних задач із автоматичною перевіркою коректності вхідних даних, умов задачі та еталонного результату, що забезпечує формування математично коректних варіантів задач і автоматизовану діагностику відповідей студента;

Удосконалено метод інтеграції формальної верифікації математичних доведень і алгоритмічних рішень у навчальний процес на основі Lean/Mathlib, що забезпечує покрокову перевірку логічної коректності міркувань, специфікацій і властивостей алгоритмів.

Дістала подальшого розвитку інформаційна технологія підтримки навчання точним наукам, яка забезпечує інтеграцію моделей предметної області та студента, алгоритмічної генерації навчальних задач, формальної перевірки математичних і алгоритмічних розв’язків, адаптивного керування навчальною траєкторією та засобів мотиваційного впливу.

Розроблено модель мотиваційного регулювання навчального процесу, у якій цифрові винагороди пов’язуються не з формальним фактом проходження завдання, а зі значущими навчальними подіями: самостійним виправленням помилки, демонстрацією структурного розуміння, наполегливою роботою без надмірних підказок, коректним доведенням або конструктивним внеском у навчальне середовище. У роботі обґрунтовано можливість реалізації такого підходу засобами блокчейну, токенізованих винагород і смартконтрактів за умови підпорядкування мотиваційного шару педагогічним цілям.

Окремо теоретично розглянуто можливість використання квантових алгоритмів для прискорення пошуку в просторі параметричних навчальних задач і сценаріїв. Такий розгляд не підміняє основних алгоритмічних рішень дисертації, але визначає перспективний напрям розвитку інформаційної

технології в умовах збільшення розмірності простору задач, параметрів, обмежень і навчальних траєкторій.

Практичне значення одержаних результатів полягає в тому, що основні наукові положення дисертаційної роботи доведено до рівня алгоритмічних рішень, програмних модулів і прототипу інтелектуальної навчальної програми «Мирна». Розроблений прототип забезпечує генерацію навчальних задач, перевірку правильності розв’язків, підтримку адаптивної логіки переходів між навчальними компонентами, елементи діагностики знань, подання формально верифікованого навчального матеріалу та мотиваційне регулювання навчального процесу.

Запропоновані моделі, методи та програмні рішення можуть бути використані у закладах вищої освіти для підтримки навчання математичних і суміжних точних дисциплін, у системах дистанційного та змішаного навчання, під час підготовки до стандартизованих іспитів з точних наук, а також при розробленні інтелектуальних освітніх платформ нового покоління. Результати роботи апробовано у наукових публікаціях, матеріалах міжнародних і всеукраїнських конференцій, постерних доповідях, звіті з науково-дослідної роботи та програмних прототипах.

Ключові слова: інтелектуальна навчальна система, точні науки, математика, STEM-освіта, адаптивне навчання, діагностична модель, knowledge tracing, баєсівське моделювання, формальна верифікація, Lean, Mathlib, proof-checking, параметрична генерація задач, мотиваційне регулювання, інформаційна технологія, прототип.

Список публікацій та особистий внесок здобувача:

За темою дисертаційної роботи опубліковано 12 наукових праць, у яких відображено основні положення, результати та висновки дисертації. Особистий внесок здобувача у спільних публікаціях зазначено після кожної позиції.

1. Kulik, A.; Trofymova, I.; Chukhray, A.; Stoliarenko, T.; Yevdokymov, O. Relevant Objectives of Developing SQL Adaptive Learning Technology. Proceedings of the 2022 IEEE 12th International Conference on Dependable Systems, Services and Technologies, DESSERT 2022. Conference paper. DOI: 10.1109/DESSERT58054.2022.10018757. EID: 2-s2.0-85147846036. Part of ISBN: 9798350333046. Greece. (Scopus). (*Особистий внесок: участь у формулюванні актуальних задач розроблення адаптивної технології навчання SQL; аналіз вимог до інтелектуальної підтримки навчання SQL; підготовка матеріалів, пов'язаних із навчальними сценаріями та апробацією підходу*)

2. Євдокимов О. О. Актуальні задачі розробки адаптивної технології навчання SQL. XVI Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті». Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Україна. *(Особистий внесок: постановка проблеми, систематизація актуальних задач розроблення адаптивної технології навчання SQL, підготовка тез і формулювання висновків)*

3. Chukhray, A.; Stoliarenko, T.; Yevdokymov, O. Development of a web-based SQL intelligent tutoring system SQLTOR. Poster. Digital Theme UK-Ukraine Research Twinning Conference, 27 March - 30 March 2023. *(Особистий внесок: участь у формуванні концепції веборієнтованої інтелектуальної навчальної системи SQLTOR, аналіз функціональних вимог і підготовка матеріалів постерної презентації)*

4. Чухрай А. Г., Євдокимов О. О. та ін. Розроблення і впровадження інтелектуальної комп'ютерної системи підготовки школярів України до НМТ з точних наук: звіт з НДР. Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського

«Харківський авіаційний інститут», 2024. *(Особистий внесок: участь у розробленні концепції інтелектуальної комп'ютерної системи підготовки до НМТ з точних наук, аналіз предметної області, формування вимог до адаптивної підтримки навчання та підготовка відповідних розділів звіту)*

5. Чухрай А. Г., Столяренко Т. Л., Євдокимов О. О., Дем'яненко В. А. Можливості використання інтелектуальних навчальних систем (ITS) в ЗВО для курсів вищої математики. Відкриті інформаційні та комп'ютерні інтегровані технології. № 102. Національний аерокосмічний університет «Харківський авіаційний інститут», 10.02.2025. DOI: 10.32620/oikit.2024.102.07. *(Особистий внесок: аналіз можливостей застосування ITS у курсах вищої математики; участь у класифікації типів математичних задач, визначенні обмежень наявних цифрових платформ і підготовці тексту статті)*

6. Кулік А.; Зеленяк О.; Чухрай А.; Прохоров О.; Яшина О.; Гавриленко О.; Євдокимов О.; Торжков А.; Заярний О. The concept of intelligent training system for Ukrainian school final STEM exam preparation. System Research and Information Technologies. Educational and Scientific Complex «Institute for Applied System Analysis» of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», 28.06.2025. DOI: 10.20535/SRIT.2308-8893.2025.2.09. (Scopus). *(Особистий внесок: участь у формуванні концепції інтелектуальної системи підготовки до фінального STEM-іспиту; аналіз вимог до діагностики знань, адаптивного добору задач і архітектури навчальної системи; підготовка фрагментів рукопису)*

7. Євдокимов О. О., Чухрай А. Г., Столяренко Т. Л. Діагностичні моделі та сенсо-економічний підхід на основі блокчейну в інтелектуальних навчальних системах. Матеріали І Науково-практичної конференції «Сучасні технологічні рішення для освіти, науки, бізнесу та міського господарства»: Секція 1. Рішення із використанням штучного інтелекту в галузі освіти та науки. Київ - Харків : ТОВ «Центр Дуальної освіти», 25 листопада 2025. 54 с. *(Особистий внесок: розроблення й опис ідеї поєднання діагностичних моделей із сенсо-економічним*

мотиваційним механізмом; формулювання блокчейн-орієнтованого підходу до цифрових винагород; підготовка тез)

8. Євдокимов О. О., Мандрікова Л. В. Інтелектуальні навчальні програми та інтегровані асистенти доведень у навчанні квантових обчислень і програмування. Інформаційні технології і автоматизація - 2025: матеріали XVIII міжнародної науково-практичної конференції. Одеса, 30-31 жовтня 2025 р. Одеса : Видавництво ОНТУ, 2025. 1316 с. *(Особистий внесок: формулювання підходу до інтеграції інтелектуальних навчальних програм і асистентів доведень; аналіз можливостей застосування формальної верифікації у навчанні квантових обчислень і програмування; підготовка тексту тез)*

9. Chukhray, A.; Yevdokymov, O.; Mandrikova, L.; Dehtiarova, T.; Havrylenko, O. Diagnostic Models and a Crypto-Based Sense-Economic Approach to Enhancing Motivation in Intelligent Mathematics Learning Systems. 2026. DOI: 10.1007/978-3-032-10477-9_11. (Scopus). *(Особистий внесок: участь у розробленні діагностичних моделей для інтелектуальних систем навчання математики, формуванні crypto-based sense-economic мотиваційного підходу, аналізі результатів та підготовці рукопису)*

10. Yevdokymov, O.; Chukhray, A.; Stoliarenko, T. Operationalizing the Formalizability of Mathematics Problems for Intelligent Tutoring Systems: Taxonomy, Measurement Protocol, and Educational Impact. Proceedings of the 5th International Workshop of IT-professionals on Artificial Intelligence, ProfIT AI 2025. Liverpool, United Kingdom, October 15-17, 2025. *(Особистий внесок: розроблення таксономії формалізованості математичних задач, формулювання протоколу вимірювання, обґрунтування освітнього впливу та підготовка основного тексту публікації)*

11. Yevdokymov, O.; Luchsheva, O. Development of a prototype intelligent web-based system for learning mathematics through an adaptive scenario. Aerospace Technic and Technology. No. 2, 2026. DOI: 10.32620/aktt.2026.2.06. *(Особистий внесок: розроблення архітектури та прототипу веборієнтованої*

інтелектуальної системи навчання математики, реалізація адаптивного сценарію, аналіз працездатності прототипу та підготовка рукопису.)

12. Yevdokymov, O.; Luchsheva, O. A Mathematical Model of Automatic Verification of Formalized Proofs and a Conservative Presentation Interface over Lean. Bulletin of V. N. Karazin Kharkiv National University, series «Mathematical modeling. Information technology. Automated control systems». 2026. Vol. 69. P. 33-40. DOI: 10.26565/2304-6201-2026-69-03. *(Особистий внесок: розроблення математичної моделі автоматичної перевірки формалізованих доведень, формування підходу до консервативного інтерфейсу подання над Lean, аналіз прикладів застосування та підготовка тексту статті.)*

Усі основні наукові положення, результати та висновки дисертаційної роботи отримано здобувачем самостійно; у спільних публікаціях здобувачеві належать результати, зазначені в описі особистого внеску.

Усі співавтори погодилися із задекларованим особистим внеском здобувача у спільних публікаціях.

ABSTRACT

*Oleksandr Yevdokymov***.** Models, Methods and Information Technology for Supporting Learning in the Exact Sciences – manuscript.

Dissertation submitted for the degree of Doctor of Philosophy (PhD) in specialty 122 “Computer Science”, field of knowledge 12 “Information Technologies”, at the National Aerospace University “Kharkiv Aviation Institute”, Kharkiv, 2026.

The dissertation is devoted to the development and scientific substantiation of a set of mathematical models, algorithmic methods, and information technology for supporting learning in the exact sciences. The proposed solutions provide formalized knowledge diagnosis, adaptive selection of learning problems, algorithmic generation of learning content, integration of formal verification of proofs and algorithms, and motivational regulation of learning activity.

The object of the research is the process of learning the exact sciences using computer-based and intelligent tutoring systems.

The subject of the research comprises mathematical models, algorithmic methods, and information technology for supporting learning in the exact sciences, which provide knowledge diagnosis, adaptive task selection, formal verification of theoretical statements and algorithms, generation of learning content, and software implementation of the corresponding mechanisms of learning support.

The dissertation presents a systematic analysis of the current state of intelligent tutoring systems, knowledge tracing approaches, adaptive outer and inner learning loops, digital platforms for supporting mathematics and related exact sciences, and the possibilities of formal verification in education. It is shown that existing digital platforms effectively support mass checking of answers and parametric variability of tasks; however, they insufficiently support diagnosing the causes of errors, working with proofs, deriving formulae, constructing algorithms, and controlling the logical rigor of reasoning.

The research substantiates an applied scientific contradiction between the need for scalable individualized tools for learning the exact sciences and the limitations of

existing systems that are mostly focused on checking final answers without sufficient support for theoretical reasoning, formal correctness, and adaptive diagnosis. To overcome this contradiction, an information technology is proposed that combines models of the learner's knowledge state, diagnostic models of errors, algorithmic problem generation, formally verified learning artifacts, and motivational reinforcement mechanisms.

A system model for supporting learning in the exact sciences has been developed. It includes verbal, structural, functional, mathematical, and machine levels of description. The structural model defines the main components of an intelligent tutoring system: the student interface, the student model, the diagnostic core, the task generator, the formal verification module, the motivational module, and the learning data storage. The functional model represents learning as a closed loop in which task selection or generation, solving, checking, diagnosis, knowledge state updating, motivational feedback, and selection of the next learning intervention form a single controlled process.

Learning problems in the exact sciences are formalized as objects that include a set of concepts and rules, possible problem statements, input data, admissible answers, and procedures for obtaining or verifying the result. The dissertation proposes a distinction between simple problems aimed at applying a known algorithm and complex problems related to constructing algorithms, deriving formulae, and proving statements. For complex problems, the concept of formalizability is introduced to determine what level of automatic or semi-automatic verification can be applied in a particular learning scenario.

A formalized diagnostic model for assessing the student's knowledge state is developed on the basis of a probabilistic representation of knowledge components using Bayesian approaches. The model is aimed not only at recording correct or incorrect answers, but also at identifying the likely causes of errors from indirect features of the answer, intermediate steps, or student behavior. This enables targeted learning interventions and provides a basis for adaptive selection of subsequent tasks.

An algorithmic model of parametric generation of learning problems with automatic checking of the correctness of input data and obtained results has been improved. The model makes it possible to generate mathematically correct variants of tasks, avoid degenerate cases, automatically determine reference answers, and use the results of generation within an adaptive learning loop. This approach is especially important for large-scale individualized learning, where each student should receive a sufficient number of diverse but semantically equivalent tasks.

The methods for integrating formal verification of mathematical proofs and algorithms into the learning process of the exact sciences have been further developed. The proposed approach uses Lean/Mathlib theorem-proving systems as a formally rigorous core, on the basis of which step-by-step learning scenarios, explanations, control questions, and mechanisms for checking the logical correctness of reasoning are formed. For algorithmic problems, formal verification is considered as a means of checking whether an algorithm satisfies its specification, invariants, and correctness requirements.

A model of motivational regulation of the learning process is developed. In this model, digital rewards are associated not with the mere fact of completing a task, but with meaningful learning events: independent correction of an error, demonstration of structural understanding, persistent work without excessive hints, a correct proof, or a constructive contribution to the learning environment. The dissertation substantiates the possibility of implementing this approach using blockchain, tokenized rewards, and smart contracts, provided that the motivational layer remains subordinate to pedagogical goals.

The dissertation also theoretically considers the possibility of using quantum algorithms to accelerate search in the space of parametric learning problems and scenarios. This consideration does not replace the main algorithmic solutions of the dissertation, but it defines a promising direction for the further development of the information technology under increasing dimensionality of the space of tasks, parameters, constraints, and learning trajectories.

The practical significance of the results lies in the fact that the main scientific provisions of the dissertation have been brought to the level of algorithmic solutions, software modules, and a prototype intelligent tutoring program named “Myrna”. The developed prototype provides generation of learning problems, checking of solutions, support for adaptive transitions between learning components, elements of knowledge diagnosis, presentation of formally verified learning material, and motivational regulation of the learning process.

The proposed models, methods, and software solutions can be used in higher education institutions to support the teaching and learning of mathematics and related exact disciplines, in distance and blended learning systems, in preparation for standardized exams in the exact sciences, and in the development of next-generation intelligent educational platforms. The research results have been tested in scientific publications, materials of international and national conferences, poster presentations, a research project report, and software prototypes.

Keywords: intelligent tutoring system, exact sciences, mathematics, STEM education, adaptive learning, diagnostic model, knowledge tracing, Bayesian modeling, formal verification, Lean, Mathlib, proof-checking, parametric problem generation, motivational regulation, information technology, prototype.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

Перелік укладено для дисертації «Моделі, методи та інформаційна технологія підтримки навчання точним наукам» з урахуванням термінів інтелектуальних навчальних систем, формальної верифікації, адаптивного навчання, математичної освіти та переліку публікацій здобувача.

AI	–	Artificial Intelligence; штучний інтелект
API	–	Application Programming Interface; програмний інтерфейс застосунку
AST	–	Abstract Syntax Tree; абстрактне синтаксичне дерево
BKT	–	Bayesian Knowledge Tracing; баєсівське відстеження знань
Coq / Rocq	–	інтерактивний асистент доведень і система формальної верифікації
DESSERT	–	International Conference on Dependable Systems, Services and Technologies
DKT	–	Deep Knowledge Tracing; глибоке відстеження знань
DOI	–	Digital Object Identifier; цифровий ідентифікатор об’єкта
EID	–	Scopus Electronic Identifier; електронний ідентифікатор запису в Scopus
HTML	–	HyperText Markup Language; мова гіпертекстової розмітки

ICTERI	–	International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications
ICMLBDA	–	International Conference on Machine Learning and Big Data Analytics
ISBN	–	International Standard Book Number; міжнародний стандартний номер книги
Isabelle/HOL	–	інтерактивний асистент доведень на основі логіки вищого порядку
ITS	–	Intelligent Tutoring System; інтелектуальна навчальна система
KT	–	Knowledge Tracing; відстеження знань
Lean	–	мова програмування та інтерактивний асистент доведень для формальної верифікації математичних тверджень
LLM	–	Large Language Model; велика мовна модель
LMS	–	Learning Management System; система управління навчанням
Mathlib	–	формалізована математична бібліотека екосистеми Lean
ML	–	Machine Learning; машинне навчання
PISA	–	Programme for International Student Assessment; міжнародне дослідження якості освіти

ProfIT AI	–	International Workshop of IT-professionals on Artificial Intelligence
Proof assistant	–	асистент доведень; програмне середовище для побудови й перевірки формальних доведень
Proof-checking	–	перевірка коректності формального або напівформального доведення
SQL	–	Structured Query Language; структурована мова запитів до баз даних.
SQLTOR	–	веборієнтована інтелектуальна навчальна система для навчання SQL
STEM	–	Science, Technology, Engineering and Mathematics; природничо-математична, інженерна й технологічна освіта
UI	–	User Interface; користувацький інтерфейс
URL	–	Uniform Resource Locator; уніфікований покажчик ресурсу
UX	–	User Experience; досвід користувача
Web	–	вебсередовище або вебтехнології, що використовуються для реалізації навчальних сервісів
АЗ	–	алгоритмічне завдання
Баєсівська модель	–	ймовірнісна модель оновлення стану знань на основі спостережуваних навчальних дій

БД	–	база даних
Блокчейн	–	розподілений реєстр даних, який може використовуватися для фіксації мотиваційних подій і цифрових винагород
Внутрішній цикл навчання	–	цикл підтримки розв’язання конкретної задачі: перевірка кроків, підказки, діагностика помилок
ДМ	–	діагностична модель
ЗВО	–	заклад вищої освіти
ЗНО	–	зовнішнє незалежне оцінювання
Зовнішній цикл навчання	–	цикл вибору наступної задачі або навчального впливу з урахуванням стану знань студента
ІКНП	–	інтелектуальна комп’ютерна навчальна програма
ІНС	–	інтелектуальна навчальна система
ІТ	–	інформаційні технології; інформаційна технологія залежно від контексту
КЗ	–	компонент знань
КЗУ	–	компонент знань та умінь
НДР	–	науково-дослідна робота
НМТ	–	національний мультипредметний тест
ПЗ	–	програмне забезпечення

Смартконтракт	–	програмний механізм автоматичного виконання правил у блокчейн-орієнтованому середовищі
СУБД	–	система управління базами даних
Токен	–	цифрова одиниця винагороди або фіксації навчального досягнення в мотиваційному модулі
УЦОЯО	–	Український центр оцінювання якості освіти
Формалізованість задачі	–	ступінь можливості подання навчальної задачі у формі, придатній для алгоритмічної або формальної перевірки
Формальна верифікація	–	математично строга перевірка коректності доведення, твердження або алгоритму засобами формальної логіки
Цифрова винагорода	–	елемент мотиваційного регулювання, що нараховується за значущу навчальну подію
ШІ	–	штучний інтелект

ВСТУП

Актуальність теми. Точні науки, насамперед математика, виконують роль «мови» сучасної науки й інженерії, забезпечують формування алгоритмічного мислення та слугують основою для моделювання складних систем. Тому рівень компетентностей у галузі точних наук є системоутворювальним чинником для підготовки фахівців у STEM, для розвитку технологічних секторів економіки та для здатності суспільства інтегрувати інновації.

Разом із тим, у національних і міжнародних освітніх вимірюваннях фіксується стійка негативна динаміка щодо математичної підготовки: зростає частка здобувачів, які не досягають базового рівня, і зменшується частка тих, хто демонструє високі результати. Для України це означає додаткові ризики в умовах обмежених ресурсів, коли якість підготовки кадрів у точних дисциплінах прямо впливає на технологічну стійкість держави. У глобальному вимірі аналогічні тенденції підтверджують, що проблема має не локальний, а системний характер.

Окремим проявом цієї проблеми є **примітивізація математичної освіти**, яка в науковому сенсі виражається не «спрощенням», а зміною структури навчальної діяльності: зростає частка завдань, де достатньо відтворити шаблон або виконати обчислення за відомою формулою, та зменшується частка навчальних дій, пов’язаних із побудовою доказу, виведенням формул, конструюванням алгоритму, аналізом умов застосовності методу. Унаслідок цього знання стають фрагментарними: студент може відтворити окремі процедури, але не володіє цілісним апаратом понять і не здатен переносити знання на нетипові задачі.

Сучасні цифрові платформи підтримки навчання частково знімають навантаження з викладача, зокрема в частині масової перевірки та створення варіативних завдань. Проте переважна більшість таких рішень ефективна насамперед для задач, де відповідь зводиться до числа, формули або вибору з варіантів, і значно слабше підтримує теоретичні компоненти навчання – логічні міркування, доведення, виведення та побудову алгоритмів.

У дослідницькій традиції інтелектуальних навчальних систем (Intelligent Tutoring Systems, ITS) сформовано підхід, у якому навчальний процес розглядається як кероване набуття компетентностей, а система виконує функції діагностики та індивідуального навчального впливу. Результати робіт окремих наукових шкіл у галузі ITS демонструють, що адаптивність може бути реалізована через «внутрішній цикл» – покрокову підтримку розв'язання – та «зовнішній цикл» – вибір наступного завдання з урахуванням поточного стану здобувача. Разом з тим, практичні платформи комп'ютерного оцінювання, що широко застосовуються у викладанні математики, переважно зосереджуються на перевірці результату, і лише обмежено – на діагностиці структури помилки та поясненні її причини.

Сучасні методи машинного навчання, зокрема моделі відстеження знань (knowledge tracing), дають можливість будувати ймовірнісне уявлення про те, чи засвоїв студент певний навчальний компонент. Однак для навчання точним наукам критичним є питання **формальної коректності**: статистично правдоподібна відповідь не тотожна логічно правильному міркуванню. У складних темах (лінійна алгебра, аналіз, дискретна математика) ключовим стає не лише «правильний результат», а правильний шлях отримання результату, включно з використанням означень, лем і теорем у коректному порядку.

Перспективним інструментом подолання цієї межі є **системи формальної перевірки доказів** (Lean, Coq/Rocq, Isabelle/HOL). Вони дозволяють перевіряти правильність доказу по кроках у межах формальної логіки, забезпечуючи рівень строгості, який принципово недоступний статистичним моделям. Водночас масове використання таких систем в освіті стримується високим порогом входу, відсутністю звичного математичного користувацького інтерфейсу та складністю інтеграції в навчальні сценарії.

Паралельно розвиваються генеративні мовні моделі (LLM), здатні створювати варіативні формулювання умов задач і пояснювальні тексти. Проте такі моделі не гарантують алгоритмічної точності та формальної коректності,

тому в навчанні точним наукам вони мають розглядатися як допоміжний інструмент, який потребує поєднання з детермінованими алгоритмами та/або формальними механізмами верифікації.

Отже, у сфері навчання точним наукам формується **науково-прикладна суперечність**: потрібні масштабовані цифрові засоби, здатні: індивідуалізувати навчання; діагностувати знання; підтримувати теоретичне мислення, натомість масові платформи переважно контролюють кінцеву відповідь, а засоби формальної перевірки доказів ще не мають достатньо доступної інтеграції в освітній процес і зручного інтерфейсу для студентів та викладачів.

Прогалина → мотивація → задача в даному дослідженні формулюється так: Існує недостатність моделей і методів, які одночасно забезпечують: а) формалізовану діагностику засвоєння навчальних компонентів; б) адаптивний вибір задач; в) алгоритмічну генерацію унікальних варіантів задач із контролем коректності; г) підтримку навчання складним задачам на доведення / виведення через інтеграцію proof-checking.

Отже, необхідно розробити **комплекс моделей, методів і інформаційної технології** підтримки навчання точним наукам.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана відповідно до наукового напряму кафедри математичного моделювання та штучного інтелекту Національного аерокосмічного університету «Харківський авіаційний інститут» та узгоджується з тематикою досліджень у сфері інтелектуальних інформаційних систем, цифрових освітніх технологій і комп’ютерної підтримки навчання. Напрям роботи відповідає актуальним потребам розвитку STEM-освіти та створення сучасних освітніх ІТ-рішень для України.

Мета і завдання дослідження. Метою роботи є усунення розриву між потребою в індивідуалізованому, діагностично орієнтованому та теоретично спрямованому навчанні точним наукам і обмеженими можливостями масових

платформ, щоб підвищити якість, адаптивність і надійність навчального процесу. Для досягнення поставленої мети необхідно вирішити такі завдання:

1) Виконати аналіз сучасного стану ITS, підходів knowledge tracing, адаптивних циклів навчання та можливостей proof-checking у навчанні точним наукам;

2) Формалізувати навчальні задачі точних наук та класифікувати їх на прості задачі, орієнтовані на застосування алгоритмів, і складні задачі, пов’язані з побудовою алгоритмів, виведенням формул і доведенням теорем.

3) Розробити формалізовану діагностичну модель оцінювання стану знань студента на основі ймовірнісного представлення компонентів знань із використанням баєсівських підходів.

4) Розробити метод інтеграції формальної верифікації математичних доведень у навчальний процес точних наук на основі систем доведень Lean/Mathlib.

5) Розробити метод інтеграції формальної верифікації алгоритмів у навчання побудові алгоритмічних рішень і застосуванню алгоритмів у прикладних та інженерних задачах.

6) Розробити алгоритмічну модель параметричної генерації навчальних задач із автоматичною перевіркою коректності вхідних даних та отриманих результатів.

7) Розробити модель мотиваційного регулювання навчального процесу на основі системи цифрових винагород із можливістю реалізації засобами смартконтрактів.

8) Розробити архітектуру та прототип інтелектуальної навчальної програми, що реалізує запропоновані моделі та методи.

9) Теоретично обґрунтувати можливість використання квантових алгоритмів для прискорення пошуку в просторі параметричних навчальних задач і сценаріїв.

10) Оцінити працездатність запропонованих рішень на рівні прототипу та визначити напрями їх подальшого розвитку.

Об’єктом дослідження є процес навчання точним наукам із використанням комп’ютерних та інтелектуальних навчальних систем.

Предметом дослідження є математичні моделі, алгоритмічні методи, інформаційна технологія та системи формальної верифікації підтримки навчання точним наукам.

Методи дослідження. Для розв’язання поставлених завдань використано:

1) Методи систематизованого огляду та порівняльного аналізу, для узагальнення сучасного стану досліджень і практичних платформ у сфері підтримки навчання точним наукам;

2) Методи математичного моделювання та теоретико-множинного представлення, для формалізації навчальних задач, стану знань та взаємозв’язків між компонентами навчального процесу;

3) Методи теорії ймовірностей і Баєсівського моделювання, для побудови діагностичних моделей засвоєння компонентів знань;

4) Методи теорії алгоритмів і програмної інженерії, для розроблення алгоритмічної моделі генерації задач і перевірки коректності відповідей;

5) Методи формальної логіки, теорії типів і формальної верифікації, для аналізу математичних тверджень і алгоритмів у середовищах Lean/Mathlib;

6) Методи архітектурного, об’єктно-орієнтованого та компонентно-орієнтованого проектування, для побудови інформаційної технології та програмного прототипу;

7) Методи теоретичного аналізу складності, для обґрунтування можливості квантового прискорення пошуку в просторі параметричних задач.

Наукова новизна одержаних результатів. У дисертаційній роботі **вперше**:

– **розроблено** *модель архітектури* інтелектуальної навчальної системи для індивідуалізованого навчання точним наукам та алгоритмічних дисциплін, яка

інтегрує *модель предметної області*, *модель студента* з байєсівським оновленням стану знань, параметричну генерацію задач, діагностичні моделі, формальну верифікацію, LLM-сервіси, мотиваційні механізми та фіксацію навчальних дій студента;

– **удосконалено** *метод* параметричної генерації навчальних задач із автоматичною перевіркою коректності вхідних даних, умов задачі та еталонного результату, що забезпечує формування математично коректних варіантів задач і автоматизовану діагностику відповідей студента;

– **удосконалено** *метод* інтеграції формальної верифікації математичних доведень і алгоритмічних рішень у навчальний процес на основі Lean/Mathlib, що забезпечує покрокову перевірку логічної коректності міркувань, специфікацій і властивостей алгоритмів.

– **дістала подальшого розвитку** *інформаційна технологія* підтримки навчання точним наукам, яка забезпечує інтеграцію моделей предметної області та студента, алгоритмічної генерації навчальних задач, формальної перевірки математичних і алгоритмічних розв’язків, адаптивного керування навчальною траєкторією та засобів мотиваційного впливу.

Практичне значення одержаних результатів полягає в тому, що основні наукові положення дисертаційної роботи доведено до рівня алгоритмічних рішень, програмних модулів і прототипу інтелектуальної навчальної програми «Мирна». Розроблений прототип забезпечує генерацію навчальних задач, перевірку правильності розв’язків, підтримку адаптивної логіки переходів між навчальними компонентами та реалізацію механізмів подання формально верифікованого навчального матеріалу. Запропоновані моделі та методи можуть бути використані у закладах вищої освіти для підтримки навчання точних дисциплін, у системах дистанційного та змішаного навчання, а також при розробленні інтелектуальних освітніх платформ.

Ефективність запропонованих рішень підтверджена їх практичною реалізацією у вигляді програмного прототипу, що дозволяє застосовувати

результати роботи у реальному навчальному процесі, що підтверджує можливість використання розробленої інформаційної технології як основи для побудови інтелектуальних навчальних систем нового покоління.

Результати роботи впроваджено (додаток А):

– в освітній процес кафедри інженерії програмного забезпечення Національного аерокосмічного університету «Харківський авіаційний інститут», м. Харків, (акт впровадження №1);

– у навчальний процес Приватного закладу (організації, установи) Приватної Гімназії "Еммануїл", м. Мукачево, (акт впровадження №2 від 02.04.2026);

– у навчальний процес **Аерокосмічного ліцею на базі Національного аерокосмічного університету «ХАІ»**, м. Харків, (акт впровадження №3 від 20.05.2026);

Апробація результатів дисертації. Основні положення та результати дисертаційної роботи доповідалися й обговорювалися на міжнародних і всеукраїнських наукових конференціях, зокрема на конференціях, індексованих у Scopus, а також на наукових семінарах відповідних підрозділів університету. зокрема: «Dependable System, Services and Technologies Conference (Athens, Greece 2022); XVI Міжнародна науково-практична конференція «СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ ТА ОСВІТІ» (Дніпро, 2022); Digital Theme UK-Ukraine Research Twinning Conference, 27 March - 30 March 2023; І Науково-практична конференція «Сучасні технологічні рішення для освіти, науки, бізнесу та міського господарства» (Київ – Харків, 25 листопада 2025); XVIII міжнародна науково-практична конференція, (Одеса, 30-31 жовтня 2025 р); 20th International Conference, ICTERI 2025, (Nice, France, September 1–4, 2025); 5th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2025) 2025, (Liverpool, United Kingdom, October 15-17, 2025);

а також на 5th International Conference on Machine Learning & Big Data Analytics (ICMLBDA 2025) Organized by ICS Global, Scopus, MES College, Marampally, Kerala, India 7-8 NOVEMBER 2025 (Hybrid Mode), де було отримано нагороду за найкращий матеріал в двох треках «Best Paper Award under the “Next Gen Analytics” and “Data-Driven Models and Analytics for Circular Economy and Sustainable Innovation” tracks».

Публікації. За темою дисертаційної роботи опубліковано 12 наукових праць: 4 статей у наукових фахових виданнях України, з них одна у виданні, внесеному до міжнародних наукометричних баз даних, 7 – у матеріалах наукових конференцій, з них 3 у збірниках матеріалів, внесених до міжнародних наукометричних баз даних.

Усі основні наукові положення, результати, висновки й рекомендації дисертаційної роботи отримано автором самостійно.

РОЗДІЛ 1 СУЧАСНИЙ СТАН ПРОБЛЕМИ ПІДТРИМКИ НАВЧАННЯ ТОЧНИМ НАУКАМ

Проблематика підтримки навчання точним наукам перебуває на перетині комп’ютерних наук, педагогічної інженерії, математичного моделювання та теорії інтелектуальних систем. У цьому розділі розглянуто сучасний стан досліджень і практичних рішень, що стосуються навчання математики та суміжних точних дисциплін, аналізується негативна динаміка рівня математичних компетентностей, а також обґрунтовується необхідність створення інформаційної технології, здатної поєднувати адаптивність, діагностику, алгоритмічну генерацію задач і формальну верифікацію доведень та алгоритмів.

Огляд побудовано з урахуванням двох груп джерел. Перша група охоплює міжнародні та національні аналітичні матеріали, у тому числі PISA, звіти УЦОЯО щодо НМТ/ЗНО, а також публікації щодо якості STEM-освіти [3–6, 10, 62, 78, 81–83]. Друга група включає наукові праці, присвячені інтелектуальним навчальним системам, діагностичним моделям, адаптивним циклам навчання, формальній верифікації та генеративному штучному інтелекту [14, 26, 28–29, 64, 69–70, 73, 77, 99, 101, 111, 113, 116].

1.1 Роль точних наук у підготовці фахівців STEM-сфери

Точні науки виконують системоутворювальну роль у підготовці фахівців STEM-сфери, оскільки забезпечують формування абстрактного мислення, навичок моделювання, уміння працювати з формальними структурами та здатності обґрунтовувати інженерні рішення [62]. У сучасній інженерії та інформаційних технологіях математичні знання вже не обмежуються обчислювальною технікою у вузькому сенсі: вони становлять основу аналізу даних, машинного навчання, моделювання складних систем, кібербезпеки, автоматизації, теорії керування, робототехніки та відмовостійких систем.

Рисунок 1.1 – Узагальнена схема проблемного поля підтримки навчання точним наукам: негативні тенденції, інтелектуальні системи, діагностика, генерація задач, формальна верифікація, генеративний ШІ

У роботах, присвячених ролі математики у STEM-активностях, підкреслюється, що математика виступає не лише інструментом розрахунку, а й мовою опису закономірностей і побудови моделей [62]. Для фахівця з інформаційних технологій математична компетентність є підґрунтям для засвоєння алгоритмів, структур даних, теорії складності, дискретної математики, баз даних, статистики, оптимізації та формальних методів. Саме тому зниження рівня математичної підготовки має наслідки не лише для успішності окремих студентів, а й для технологічної спроможності країни.

В українському контексті значення точних наук додатково посилюється потребою підготовки фахівців для оборонної, авіаційної, космічної, кібербезпекової та програмно-інженерної сфер. Для таких напрямів недостатньо володіти лише прикладним інструментарієм; необхідно розуміти причинно-наслідкові зв’язки, уміти оцінювати коректність рішень і переносити знання на нові задачі. Відповідно, система освіти має забезпечувати не тільки засвоєння процедур, а й формування здатності до доведення, аналізу обмежень, побудови алгоритмів і перевірки їх властивостей.

Проблема полягає в тому, що традиційні форми навчання точним наукам часто орієнтовані на масове пояснення і контроль кінцевого результату. Вони добре працюють у ситуаціях, де всі студенти мають приблизно однаковий темп засвоєння і можуть регулярно отримувати зворотний зв’язок від викладача. Проте в умовах зростання чисельності груп, змішаного та дистанційного навчання, різного початкового рівня підготовки і збільшення обсягу інформації така модель стає недостатньою [24].

З позицій інженерії навчальних систем підтримка навчання точним наукам повинна розглядатися як окрема інформаційна технологія, а не як набір розрізнених електронних матеріалів. Така технологія має враховувати структуру знань, тип задачі, історію діяльності студента, характер помилок і можливість автоматизованої перевірки. Саме ця постановка відрізняє інтелектуальні навчальні системи від звичайних електронних курсів або банків тестів [101, 111].

Таблиця 1.1 – Значення точних наук для підготовки STEM-фахівців

Аспект підготовки	Роль точних наук	Вимоги до навчальної підтримки
Формування логічного мислення	Розвиток уміння будувати послідовні міркування, працювати з означеннями, правилами і доведеннями.	Потрібна підтримка поетапного розв’язання та перевірки логіки міркування.
Алгоритмічне мислення	Здатність формулювати алгоритми, аналізувати їх коректність і обмеження.	Необхідні задачі на побудову алгоритмів, а не лише на їх застосування.
Інженерне моделювання	Побудова моделей об’єктів, процесів і систем із формальними залежностями.	Потрібна інтеграція математичних моделей із програмною реалізацією.
Підготовка до критичних систем	Вимога коректності, надійності та верифікованості рішень.	Доцільне використання proof-checking та формальних методів.
Адаптація до цифрових технологій	Здатність використовувати математичні інструменти в ІТ, ШІ, аналізі даних та автоматизації.	Потрібна персоналізована діагностика і добір задач.

Рисунок 1.2. – Схема ролі точних наук у підготовці STEM-фахівця: логіка, алгоритми, моделювання, формальна коректність, інженерні застосування

1.2 Негативні тенденції у рівні знань з математики та інших точних дисциплін

Актуальність створення нових засобів підтримки навчання точним наукам підтверджується не лише внутрішніми освітніми спостереженнями, а й результатами міжнародних і національних вимірювань. За результатами PISA 2022 середній показник з математики в країнах OECD суттєво знизився порівняно з попереднім циклом; у глобальному звіті OECD це подано як один із найпомітніших спадів за історію дослідження [82]. Українські регіони, які брали участь у PISA 2022, також демонструють значні виклики: за даними країнової нотатки OECD, частка учнів, що досягли щонайменше другого рівня математичної грамотності, була нижчою за середній показник OECD [83].

Особливість PISA полягає в тому, що воно оцінює не стільки знання шкільної програми, скільки здатність застосовувати математичні знання до життєвих і проблемних ситуацій [3]. Тому низькі або нестійкі результати у PISA свідчать не лише про труднощі з обчисленнями, а й про проблеми з математичним моделюванням, інтерпретацією умов, вибором стратегії, роботою з нетиповими задачами та перенесенням знань.

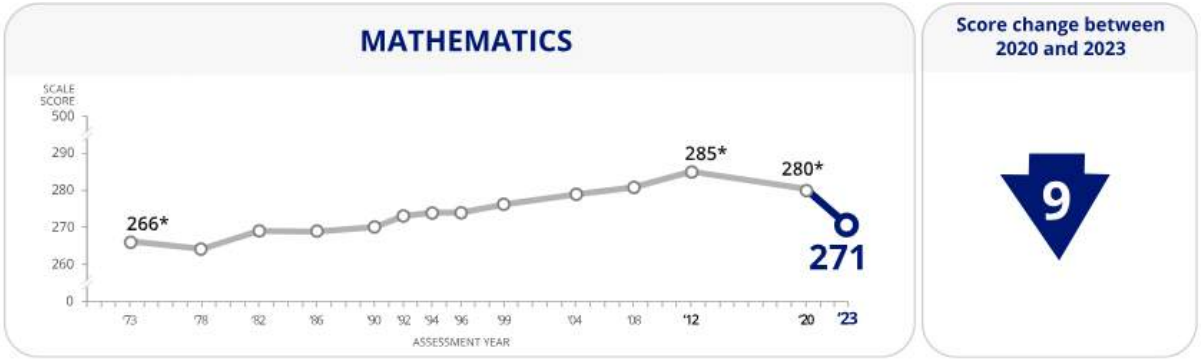


Рисунок 1.3 – Тренд середніх балів з математики NAEP для 13-річних учнів.

Національний контекст відображається у звітах Українського центру оцінювання якості освіти щодо НМТ. УЦОЯО публікує офіційні звіти про проведення НМТ, що містять статистичні й аналітичні матеріали щодо розподілу результатів за предметами [4–6]. При цьому сам УЦОЯО наголошує на обмеженнях інтерпретації даних НМТ: ці результати використовуються як вступне випробування і не є прямою державною підсумковою атестацією всієї сукупності випускників [4–6]. Однак масовість цих даних робить їх цінним індикатором труднощів, з якими стикаються вступники під час виконання математичних завдань.

У власному дослідженні можливостей використання ITS для курсів вищої математики в ЗВО було узагальнено тенденції щодо зниження математичної підготовки та показано, що одна з причин полягає в обмеженості традиційних засобів навчання щодо індивідуальної діагностики і своєчасного зворотного

зв’язку [24]. У статті також запропоновано розрізняти прості задачі на застосування відомого алгоритму та складні задачі на побудову алгоритму, доведення або виведення формул, що прямо пов’язано з характером сучасної кризи математичної освіти [24].

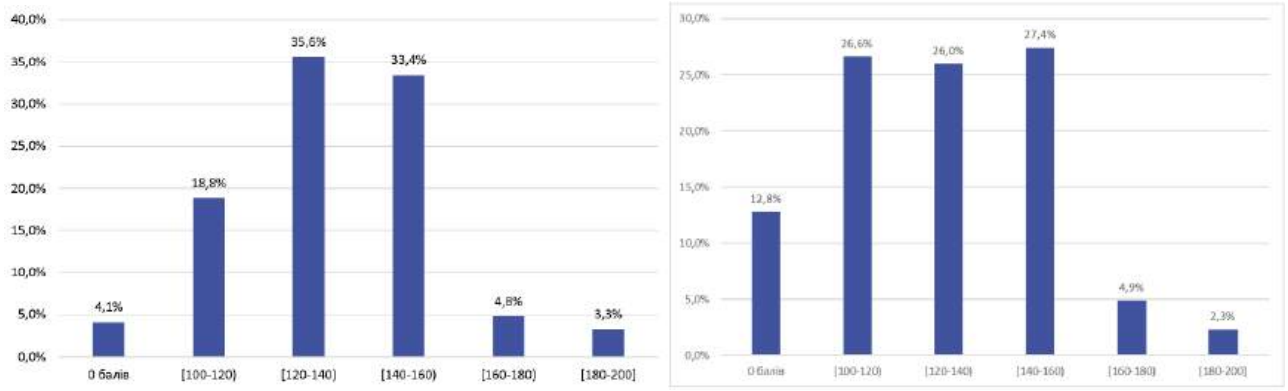


Рисунок 1.4 – Розподіл результатів учасників НМТ
з математики за 2023/2024 рік

Негативні тенденції не слід пояснювати лише зовнішніми факторами. Вони мають структурний характер: навчальний процес часто залишається орієнтованим на контроль кінцевої відповіді, а не на розуміння процесу її отримання. Унаслідок цього студент може продемонструвати формальну правильність окремих процедур, але не вміти пояснити, чому обраний метод застосовний, які умови його коректності, де саме виникла помилка у випадку неправильного розв’язання.

Таблиця 1.2 – Джерела даних щодо проблематики математичної підготовки

Джерело	Що відображає	Значення для дисертації
PISA 2022 [3, 82, 83]	Математичну грамотність і здатність застосовувати знання у проблемних ситуаціях.	Підтверджує міжнародний характер проблеми і важливість задач на застосування та міркування.

Звіти УЦОЯО щодо НМТ [4–6]	Масові результати вступних випробувань із математики та інших предметів.	Дає емпіричне підґрунтя для аналізу труднощів українських вступників.
Дослідження ITS у вищій математиці [24]	Аналіз можливостей інтелектуальних систем у курсах вищої математики.	Показує потребу в адаптивності, генерації задач і глибшій діагностиці.
Концепція ITS для підготовки до НМТ [64]	Архітектура діагностично орієнтованої системи підготовки до STEM-іспиту.	Обґрунтовує необхідність внутрішнього і зовнішнього циклів навчання.

Рисунок 1.5 – Діаграма негативних тенденцій у математичній підготовці за даними PISA та НМТ/ЗНО з поясненням обмежень інтерпретації таких даних

1.3 Примітивізація математичної освіти як системна проблема

Під примітивізацією математичної освіти у межах цієї роботи розуміється не зниження формальної складності навчальної програми, а зсув центру навчальної діяльності від побудови міркувань до відтворення шаблонних процедур. Такий зсув проявляється у переважанні задач, де достатньо підставити числа у формулу або повторити відомий алгоритм, і в недостатній кількості задач, що потребують доведення, виведення, побудови алгоритму, аналізу умов застосування методу або перевірки коректності рішення.

Алгоритмізація навчання у класичному розумінні не означає механічного зведення всього навчання до шаблонів. Навпаки, у працях Л. Н. Ланди та А. Г. Чухрая алгоритмічний підхід пов’язаний із чітким описом діяльності, виділенням кроків, умов застосування, контрольних точок і способів формування умінь [2, 7]. Примітивізація виникає тоді, коли алгоритм подається як готовий рецепт без розкриття підстав, інваріантів, обмежень і логіки вибору наступного кроку.

У навчанні точним наукам особливо небезпечною є ситуація, коли студент засвоює «ознаки задачі», але не засвоює зміст методу. Наприклад, формальний вигляд рівняння може підказувати застосування певної формули, але студент не перевіряє область допустимих значень, умови невиродженості, властивості функції або необхідність доведення. Такий підхід призводить до нестійких знань: за мінімальної зміни умови задача перестає розпізнаватися, а перенесення знань не відбувається.

Психолого-педагогічні дослідження когнітивного навантаження, мультимедійного навчання і самопояснення показують, що ефективне навчання потребує керованого балансу між складністю матеріалу, структурою подання і активною діяльністю студента [22, 74, 88, 106]. Якщо система підтримки навчання лише збільшує кількість однотипних вправ, вона може підсилювати процедурну вправність, але не обов’язково формує концептуальне розуміння. Тому сучасна інформаційна технологія повинна підтримувати не тільки

практику, а й пояснення, діагностику причин помилок і роботу зі складними задачами.

Для подолання примітивізації необхідно розділити навчальні задачі за характером діяльності. Простим задачам відповідають алгоритмічна генерація, перевірка результату і параметричне варіювання. Складним задачам потрібні інші засоби: поетапне доведення, побудова алгоритму, перевірка інваріантів, формальна верифікація, аналіз структури міркування. Саме таке розмежування було покладене в основу подальшого системного моделювання в розділі 2 і методів, розроблених у розділі 3.

Таблиця 1.3 – Ознаки примітивізації та засоби її подолання

Ознака проблеми	Наслідок	Засіб подолання
Переважання шаблонних задач	Формальна вправність без перенесення знань.	Введення складних задач на доведення, виведення і побудову алгоритмів.
Перевірка лише кінцевої відповіді	Неможливість локалізувати причину помилки.	Діагностичні моделі за непрямими ознаками та аналіз структури відповіді.
Відсутність пояснення умов застосування методу	Помилкове використання правил у нетипових ситуаціях.	Формалізація умов, предикатів коректності та контрольних точок.
Мотивація лише через оцінку	Зниження внутрішньої зацікавленості.	Мотиваційні механізми, пов’язані з прогресом і смисловою цінністю навчального результату.

Рисунок 1.6 – Схема «процедурне навчання → концептуальне навчання»: від шаблонної відповіді до доведення, виведення та побудови алгоритму

1.4 Інтелектуальні навчальні системи та адаптивні цикли навчання

Інтелектуальні навчальні системи (Intelligent Tutoring Systems, ITS) виникли як спроба наблизити комп'ютерне навчання до індивідуального викладання. На відміну від звичайних електронних курсів, ITS містять модель предметної області, модель студента, модель навчального впливу та механізми адаптації [77, 111, 113]. У таких системах навчання розглядається як керований процес, у якому кожна дія студента може змінювати внутрішню модель його знань і впливати на вибір наступного кроку.

Загальне порівняння поведінки систем викладання показує, що ефективність ITS значною мірою залежить від того, наскільки точно система може оцінити стан студента і сформувати корисний наступний навчальний вплив [23, 113]. Для точних наук це особливо важливо, оскільки помилка може бути не лише неправильною відповіддю, а й порушенням логіки доведення, некоректним

застосуванням теореми, неправильним вибором алгоритму або помилкою у формальному перетворенні.

У праці, присвяченій концепції інтелектуальної системи підготовки до українського фінального STEM-іспиту, запропоновано діагностично орієнтовану архітектуру, що спирається на принцип раціонального керування за діагнозом і передбачає наявність багатьох діагностичних моделей [64]. Такий підхід важливий для цієї дисертації, оскільки він зміщує акцент з перевірки факту помилки на встановлення її причин, що є основою для подальшої адаптації навчального процесу.

Для опису адаптивності ITS часто використовують розмежування зовнішнього і внутрішнього циклів навчання. Зовнішній цикл відповідає за вибір наступної задачі, теми або сценарію, тоді як внутрішній цикл забезпечує підтримку студента в межах конкретної задачі: підказки, аналіз проміжних кроків, перевірку часткових відповідей і діагностику помилок [64, 89–90, 113].

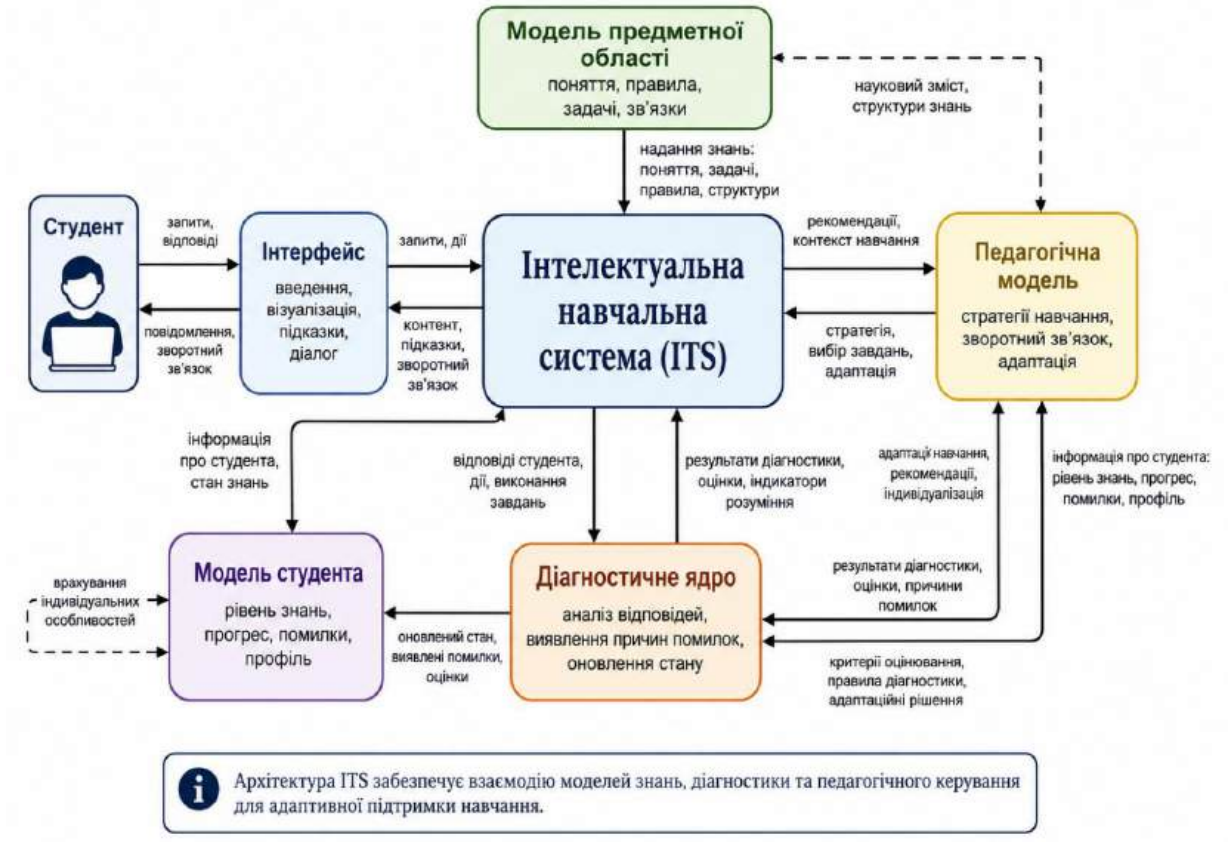


Рисунок 1.7 – Узагальнена архітектура ITS: модель предметної області, модель студента, педагогічна модель, інтерфейс, діагностичне ядро

1.4.1 Зовнішній цикл навчання

Зовнішній цикл навчання визначає, яку задачу або навчальний вплив студент отримає далі. У традиційних електронних курсах цей цикл часто задається фіксованою послідовністю тем або вправ. В інтелектуальній системі зовнішній цикл повинен враховувати стан знань, історію помилок, складність задачі, навчальні цілі, мотиваційний стан і доступність необхідних пояснень.

У межах точних наук зовнішній цикл має вирішувати не лише задачу «наступна тема», а й задачу вибору такого навчального об’єкта, який активує потрібні компоненти знань. Якщо студент робить помилку у правилі перетворення рівнянь, система не повинна випадково переходити до нової теми; вона має запропонувати задачу, що сприяє усуненню виявленої причини помилки. Саме тому зовнішній цикл пов’язаний із діагностичними моделями і баєсівським оновленням стану знань [25, 27, 68, 91–92].

У системі підготовки до НМТ, описаній у [64], зовнішній цикл повинен забезпечувати рух студента між задачами різних типів, складності та навчальної цінності. Для дисертаційної роботи це означає, що зовнішній цикл має бути реалізований не як простий «рандомізатор задач», а як метод адаптивного добору задач на основі стану знань і діагностичного висновку.

1.4.2 Внутрішній цикл навчання

Внутрішній цикл навчання описує взаємодію студента з конкретною задачею. Він включає подання умови, приймання відповіді або проміжного кроку, перевірку, підказку, уточнення, повторну спробу і формування діагностичного сигналу. Для простих задач внутрішній цикл може обмежуватися перевіркою відповіді і коротким поясненням. Для складних задач на доведення або побудову алгоритму внутрішній цикл повинен підтримувати аналіз структури міркування.

Праці з розроблення внутрішнього циклу комп’ютерних тьюторів підкреслюють значення покрокової підтримки, зворотного зв’язку і адаптації до конкретної помилки студента [89–90]. У точних науках це набуває особливої

ваги, оскільки проміжний крок часто не є просто “частиною відповіді”, а вказує на певний спосіб мислення. Наприклад, неправильно обраний наступний крок у доведенні може свідчити про нерозуміння логічної імплікації, а помилка в інваріанті алгоритму – про нерозуміння умов коректності.

Отже, ефективна ITS для точних наук повинна підтримувати обидва цикли: зовнішній цикл добору задач і внутрішній цикл аналізу розв’язання. Їхнє поєднання формує основу для розроблення інформаційної технології, що описується у наступних розділах цієї дисертації.

Рисунок 1.8 – Схема зовнішнього та внутрішнього циклів навчання: вибір задачі, розв’язання, підказка, діагностика, оновлення моделі студента

1.5 Методи відстеження та діагностики знань

Методи відстеження знань (knowledge tracing) спрямовані на оцінювання того, наскільки студент засвоїв окремі компоненти предметної області. Класичною моделлю є Bayesian Knowledge Tracing (BKT), що оцінює ймовірність засвоєння навички на основі спостережень про правильні та

неправильні відповіді [27]. Подальші роботи розширили цей підхід логістичними моделями, глибокими нейронними мережами та іншими способами побудови моделей студента [30, 68, 91–92].

У контексті точних наук важливо відрізняти відстеження знань від діагностики. Knowledge tracing відповідає на питання: «яка ймовірність того, що студент засвоїв компонент знань?». Діагностична модель відповідає на інше питання: «яка прихована причина конкретної помилкової відповіді або неправильного кроку?”. Це розрізнення є принциповим для дисертаційної роботи, оскільки система підтримки навчання повинна не лише зменшувати або збільшувати оцінку рівня знань, а й встановлювати причинний зв’язок між помилкою і порушеним компонентом.

Діагностичні моделі інтелектуальних навчальних систем, розроблені у працях А. С. Куліка, А. Г. Чухрая та співавторів, орієнтовані на виявлення причин помилок під час розв’язування алгебраїчних задач і формування коригувального впливу [63]. У подальших роботах розвивалися підходи до внутрішнього циклу, адаптації та педагогічної підтримки [89–90]. У цій дисертації ці ідеї узагальнюються для ширшого класу задач точних наук, зокрема задач на доведення, виведення формул та побудову алгоритмів.

Окремий напрям становить аналіз структури математичного виразу. Використання дерева виразу для адаптивного навчання дозволяє порівнювати студентську відповідь з еталоном не лише за кінцевим значенням, а й за структурою [23]. Такий підхід наближає систему до змістовної діагностики: вона може виявити перше розходження, тип помилки, локалізувати порушений компонент знань і сформувати більш точний навчальний вплив.

Таблиця 1.4 – Порівняння відстеження знань і діагностичних моделей

Характеристика	Knowledge tracing	Діагностична модель
Основне питання	Який рівень засвоєння компонента знань?	Яка причина конкретної помилки?

Тип даних	Послідовність правильних/неправильних відповідей, траєкторія дій.	Непрямі ознаки помилки: форма відповіді, структура виразу, крок розв’язання.
Результат	Ймовірність засвоєння або оцінка стану студента.	Гіпотеза про першопричину помилки та пов’язані компоненти знань.
Роль у системі	Оновлення моделі студента і вибір наступної задачі.	Пояснення помилки, корекція траєкторії, локалізований навчальний вплив.

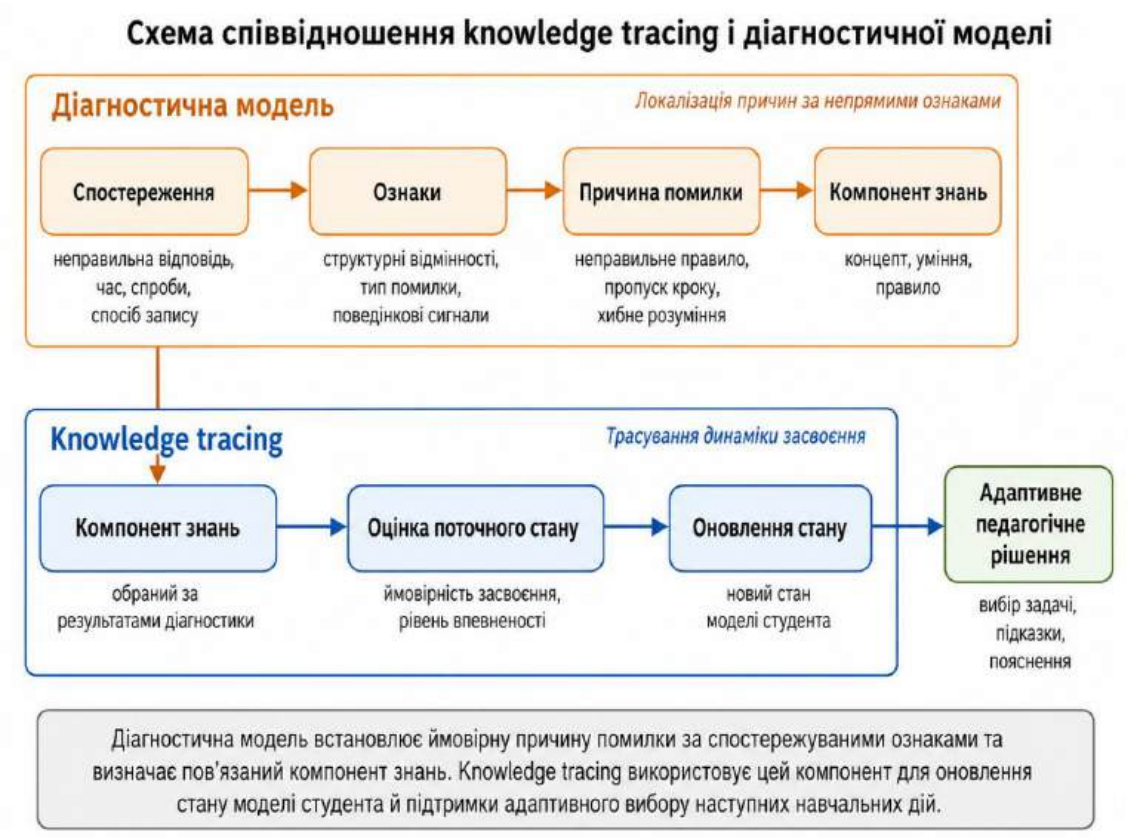


Рисунок 1.9 – Схема співвідношення knowledge tracing і діагностичної моделі: спостереження → ознаки → причина → компонент знань → оновлення стану.

1.6 Аналіз сучасних цифрових платформ підтримки навчання точним наукам

Сучасний ринок цифрових засобів навчання математики охоплює широкий спектр платформ: від банків задач і систем автоматизованої перевірки до адаптивних курсів, інтерактивних геометричних середовищ і AI-асистентів. У статті щодо концепції ITS для підготовки до українського фінального STEM-іспиту наведено огляд платформ OnlineMSchool, IXL Learning, Khan Academy / Khanmigo, LearnBop, Mangahigh, GeoGebra, CueThink, WeBWorK, IMathAS та інших рішень [64].

Ці платформи вирішують різні частини проблеми. Наприклад, WeBWorK і IMathAS підтримують алгоритмічну генерацію та автоматизовану перевірку математичних задач; GeoGebra забезпечує візуалізацію і маніпулювання математичними об’єктами; Khan Academy пропонує навчальні відео, вправи і відстеження прогресу, а Khanmigo використовує AI-орієнтований підхід до тьюторської підтримки [41, 52, 60, 64, 98].

Разом із тим, жоден із розглянутих класів платформ повністю не закриває потребу, що сформульована в цій дисертації. Системи комп’ютерного оцінювання добре працюють із параметризованими задачами та кінцевими відповідями, але обмежено підтримують діагностику причин помилок і задачі на доведення. Адаптивні платформи можуть персоналізувати траєкторію, але не завжди забезпечують формальну коректність математичних міркувань. AI-асистенти створюють гнучкі пояснення, але потребують механізмів перевірки, оскільки генеративний текст не гарантує математичної правильності [9, 64, 99, 110].

Аналіз існуючих платформ показує, що практична сила сучасних систем полягає у масштабованості й доступності, однак їхнім слабким місцем лишається глибина діагностики. У більшості випадків система бачить правильність відповіді, але не завжди встановлює, чому відповідь неправильна. Для задач на

доведення, виведення формул або побудову алгоритмів це обмеження стає критичним, оскільки навчальна цінність полягає саме у структурі міркування.

Таблиця 1.5 – Узагальнена характеристика цифрових платформ підтримки навчання математики

Платформа / клас систем	Сильні сторони	Обмеження щодо задач дисертації
WeBWorK, IMathAS [41, 52, 64, 98]	Алгоритмічна генерація задач, автоматична перевірка, масове використання в курсах математики.	Обмежена діагностика причин помилок і підтримка складних доведень.
Khan Academy / Khanmigo [60, 64]	Великий навчальний контент, адаптивні вправи, AI-підтримка і персоналізація.	Потреба у формальній перевірці математичної коректності складних міркувань.
GeoGebra [64]	Сильна візуалізація, інтерактивне дослідження алгебраїчних і геометричних об’єктів.	Не є повноцінною діагностичною ITS для виявлення причин помилок.
IXL, Mangahigh, LearnBop [64]	Практика, пояснення, адаптація до рівня, ігрові механіки.	Переважна орієнтація на вправи і результати, а не на формальну верифікацію.
AI-асистенти та LLM-підходи [9, 99, 110]	Генерація пояснень, варіантів умов, діалоговий формат підтримки.	Ризик помилкових або неперевірених математичних тверджень без зовнішнього контролю.

Рисунок 1.10 – Порівняльна схема сучасних цифрових платформ: автоматизоване оцінювання, адаптивне навчання, візуалізація, AI-тьюторинг, proof-checking.

1.7 Системи формальної верифікації доказів і алгоритмів у навчанні

Системи формальної верифікації доказів і алгоритмів відкривають принципово новий рівень підтримки навчання точним наукам. Якщо традиційна система перевіряє відповідь або тестові приклади, то proof assistant перевіряє коректність міркування у формальній логіці. У цьому сенсі Lean, Rocq/Coq, Isabelle/HOL та інші системи дозволяють зробити навчання доведень і алгоритмів не лише пояснювальним, а й формально контрольованим [14, 17, 26, 28–29, 53, 70, 72–73, 79, 96, 108, 116, 119–120].

Lean і бібліотека Mathlib є особливо важливими для математичної освіти, оскільки Mathlib містить велику кількість формалізованих математичних понять, лем і теорем [14, 28–29, 73]. У навчальному контексті це дозволяє

використовувати формально доведені твердження як джерело коректного теоретичного матеріалу. При цьому науковий внесок полягає не у повторному доведенні класичних теорем, а в побудові методів перетворення формального артефакту на поетапний навчальний сценарій.

Rocq Prover, раніше відомий як Coq, є інтерактивним довідником, що призначений для розроблення математичних доведень, формальних специфікацій і доказів відповідності програм специфікаціям [96]. Isabelle/HOL підтримує інтерактивне доведення у логіці вищого порядку і має розвинені засоби формалізації математичних і програмних тверджень [79]. Для навчання алгоритмів ці системи важливі тим, що дозволяють перевіряти не лише результат виконання, а й властивості алгоритму: коректність, інваріанти, завершуваність, відповідність специфікації [13, 31, 38–39, 47, 51, 65–66].

Окремим прикладом освітньої адаптації proof assistant є Waterproof – програмне середовище і плагін для Coq/Rocq, створені для навчання студентів написанню математичних доведень [117–119]. Waterproof дозволяє записувати доведення у стилі, ближчому до рукописних математичних текстів, і надає зворотний зв’язок щодо логічної коректності кроків. Це важливий приклад того, що системи формальної верифікації можуть бути не лише інструментами дослідника, а й навчальним середовищем.

Водночас формальна верифікація має високий поріг входу. Синтаксис proof assistant, вимоги до формалізації та необхідність точного опису кожного кроку можуть ускладнювати навчання, якщо студент змушений одночасно опановувати і математику, і формальну мову.

Тому перспективним напрямом є створення проміжного навчального інтерфейсу: система використовує Lean/Mathlib або Rocq як формально верифіковане ядро, але студент взаємодіє з поетапним поясненням, підказками, контрольними питаннями і лише поступово наближається до формального синтаксису.

Таблиця 1.6 – Роль систем формальної верифікації у навчанні точним наукам

Система / підхід	Навчальний потенціал	Обмеження
Lean / Mathlib [14, 28–29, 73]	Формально доведені математичні твердження, велика бібліотека, можливість створення навчальних сценаріїв на основі доказів.	Потреба у дидактичному інтерфейсі, що приховує надмірну складність синтаксису.
Rocq/Coq [17, 26, 96]	Формалізація доказів і програмних специфікацій, перевірка властивостей алгоритмів.	Високий поріг входу для студентів без попередньої підготовки.
Isabelle/HOL [79]	Формалізація в логіці вищого порядку, сильна доказова інфраструктура.	Складність інтеграції у масові LMS без адаптованого інтерфейсу.
Waterproof [117–119]	Освітній proof-checking для навчання доведенням у стилі, близькому до рукописних доказів.	Потреба в адаптації до конкретних курсів і навчальних сценаріїв.

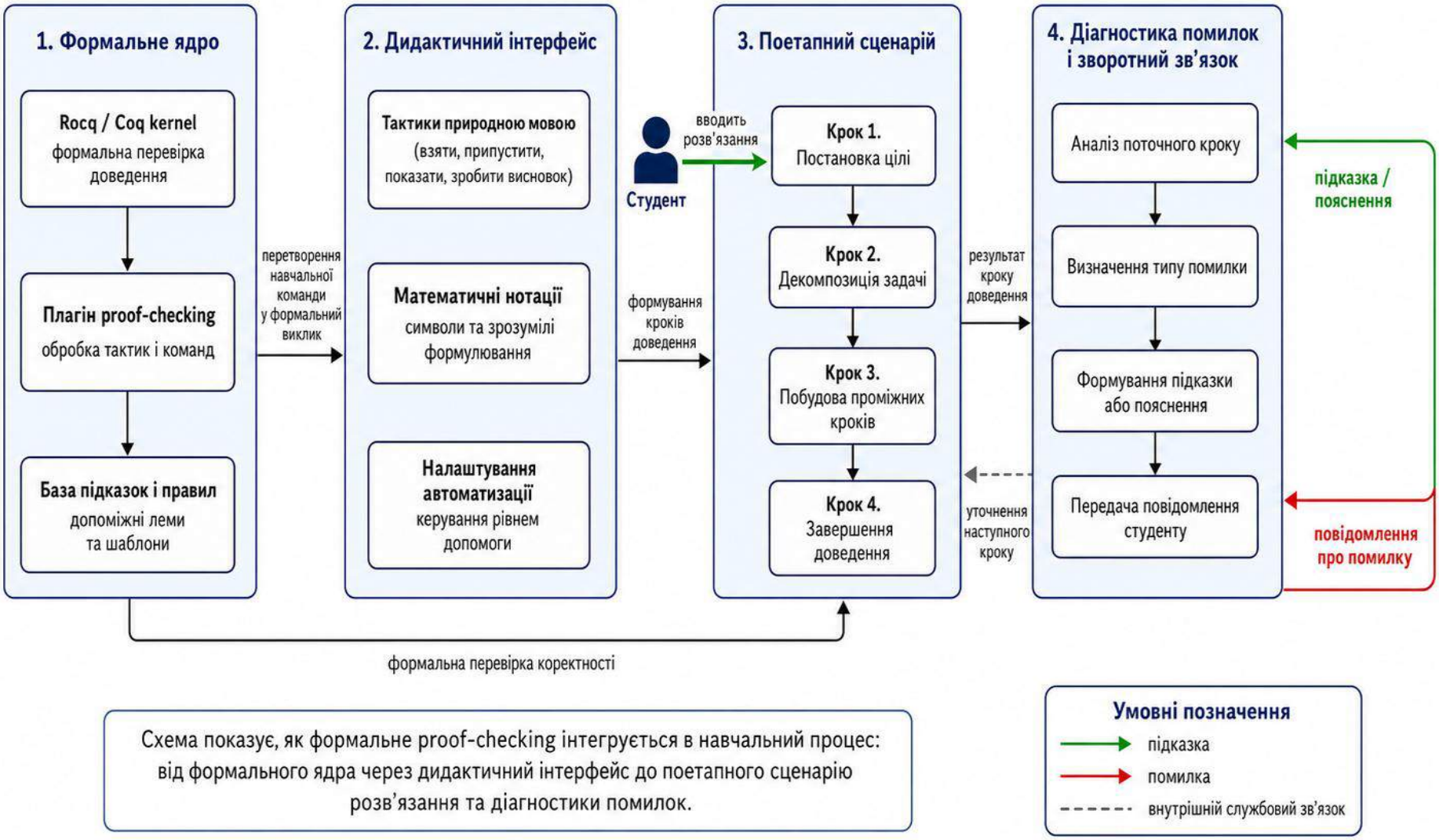


Рисунок 1.11 – Схема інтеграції proof-checking у навчання: формальне ядро → дидактичний інтерфейс → поетапний сценарій → діагностика помилок

1.8 Генеративний штучний інтелект у навчанні точним наукам: можливості та обмеження

Генеративний штучний інтелект може бути корисним у навчанні точним наукам як інструмент створення пояснень, варіацій умов задач, формулювання підказок, перекладу формального змісту природною мовою та підтримки діалогового формату навчання [9, 99, 110]. Для викладача такі інструменти можуть зменшувати трудомісткість підготовки матеріалів, а для студента – створювати відчуття доступного індивідуального пояснення.

Разом із тим, генеративна модель не є математичним доказовим механізмом. Вона може створювати правдоподібний текст, але не гарантує коректності математичного висновку, правильності формули або допустимості кроку доведення. У задачах точних наук це обмеження є принциповим: навіть невелика помилка у логічному кроці, області допустимих значень або інваріанті алгоритму може повністю зруйнувати розв’язання. Саме тому генеративний ШІ повинен використовуватися не як остаточний арбітр правильності, а як допоміжний шар над формалізованими або алгоритмічно перевірюваними структурами.

Сучасні досягнення у поєднанні машинного навчання з формальним або символьним доведенням демонструють перспективність такого напряму. Наприклад, AlphaGeometry поєднує нейросимвольний підхід із пошуком доведень у геометрії та показує, що штучний інтелект може бути корисним для складних математичних задач, коли його робота спирається на структуроване подання та перевірку [109]. Однак такі результати не усувають потреби у навчальному інтерфейсі, діагностиці й адаптації до студента.

Для цієї дисертації генеративний ШІ розглядається як потенційний допоміжний інструмент у трьох місцях: формування природномовних пояснень до формально перевірених тверджень; генерація варіантів текстових умов задач на основі параметричного шаблону; створення підказок і зворотного зв’язку за результатами діагностики. В усіх трьох випадках критичні математичні

перевірки мають залишатися в алгоритмічному або формально верифікованому ядрі системи [64, 99, 110].

Отже, перспективна інформаційна технологія підтримки навчання точним наукам повинна не протиставляти генеративний ШІ формальним методам, а поєднувати їхні сильні сторони: генеративну мовну гнучкість, алгоритмічну точність, діагностичну інтерпретованість і формальну строгість.

Таблиця 1.7 – Можливості та обмеження генеративного ШІ у навчанні точним наукам

Можливість	Корисний ефект	Необхідне обмеження
Генерація пояснень	Зменшення трудомісткості підготовки навчального контенту.	Пояснення мають спиратися на перевірений математичний артефакт.
Варіативність умов задач	Створення різноманітних текстових постановок.	Параметри й відповіді мають перевірятися алгоритмічно.
Діалогові підказки	Індивідуальні пояснення та підтримка студента.	Підказки повинні узгоджуватися з діагностичною моделлю помилки.
Автоматизація навчальних сценаріїв	Швидке формування чернеток поетапних матеріалів.	Фінальна логіка сценарію має проходити контроль викладача або proof-checking.

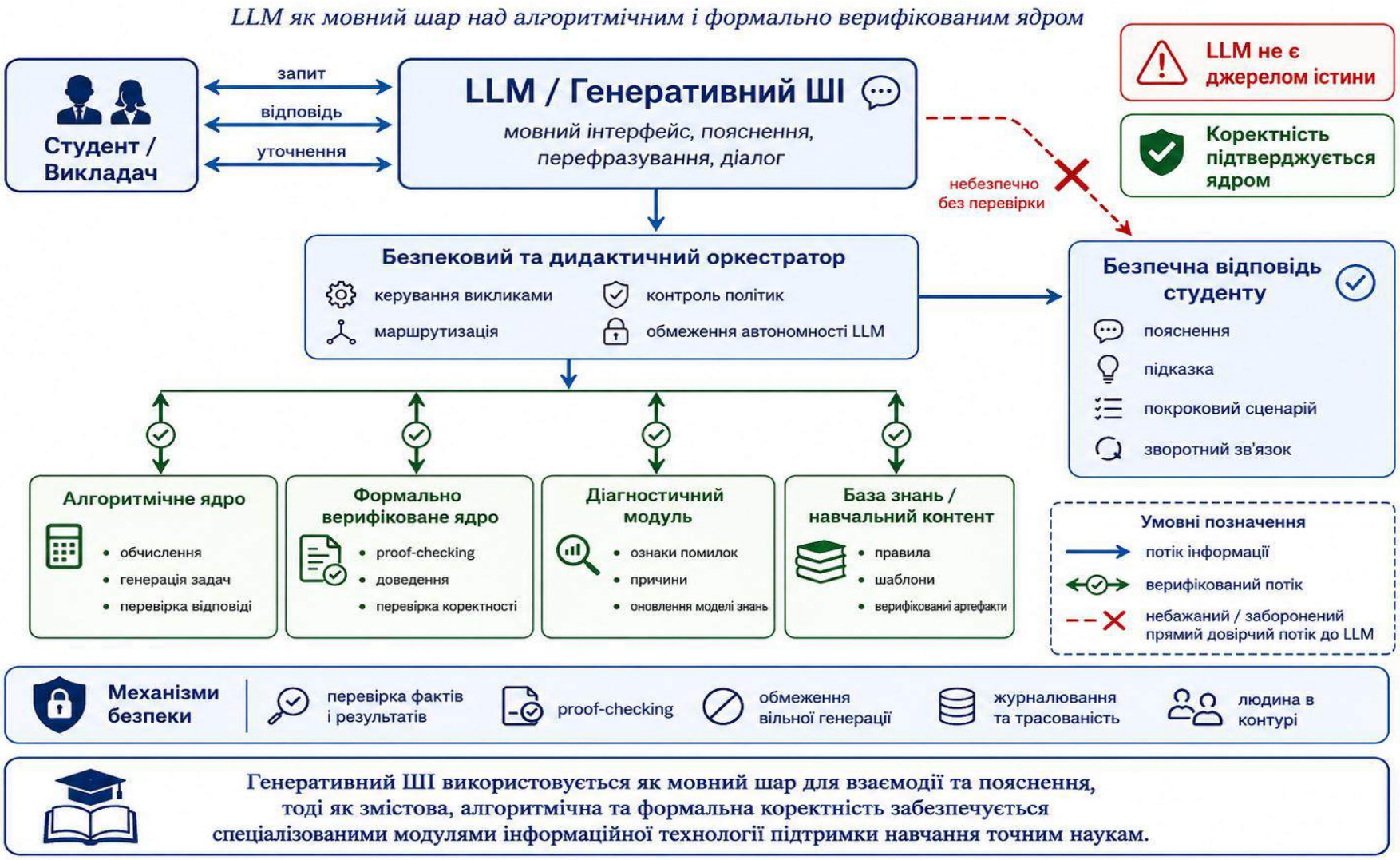


Рисунок 1.12 – Схема безпечного використання генеративного ШІ у навчанні точним наукам: LLM як мовний шар над алгоритмічним і формально верифікованим ядром

1.9 Постановка науково-прикладного завдання та вимоги до інформаційної технології

Проведений аналіз показує, що сучасний стан проблеми підтримки навчання точним наукам характеризується одночасною наявністю кількох викликів. По-перше, національні й міжнародні дані засвідчують складність підтримання високого рівня математичної підготовки [4–6, 82–83]. По-друге, традиційні цифрові платформи переважно автоматизують подання і перевірку задач, але не завжди забезпечують глибоку діагностику причин помилок [41, 52, 64, 98]. По-третє, складні задачі на доведення, виведення та побудову алгоритмів потребують формальної строгості, яку не гарантують звичайні адаптивні або генеративні системи [14, 17, 26, 28–29, 53, 70, 72–73, 79, 96, 108, 116–120].

У зв’язку з цим науково-прикладне завдання дисертації полягає у створенні комплексу моделей, методів та інформаційної технології підтримки навчання точним наукам, що поєднує: формалізоване подання задач; діагностичні моделі встановлення причин помилок; адаптивний вибір задач; алгоритмічну генерацію параметризованих задач; інтеграцію формальної верифікації математичних доведень і алгоритмів; підтримку мотиваційного регулювання.

Вимоги до інформаційної технології можна поділити на функціональні, діагностичні, математичні, інженерні та педагогічні. Функціональні вимоги стосуються підтримки повного навчального циклу: вибір задачі, розв’язання, перевірка, діагностика, оновлення стану знань, мотиваційний сигнал і вибір наступного кроку. Діагностичні вимоги передбачають аналіз непрямих ознак помилки та встановлення її ймовірної причини. Математичні вимоги стосуються коректності задач, параметрів, відповідей і доведень. Інженерні вимоги передбачають модульну архітектуру, API, збереження даних і можливість інтеграції з зовнішніми засобами. Педагогічні вимоги полягають у пояснюваності, адаптивності, доступності та підтримці внутрішньої мотивації.

Підсумовуючи, можна сформулювати основну суперечність: сучасна освіта потребує масштабованих цифрових засобів, здатних підтримувати індивідуальне навчання точним наукам, однак наявні системи здебільшого не поєднують у межах єдиної технології глибоку діагностику, параметричну генерацію задач, формальну верифікацію і мотиваційне регулювання. Подолання цієї суперечності визначає зміст наступних розділів дисертації.

Таблиця 1.8 – Вимоги до інформаційної технології підтримки навчання точним наукам

Група вимог	Зміст вимог	Зв’язок із подальшими розділами
Функціональні	Підтримка повного навчального циклу, генерації задач, перевірки, підказок і адаптації.	Машинна модель і прототип у розділі 4.
Діагностичні	Встановлення причин помилкових відповідей за непрямими ознаками.	Діагностичні моделі в розділі 2 і методи у розділі 3.
Математичні	Формалізація задач, предикатів коректності, стану знань і операторів оновлення.	Математична модель системи в розділі 2.
Формально-верифікаційні	Інтеграція Lean/Mathlib, Rocq/Coq, Isabelle/HOL або споріднених засобів для доведень і алгоритмів.	Методи формальної верифікації в розділі 3.
Педагогічні	Пояснюваність, поетапність, адаптивність, мотиваційна підтримка.	Навчальні сценарії, мотиваційна модель і прототип.

Висновки до розділу 1

1. Точні науки є фундаментальною основою підготовки STEM-фахівців, оскільки формують логічне, алгоритмічне та модельне мислення, необхідне для інженерних і програмно-технічних спеціальностей.

2. Міжнародні та національні дані, зокрема PISA і звіти УЦОЯО щодо НМТ/ЗНО, підтверджують наявність стійких викликів у рівні математичної підготовки, хоча інтерпретація цих даних потребує врахування методологічних обмежень.

3. Однією з ключових системних проблем є примітивізація математичної освіти, що проявляється у переважанні шаблонних процедур над діяльністю з доведення, виведення, побудови алгоритмів і перевірки умов коректності.

4. Інтелектуальні навчальні системи є перспективним напрямом подолання зазначених проблем, однак для навчання точним наукам вони повинні підтримувати як зовнішній цикл адаптивного вибору задач, так і внутрішній цикл поетапної діагностики розв’язання.

5. Методи knowledge tracing забезпечують оцінювання стану знань, але для задач точних наук їх необхідно доповнювати діагностичними моделями, які встановлюють причини помилок за непрямими ознаками відповіді або поведінки студента.

6. Сучасні цифрові платформи підтримки навчання математики мають значний практичний потенціал, проте здебільшого не поєднують у межах єдиної системи глибоку діагностику, формальну верифікацію і адаптивне формування складних навчальних сценаріїв.

7. Системи формальної верифікації доказів і алгоритмів, зокрема Lean/Mathlib, Rocq/Coq, Isabelle/HOL і освітнє середовище Waterproof, створюють умови для переходу від поверхневої перевірки відповіді до контролю логічної коректності математичного міркування.

8. Генеративний штучний інтелект може бути корисним мовним і дидактичним шаром, але в навчанні точним наукам він повинен працювати у зв’язці з алгоритмічною перевіркою та формально верифікованим ядром.

9. Науково-прикладне завдання дисертації полягає у розробленні моделей, методів та інформаційної технології підтримки навчання точним наукам, які інтегрують діагностику, адаптацію, генерацію задач, формальну верифікацію та мотиваційне регулювання.

Рисунок 1.13 – Підсумкова схема постановки науково-прикладного завдання: проблема → прогалина → вимоги → моделі → методи → інформаційна технологія

РОЗДІЛ 2 СИСТЕМНЕ МОДЕЛЮВАННЯ ПІДТРИМКИ НАВЧАННЯ ТОЧНИМ НАУКАМ

2.1 Мета та постановка задачі моделювання навчального процесу

Побудова *системи підтримки навчання точним наукам* вимагає не лише опису окремих програмних функцій, а й цілісного моделювання навчального процесу як складної відкритої системи, у якій взаємодіють здобувач освіти, навчальні задачі, формалізовані знання, механізми контролю, адаптації та мотивації. Такий підхід узгоджується із системною традицією моделювання, відповідно до якої адекватний опис об’єкта досягається шляхом послідовного переходу від змістовного подання до структурного, функціонального, математичного та машинного рівнів. Для інтелектуальних навчальних систем це особливо важливо, оскільки помилка на будь-якому з рівнів призводить або до педагогічної неефективності, або до технічної нереалізованості, або до відсутності інтерпретованості отриманих результатів [1, 2, 4].

У даному розділі *навчальний процес* розглядається як система керованого набуття компетентностей у предметній області точних наук. Керованість означає, що система не лише фіксує результати діяльності здобувача освіти, а й формує наступні навчальні впливи: добір задач, вибір режиму допомоги, спосіб перевірки, форму подання матеріалу, а також мотиваційне підкріплення. Таке розуміння поєднує класичний для intelligent tutoring systems поділ на зовнішній та внутрішній цикли навчання з сучасними вимогами до формальної коректності математичних міркувань, алгоритмічної генерації контенту та роботи з багатокомпонентними станами знань [6, 24, 36].

Метою моделювання є побудова комплексу взаємопов’язаних моделей, які дозволяють описати: а) предметну область задач точних наук; б) стан знань студента та його зміну під впливом навчальних дій; в) механізми адаптивного вибору та генерації задач; г) інтеграцію формальної верифікації доведень і алгоритмів у навчальний процес; д) мотиваційне регулювання навчальної активності; е) перспективні можливості прискорення окремих обчислювальних

процедур у просторі навчальних задач. Інакше кажучи, мета полягає не лише у створенні окремих моделей, а у формуванні єдиної моделі-основи, на якій може бути побудована інформаційна технологія підтримки навчання [4, 5].

Постановка задачі моделювання спирається на *принципи цілеспрямованості, ідеалізації, декомпозиції, ітераційності, формалізації, абстрагування* та *перевірки*. Принцип цілеспрямованості означає, що межі моделі визначаються не повнотою опису реального освітнього процесу, а здатністю моделі відповісти на ключові для системи питання: як представити знання; як обрати наступну задачу; як перевірити коректність розв'язання; як класифікувати помилку; як пов'язати навчальний результат з мотиваційним сигналом. Принцип ідеалізації означає свідоме спрощення реального процесу навчання до такої міри, за якої зберігаються суттєві для дослідження відношення, але усуваються другорядні фактори, що не піддаються формалізації на поточному етапі [1, 2].

Принцип декомпозиції реалізується через виділення окремих підсистем: підсистеми *представлення задач*, підсистеми *діагностики знань*, підсистеми *формальної верифікації*, підсистеми *мотиваційного регулювання* та підсистеми *генерації варіантів задач*. Ітераційність означає, що побудова моделей не є одноразовою дією: вербальні, структурні, математичні та машинні моделі уточнюються на основі взаємних перевірок, експериментальної реалізації та порівняння з реальною поведінкою навчальної системи. Формалізація передбачає перехід від якісних описів до множин, відображень, функцій, логічних предикатів та алгоритмів. Абстрагування дозволяє працювати з моделлю як з самостійним математичним об'єктом, а перевірка вимагає оцінювання узгодженості між рівнями моделювання.

У межах цього розділу система підтримки навчання точним наукам моделюється для випадку, коли основною предметною сферою є навчання математики та суміжних алгоритмізовних дисциплін. Під алгоритмізовними задачами розуміються такі задачі, для яких можливо виділити або процедуру

одержання правильного результату, або формально перевірювану структуру доказу чи алгоритмічного розв’язання. Це дозволяє поєднати два принципово різні, але пов’язані класи задач: прості задачі на застосування правил і складні задачі на побудову алгоритмів, виведення формул та доведення тверджень [3, 4, 5].

Отже, задача системного моделювання полягає у побудові такого опису навчального процесу, який одночасно є педагогічно змістовним, математично формалізованим і програмно реалізовним. Результатом цього моделювання має стати ієрархія моделей: вербальна модель, що фіксує зміст і межі системи; структурна модель, що описує її компоненти та зв’язки; функціональна модель, що відображає логіку роботи системи; математична модель, що задає формальні залежності між її станами, спостереженнями та керуючими діями; і, нарешті, машинна модель, що реалізує ці залежності в алгоритмах та програмних модулях [1, 2].

Таблиця 2.1 – Рівні моделювання системи підтримки навчання точним наукам

Рівень моделювання	Основне призначення	Результат
Вербальний	Змістовний опис системи, її меж та сутностей	Понятєва схема предметної області
Структурний	Виділення компонентів та зв’язків	Графічна/блочна структура системи
Функціональний	Опис виконуваних функцій і циклів	Сценарії роботи та потоки даних
Математичний	Формалізація станів, функцій, предикатів, операторів оновлення	Система співвідношень і правил
Машинний	Алгоритмічна та програмна реалізація	Прототип інформаційної технології

Рисунок 2.1 – Блок-схема системного моделювання підтримки навчання точним наукам: перехід від вербальної моделі до машинної

2.2 Вербальна модель системи підтримки навчання точним наукам

Вербальна модель є початковим рівнем формалізації системи підтримки навчання точним наукам. Її призначення полягає в описі системи природною мовою так, щоб були однозначно визначені її межі, сутності, типи взаємодій і цільові результати. На цьому рівні не задаються точні математичні залежності, але фіксуються змістовні ролі кожного компонента, типи навчальних задач, характер навчальних дій та загальна логіка функціонування системи. Вербальна модель виконує роль мосту між педагогічною постановкою задачі та подальшою математичною формалізацією [1, 2].

Систему підтримки навчання точним наукам доцільно розглядати як людиномашинну навчальну систему з діагностичним ядром. Її центральним елементом є здобувач освіти, який взаємодіє з навчальними задачами, навчальним контентом і механізмами зворотного зв'язку. На відміну від

звичайної системи тестування, де увага зосереджена на фіксації правильної чи неправильної відповіді, у даній системі основним об’єктом аналізу є процес розв’язання: проміжні кроки, структура перетворень, характер помилок, послідовність дій, час реакції та ознаки залученості здобувача освіти [6, 24].

У вербальній моделі виділяються такі базові сутності: студент; викладач або асистент курсу; навчальна задача; навчальний компонент знань; еталонне рішення; формально верифікований артефакт; діагностична модель; мотиваційна подія; токенізована винагорода; сховище навчальних даних; модуль генерації; модуль адаптації; модуль рекомендацій. Студент є джерелом навчальних дій і носієм поточного стану знань. Викладач виступає носієм предметної моделі та джерелом правил формування задач, критерію коректності, а також конфігурації навчального курсу. Навчальна задача є мінімальною змістовною одиницею навчального впливу.

Навчальна задача у системі не зводиться лише до текстової умови. Вона включає множину правил і концепцій, потрібних для її розв’язання, набір можливих формулювань, множину вхідних даних, допустимі типи відповідей і множину процедур чи доказових стратегій, які приводять до результату. Для частини задач ці процедури можуть бути виражені алгоритмічно, а для іншої частини – у формі доказових кроків, що потребують формальної верифікації. Таким чином, задача є не лише тестовим запитанням, а об’єктом, що поєднує предметний зміст, спосіб контролю та місце в індивідуальній траєкторії навчання [5, 22, 23].

Стан знань студента у вербальній моделі трактується як сукупність оцінок засвоєння окремих компонентів знань і вмінь. Під компонентом знань розуміється мінімальна змістовна одиниця, володіння якою суттєво впливає на можливість розв’язування певного класу задач. Прикладами таких компонентів для вищої математики можуть бути вміння розпізнавати тип коренів характеристичного рівняння, застосовувати правило інтегрування частинами, аналізувати структуру матриці, будувати загальний вигляд розв’язку або

коректно виконувати елементарні алгебраїчні перетворення. Один і той самий компонент може входити до складу різних задач, а одна задача, навпаки, може вимагати одночасного використання кількох компонентів [7, 8, 9].

Вербальна модель підтримує існування двох взаємодоповнювальних циклів навчання. Зовнішній цикл відповідає за вибір наступної задачі, режиму її подання та місця в індивідуальній траєкторії. Внутрішній цикл працює на рівні поточної задачі й забезпечує покроковий аналіз розв'язання, виявлення помилки, видачу підказки, демонстрацію наступного кроку або переведення студента до режиму самостійного завершення. Поєднання цих двох циклів дозволяє системі одночасно вирішувати завдання довгострокової адаптації та короткострокової адресної допомоги [6, 26, 27].

Для *простих задач* основна функція системи полягає у перевірці правильності відповіді, контролі коректності параметрів згенерованого варіанта та, за потреби, у поясненні причини помилки. Для *складних задач*, пов'язаних із побудовою алгоритмів, доведенням теорем або виведенням формул, система повинна аналізувати не лише результат, а й структуру міркування. У таких випадках виникає потреба в особливому типі навчального контенту, який може бути частково або повністю пов'язаний з формально верифікованими об'єктами в середовищах **Lean**, **Coq/Rocq** чи **Isabelle/HOL**.

Модуль формальної верифікації в межах вербальної моделі виконує дві функції. По-перше, він забезпечує остаточний контроль коректності математичних доведень і алгоритмів, знімаючи частину навантаження з людини та переносячи контроль істинності на мале довірене ядро *proof assistant*. По-друге, він виступає джерелом структурованого, перевіреного навчального контенту: студент може отримувати не лише сигнал «правильно/неправильно», а й поетапне подання доведення або алгоритму, сформоване на основі формально перевіреного артефакту [17, 18, 20]. Вербальна модель передбачає також існування мотиваційного шару системи. На відміну від звичайної гейміфікації, яка часто зводиться до зовнішніх балів, рейтингів чи значків, у даній системі

мотиваційні стимули пов’язуються із якісними навчальними подіями: самостійним виправленням концептуальної помилки, виявленням глибокої структурної закономірності, наполегливістю без надмірної кількості підказок, оригінальним коректним способом розв’язання або внеском до колективної бази знань. Такі події інтерпретуються як носії навчальної цінності і можуть бути перетворені на цифрові винагороди [12, 14, 15].

Суттєвим елементом вербальної моделі є збереження історії взаємодії. Система повинна фіксувати не лише остаточний результат виконання задачі, а й послідовність проміжних дій, час між спробами, кількість звернень до підказок, повторні спроби, зміни введеного виразу та інші характеристики взаємодії. Саме ця історія дозволяє розрізняти поверхневу помилку від глибинного нерозуміння, а також формувати поведінковий профіль студента, що використовується в адаптації та мотиваційному регулюванні.

Таблиця 2.2 – Основні сутності вербальної моделі

Сутність	Змістовне тлумачення	Роль у системі
Студент	Носій поточного стану знань та навчальної активності	Генерує дії, відповіді та поведінкові сигнали
Навчальна задача	Об’єкт контролю й навчального впливу	Перевіряє та формує компоненти знань
Компонент знань	Мінімальна змістовна одиниця предметної області	Складає основу моделі знань
Діагностична модель	Модель інтерпретації правильних і помилкових дій	Оновлює стан знань
Формально верифікований артефакт	Доведення або алгоритм, перевірені proof assistant	Гарантує строгість теоретичного контенту
Мотиваційна подія	Осмислена навчальна дія, що має цінність	Ініціює цифрову винагороду

Таким чином, вербальна модель системи підтримки навчання точним наукам описує її як діагностично-керовану навчальну екосистему, де змістовні задачі, знання, перевірка, адаптація, формальна верифікація та мотивація поєднані в єдиний замкнений цикл. У цьому циклі кожна навчальна дія породжує нові дані про стан знань, а кожен новий стан знань впливає на вибір наступної дії системи. Саме така логіка стає підґрунтям для побудови структурної, функціональної та математичної моделей у наступних підрозділах.

Рисунок 2.2 – Вербальна концептуальна схема системи: студент, задача, знання, діагностика, верифікація, мотивація, адаптація

2.3 Структурна модель системи

2.3.1 Основні компоненти системи

Структурна модель конкретизує вербальну модель через виділення компонентів системи та зв'язків між ними. Якщо вербальна модель відповідає на питання «що являє собою система», то структурна модель відповідає на питання «з яких частин вона складається». Для систем підтримки навчання точним

наукам така декомпозиція має принципове значення, оскільки різні функції системи мають різну природу: педагогічну, алгоритмічну, логіко-формальну, мотиваційну та інфраструктурну [1, 2].

У найзагальнішому вигляді система складається з інтерфейсного шару, діагностичного ядра, шару генерації та добору задач, шару формальної верифікації, мотиваційного шару та інфраструктурного шару збереження і обміну даними. Інтерфейсний шар забезпечує взаємодію зі студентом і викладачем, діагностичне ядро інтерпретує результати навчальних дій, генератор задач формує або добирає наступний варіант, шар формальної верифікації перевіряє строгі математичні артефакти, мотиваційний шар трансформує значущі навчальні події у винагороди, а інфраструктурний шар забезпечує стійке збереження стану системи та журналювання подій [6, 10, 24].

Компонент студентського інтерфейсу повинен підтримувати кілька режимів роботи: ознайомлення з умовою задачі, покрокове введення розв’язання, отримання підказок, перегляд формально верифікованих пояснень, а також перегляд мотиваційного профілю і отриманих винагород. Для складних задач інтерфейс має бути здатний подавати математичні вирази, символи, формули, а в перспективі – забезпечувати дружню до користувача надбудову над формальною мовою Lean. Це означає, що інтерфейс не є пасивним засобом введення, а активним компонентом, який формує тип доступної студенту взаємодії [17, 18, 21].

Компонент моделі студента зберігає поточні оцінки засвоєння компонентів знань, історію взаємодій, параметри поведінкового профілю та стан мотивації. Його структура має бути такою, щоб підтримувати як дискретне оновлення ймовірностей засвоєння окремих умінь, так і накопичення безперервних характеристик взаємодії – часу відповіді, звернень до підказок, повторних спроб, ознак зосередженості або пасивності. У структурній моделі цей компонент виступає центральним сховищем стану навчального процесу [7, 8, 9].

Діагностичний компонент призначений для аналізу результату та процесу розв’язання. Для простих задач його функція зводиться до контролю правильності відповіді та визначення того, які компоненти знань були задіяні. Для складних задач він взаємодіє з модулем структурного аналізу відповіді, який подає математичний вираз у вигляді дерева операцій, знаходить перше структурне розходження з еталонним розв’язанням і відносить його до одного з наперед заданих типів помилок. Саме діагностичний компонент є джерелом сигналів для оновлення стану знань та видачі цільових підказок [23, 25].

Генератор задач у структурній моделі представлено як окремий компонент, оскільки він працює з власною моделлю простору задач і має специфічні вхідні дані: шаблон задачі, домен параметрів, множину обмежень на коректність, функцію обчислення еталонної відповіді та процедуру перевірки допустимості. Генератор не лише створює нові параметризовані задачі, але й готує службові артефакти для перевірки: правильну відповідь, набір допустимих варіантів формулювання, а інколи і діагностичні шаблони типових помилок.

Компонент формальної верифікації об’єднує дві різні, але пов’язані функції. Перша – це формальна перевірка математичних доведень, де об’єктом верифікації є твердження, доказові кроки та відповідний proof term. Друга – це формальна перевірка алгоритмів, де об’єктом виступає алгоритмічне рішення, його специфікація та властивості коректності. У структурній моделі цей компонент відокремлюється від інтерфейсу та генеративних сервісів, оскільки лише довірене ядро proof assistant може виступати остаточним критерієм істинності [17, 18, 19].

Мотиваційний компонент реалізує сенсо-економічний підхід до навчальних винагород. На відміну від традиційного шару гейміфікації, який обмежується балами та бейджами, цей компонент повинен зберігати типи значущих навчальних подій, їхню оцінку, правила відображення у цифрові токени, журнали нарахувань та процедури їх подальшого використання. У разі використання смартконтрактів структурна модель включає також приватну

блокчейн-мережу, сховище токенів та керуючий оракул, який отримує від діагностичного ядра підтверджені події навчальної цінності [15, 16, 37].

Інфраструктурний шар об’єднує сховище задач, сховище еталонних рішень, журнали дій студентів, профілі користувачів, сховище мотиваційних подій, а також програмні інтерфейси взаємодії між модулями. Для реального прототипу це означає наявність серверної частини, бази даних, API, сервісів обробки математичних виразів та окремих підсистем, які можуть працювати синхронно або асинхронно. Структурна модель на цьому рівні є переходом до майбутньої машинної моделі, але ще не містить деталізації конкретних класів і таблиць бази даних.

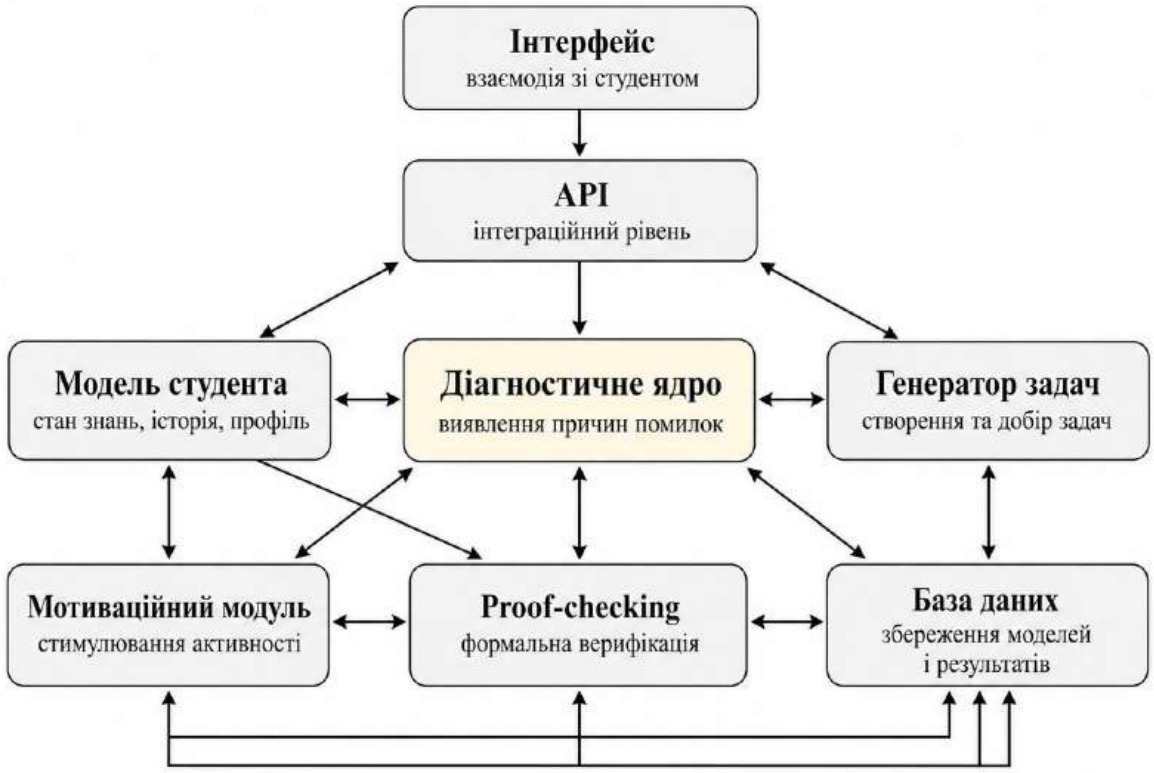


Рисунок 2.3 – Структурна схема системи з компонентами: інтерфейс, модель студента, діагностичне ядро, генератор задач, proof-checking, мотиваційний модуль, база даних, API

Таким чином, основні компоненти системи формують багатошарову архітектуру, де кожний шар виконує власну роль, але всі вони підпорядковані спільній меті – підтримці навчання точним наукам на основі діагностики, адаптації, формальної строгості та мотиваційної підтримки.

Таблиця 2.3 – Основні структурні компоненти системи

Компонент	Основні функції	Основні вхідні/вихідні дані
Студентський інтерфейс	Подання задач, введення кроків, підказки, візуалізація стану	Вхід: задача, підказки; вихід: відповіді, кроки, взаємодії
Модель студента	Зберігання стану знань і поведінкового профілю	Вхід: результати дій; вихід: оцінки знань
Діагностичне ядро	Оцінка правильності та класифікація помилок	Вхід: відповіді, еталони; вихід: діагностичні сигнали
Генератор задач	Створення параметричних варіантів і еталонів	Вхід: шаблон, обмеження; вихід: задача, відповідь
Модуль формальної верифікації	Перевірка доведень і алгоритмів	Вхід: формальний об’єкт; вихід: результат верифікації
Мотиваційний модуль	Оцінка значущих подій, токенізація	Вхід: HSS, події; вихід: винагорода
Інфраструктурний шар	Зберігання, API, журналювання	Вхід/вихід: дані всіх модулів

Взаємодія між компонентами

Взаємодія між компонентами системи визначає її здатність працювати як єдиний керований контур. На відміну від статичного переліку модулів, який ще не гарантує цілісності системи, саме характер потоків даних та керуючих впливів дозволяє розглядати інтелектуальну навчальну програму як кібернетичну систему підтримки навчання. У цій системі вихід одного компонента стає

вхідним сигналом для іншого, а інформація про результат навчальної дії повертається у модель студента й впливає на наступне керування.

Базовий цикл взаємодії починається з формування або вибору задачі. Зовнішній планувальник звертається до моделі студента та генератора задач, щоб підібрати такий варіант навчального завдання, який відповідає поточному рівню засвоєння, не є тривіальним і водночас не перевищує допустимого рівня складності. Після цього задача та пов’язаний із нею навчальний контент передаються в студентський інтерфейс. Отже, перший потік даних має напрям «модель студента → генератор/добір → інтерфейс».

Після початку розв’язання в систему надходять не лише кінцеві відповіді, а й проміжні кроки, виправлення, час між спробами, запити на підказку, варіації у введенні математичного виразу. Ці дані одночасно використовуються двома компонентами. По-перше, вони надходять до діагностичного ядра, яке визначає правильність поточного кроку, тип помилки або факт успішного просування. По-друге, вони фіксуються в модулі поведінкових спостережень, який накопичує параметри залученості та стійкості навчальної активності. Таким чином, один і той самий набір навчальних дій породжує два типи сигналів: дискретний предметний і неперервний поведінковий.

Якщо задача належить до класу простих, діагностичне ядро взаємодіє переважно з алгоритмічним модулем перевірки відповіді. Якщо ж задача є складною, проміжні дані також передаються до модуля формального або структурного аналізу. У випадку математичних перетворень система будує дерево виразу й порівнює його з еталонною структурою; у випадку доведення або алгоритму можливе перетворення людсько-орієнтованого подання до формального ядра Lean. Звідси випливає, що взаємодія між інтерфейсом, діагностикою та proof-checking є не послідовною, а умовно-розгалуженою: вона визначається класом задачі та типом навчальної дії.

Результат діагностики повертається до моделі студента у формі оновлення оцінок засвоєння відповідних компонентів знань. Якщо виявлено правильне

застосування правила, імовірність засвоєння зростає; якщо зафіксовано концептуальну помилку або типову структурну невідповідність, оновлюється стан відповідних навичок і водночас активується прив’язана до цього типу помилки діагностична модель. У такий спосіб формується замкнений зворотний зв’язок: відповіді студента змінюють модель знань, а оновлена модель знань змінює майбутній добір задач.

Потік взаємодії з мотиваційним модулем проходить паралельно до предметного діагностичного потоку. Мотиваційний модуль не підміняє перевірку знань, а отримує від діагностичного ядра вже інтерпретовані події: самостійне виправлення, структурне розуміння, наполегливе виконання без надмірних підказок, оригінальний коректний спосіб розв’язання, пояснення для інших. На основі цих подій модуль обчислює величину винагороди і, за потреби, ініціює операції смартконтракту чи запис в систему токенізованих досягнень. Таким чином, мотиваційний потік залежить від предметної інтерпретації, але не впливає безпосередньо на істинність або хибність розв’язання.

Окремим напрямом взаємодії є робота викладача або адміністратора курсу. Викладач поповнює бібліотеку шаблонів задач, визначає множини правил і концепцій, позначає зв’язки між задачами та компонентами знань, уточнює типові помилки, а також налаштовує параметри генерації та мотиваційних ваг. Ці дії не входять до поточного студентського навчального циклу, але утворюють метарівень керування, на якому система навчається в широкому розумінні – через накопичення нових еталонів, сценаріїв і діагностичних моделей.

З точки зору потоків даних, система містить три ключові контури: навчальний контур, у якому рухаються задачі, відповіді та підказки; діагностичний контур, у якому рухаються оцінки знань, типи помилок та поведінкові ознаки; мотиваційний контур, у якому рухаються значущі події, значення токенів та журнали винагород. Узгодження цих трьох контурів є принциповою умовою цілісності системи. Якщо вони роз’єднані, навчання або

втрачає адаптивність, або перетворюється на зовнішньо мотивований, але педагогічно слабкий процес [11, 12, 16].

Отже, структурна модель взаємодії між компонентами показує, що система підтримки навчання точним наукам є мережею пов'язаних сервісів і моделей, а не просто набором незалежних модулів. Саме завдяки цим зв'язкам стає можливою побудова функціональної моделі системи, у якій взаємодія компонентів розглядається як впорядкована послідовність функцій і переходів.

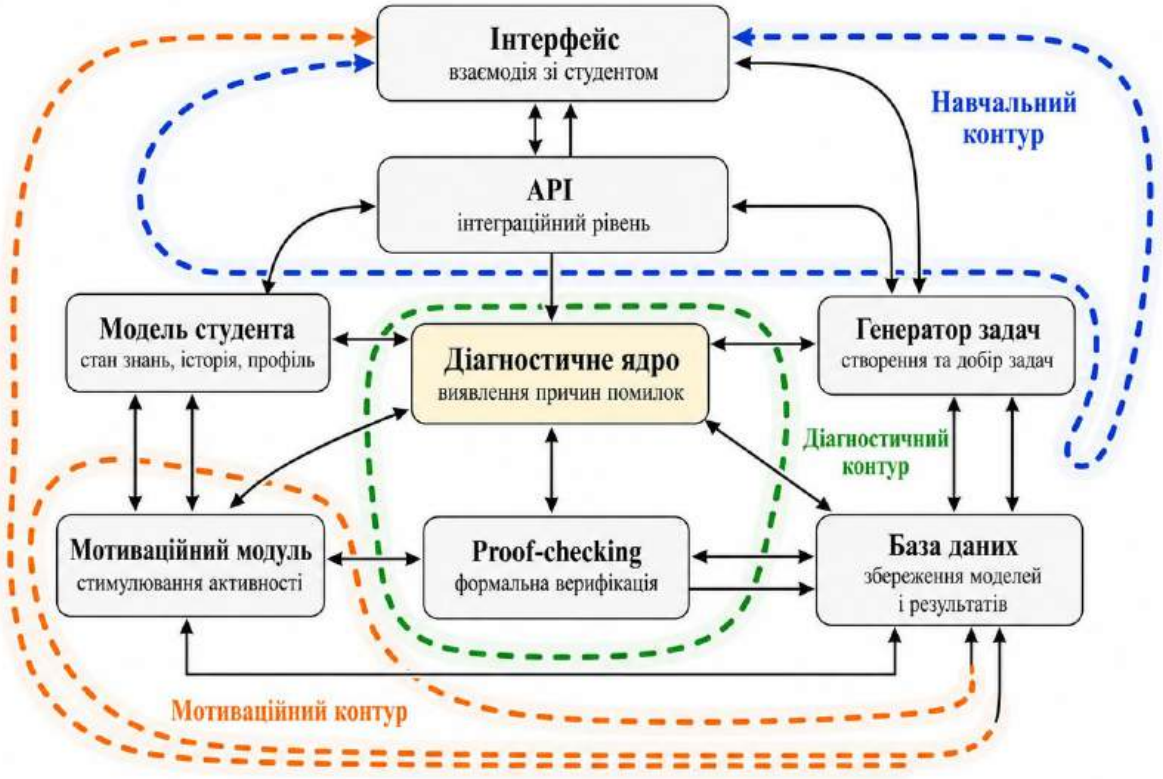


Рисунок 2.4. – Схема потоків даних між компонентами системи з трьома контурами: навчальним, діагностичним і мотиваційним

2.4 Функціональна модель системи

Функціональна модель системи відображає не її склад, а порядок виконання функцій, через який досягається педагогічний результат. Якщо структурна модель відповідає на питання «з яких елементів побудована система», то функціональна модель описує «що саме робить система», «у якій послідовності» та «як дані перетворюються на керуючі рішення». Для системи

підтримки навчання точним наукам це особливо важливо, оскільки її ефективність визначається не наявністю окремих модулів, а узгодженою роботою циклів адаптації, діагностики, перевірки та мотивації [6, 10].

У функціональному сенсі система реалізує замкнений навчальний цикл. На початку кожної ітерації формується поточний навчальний вплив: обирається або генерується задача, визначається режим її подання, за необхідності добирається пов'язаний навчальний контент. Далі студент виконує навчальну дію – читає умову, вводить відповідь, виконує проміжний крок, звертається до підказки або переглядає пояснення. Ця дія породжує нові дані, які піддаються перевірці та інтерпретації, після чого оновлюється стан знань і формується наступний крок навчального процесу [6, 26, 27].

На цьому рівні систему можна описати множиною функцій

$$F = \{F_{sel}, F_{gen}, F_{eval}, F_{diag}, F_{ver}, F_{upd}, F_{mot}, F_{cnt}\}$$

де F_{sel} – функція добору задачі, F_{gen} – функція генерації параметризованого варіанта, F_{eval} – первинна перевірка відповіді, F_{diag} – діагностика знань і помилок, F_{ver} – формальна верифікація, F_{upd} – функція оновлення стану знань, F_{mot} – функція мотиваційного оцінювання, F_{cnt} – функція формування навчального контенту. Сукупність цих функцій утворює функціональне ядро системи.

$$F = \{F_{sel}, F_{gen}, F_{eval}, F_{diag}, F_{ver}, F_{upd}, F_{mot}, F_{cnt}\} \quad (2.1)$$

Функція добору задачі F_{sel} працює в зовнішньому циклі навчання. Її призначення полягає у виборі такого завдання, яке одночасно відповідає поточному стану знань студента, підтримує педагогічний принцип «від простого до складного», не допускає різких стрибків складності та забезпечує навчальну новизну. На вхід цієї функції надходять: стан знань K_t, множина доступних шаблонів задач Ω, навчальні цілі курсу G та історія вже розв'язаних задач H_t. На виході формується шаблон або конкретна задача q_t [6, 7, 8].

Функція генерації F_{gen} застосовується тоді, коли обраний клас задач підтримує параметризацію. Вона перетворює шаблон задачі разом із доменом параметрів у конкретний варіант із коректною умовою, допустимими вхідними даними та еталонною відповіддю. Тут виконуються дві перевірки: допустимості параметрів і допустимості результату. Функція F_{gen} є критичною для забезпечення масштабованості системи, оскільки саме вона дозволяє уникати повторення одних і тих самих варіантів та формувати індивідуалізований навчальний потік.

Функція первинної перевірки F_{eval} визначає, чи відповідає отриманий студентський об’єкт очікуваному типу та формату. Для простих задач це може бути перевірка числового значення, еквівалентності формул або відповідності правилам синтаксису. Для складних задач F_{eval} виконує лише первинний контроль і передає дані в глибші модулі. Таким чином, F_{eval} не завжди є остаточним засобом встановлення істинності, але майже завжди виступає першим фільтром некоректних або неповних відповідей.

Функція діагностики F_{diag} є центральною в навчальному циклі. Вона аналізує не лише факт помилки, а й її природу. Для цього використовуються типові діагностичні моделі, структурний аналіз виразів, а також інформація про послідовність дій студента. Результатом роботи F_{diag} є не просто бітова ознака «правильно/неправильно», а структурований діагностичний сигнал d_t, який вказує на задіяні компоненти знань, тип помилки, імовірний рівень її концептуальності та можливий напрям виправлення [23, 25].

Функція формальної верифікації F_{ver} активується для тих класів задач, де контроль істинності не може бути надійно забезпечений лише числовою чи синтаксичною перевіркою. Її роль полягає у перетворенні людсько-орієнтованого запису в консервативне формальне подання та в повторній перевірці формального артефакту малим довіреним ядром proof assistant. На функціональному рівні це означає, що система має двошаровий механізм

перевірки: швидкий педагогічний шар і строгий формальний шар, який виконує роль остаточного арбітра.

Функція оновлення F_{upd} перетворює результати діагностики й верифікації на новий стан знань. Саме на цьому кроці система навчається разом зі студентом: кожна навчальна дія змінює внутрішню модель засвоєння, а не лише записується в журнал. Функція оновлення повинна підтримувати як покомпонентне оновлення ймовірностей засвоєння, так і зміну агрегованих характеристик – рівня складності доступних задач, історії проходження курсу, поведінкового профілю тощо [7, 8, 9].

Функція мотиваційного оцінювання F_{mot} працює поверх предметної діагностики. Вона інтерпретує результати F_{diag} і F_{upd} з точки зору значущості навчальної події. Наприклад, звичайна правильна відповідь може не породжувати винагороду, тоді як самостійне виправлення концептуальної помилки після серії осмислених кроків може породити подію типу recovery sense. Тобто F_{mot} не дублює оцінювання знань, а накладає на нього додатковий шар інтерпретації навчальної цінності [15, 37].

Нарешті, функція формування навчального контенту F_{cnt} відповідає за те, у якій формі буде подано теоретичний матеріал, підказки та пояснення. Для задач, пов’язаних із перевіркою доведень, ця функція може генерувати покроковий HTML-матеріал на основі формально верифікованого артефакту, не змінюючи при цьому множину доведених тверджень. Таким чином, у системі розділяються поняття істинності і дидактичного подання: істинність забезпечує формальна верифікація, а подання забезпечує функція F_{cnt}.

Узагальнено функціональний цикл системи можна подати як композицію функцій:

$$(K_t, H_t) \to F_{sel} \to q_t \to F_{gen} \to a_t \to F_{eval} \to F_{diag} \to F_{ver} \to F_{upd} \to K_{t+1} \to F_{mot}$$

У цій композиції формування наступної задачі залежить від попереднього стану, а сам стан змінюється під впливом завершеної навчальної дії. Таким

чином, функціональна модель має замкнений характер і безпосередньо реалізує ідею керування навчанням на основі діагностики.

$$(K_t, H_t) \to F_{sel} \to q_t \to F_{gen} \to a_t \to F_{eval} \to F_{diag} \to F_{ver} \to F_{upd} \to K_{t+1} \to F_{mot} \quad (2.2)$$

Стан системи на t-му кроці доцільно описувати впорядкованим кортежем

$$X_t = \langle K_t, B_t, M_t, H_t \rangle,$$

де K_t – стан знань, B_t – поведінковий профіль, M_t – мотиваційний стан, H_t – історія взаємодії. Тоді загальна динаміка функціонування може бути записана як

$$X_{t+1} = \Phi(X_t, q_t, a_t, d_t),$$

де d_t – діагностичний сигнал, а Φ – узагальнений оператор оновлення функціонального стану. Таке представлення показує, що система працює не лише з правильністю відповіді, а з багатовимірним станом навчального процесу.

$$X_t = \langle K_t, B_t, M_t, H_t \rangle \quad (2.3)$$

$$X_{t+1} = \Phi(X_t, q_t, a_t, d_t) \quad (2.4)$$

Функціональна модель дозволяє чітко визначити місця інтеграції окремих дослідницьких результатів у єдину інформаційну технологію. Так, баєсівська діагностика відповідає за функцію F_{upd}, структурний аналіз виразів – за частину F_{diag}, Lean/Mathlib – за F_{ver} і F_{cnt}, генератор параметризованих задач – за F_{gen}, а сенсо-економічний модуль – за F_{mot}. Завдяки такому розподілу система залишається модульною, але при цьому кожний новий метод має чітко визначене місце в загальному функціональному контурі.

Отже, функціональна модель системи підтримки навчання точним наукам фіксує її як замкнений цикл перетворення навчальних дій у нові керуючі рішення. Саме ця модель слугує безпосереднім переходом до математичної формалізації, де кожна із функцій набуває вигляду відображень, операторів, предикатів і алгоритмів.

Рисунок 2.5 – Функціональна схема системи як замкнений навчальний цикл: вибір задачі → розв'язання → перевірка → діагностика → оновлення знань → мотиваційний сигнал → наступна задача

2.5 Математична модель системи

Математична модель системи підтримки навчання точним наукам повинна забезпечувати формальний опис навчальних задач, стану знань студента, правил оновлення цього стану, механізмів виявлення та класифікації помилок, а також базових величин, що використовуються в адаптивному доборі задач і мотиваційному регулюванні. На відміну від суто вербального чи структурного опису, математична модель повинна допускати аналітичне дослідження, алгоритмічну реалізацію та перевірку коректності окремих рішень.

У даному підрозділі математична модель системи будується поетапно: спочатку формалізуються навчальні задачі як об’єкти предметної області, далі вводиться формалізована діагностична модель стану знань студента, після чого описуються пов’язані з нею моделі мотивації, простору параметризованих задач і, нарешті, теоретична модель квантового прискорення окремих пошукових

процедур. Така послідовність відповідає логіці переходу від змісту навчального впливу до механізмів його оптимізації.

2.5.1 Формалізація навчальних задач точних наук

Формалізація навчальних задач потрібна для того, щоб система працювала не з неструктурованими текстовими умовами, а з об'єктами, для яких визначено необхідні знання, вхідні параметри, тип відповіді та процедуру перевірки. Для цього введемо загальну модель задачі у вигляді кортежу

$$Q = (R, Given, Input, Answer, Algorithm),$$

де R – множина правил, понять і теоретичних положень, що перевіряються задачею; $Given$ – множина можливих постановок однієї й тієї самої задачі; $Input$ – множина вхідних даних; $Answer$ – множина допустимих відповідей; $Algorithm$ – множина процедур, за допомогою яких може бути отримано правильний результат [4, 5].

$$Q = (R, Given, Input, Answer, Algorithm) \quad (2.5)$$

Такий запис є універсальним для широкого класу задач точних наук. Він дозволяє розглядати умову задачі окремо від правил і понять, необхідних для її розв'язання, а також відокремити процедуру обчислення або доведення від конкретного тексту постановки. Це, у свою чергу, відкриває можливість незалежної генерації різних словесних формулювань однієї задачі, перевірки коректності параметрів та побудови кількох допустимих способів розв'язання.

Множина R виконує особливу роль, оскільки саме вона пов'язує задачу з моделлю знань студента. Якщо задача перевіряє володіння кількома поняттями, то кожне з них повинно бути явно відображене у відповідних елементах R. У загальному випадку [7, 8, 9]

$$R = \{r_1, r_2, \dots, r_n\} \quad (2.6)$$

де r_i – окремий елемент знань або вмінь: означення, правило перетворення, алгоритмічний крок, схема доведення або типовий структурний шаблон. Аналогічно множина можливих постановок задачі задається як

$$Given = \{g_1, g_2, \dots, g_l\} \quad (2.7)$$

Множина Given дозволяє інтерпретувати одну й ту саму математичну задачу як сімейство мовних реалізацій. Ця властивість є важливою для сучасних систем підтримки навчання, оскільки дає змогу комбінувати алгоритмічну точність моделі задачі з лексичною гнучкістю генеративних моделей штучного інтелекту. При цьому зміна текстового формулювання не повинна змінювати зміст задачі і набір правил R, що перевіряються.

Множина Input у більшості випадків складається з числових параметрів або їхніх комбінацій:

$$Input = \{inp_1, inp_2, \dots, inp_m\} \quad (2.8)$$

Множина Answer задає допустимий тип відповіді – число, формулу, логічне твердження, структуру дерева виразу, proof term або програмний код [17, 18, 22]:

$$Answer = \{ans_1, ans_2, \dots, ans_k\} \quad (2.9)$$

Нарешті, множина Algorithm визначає способи отримання відповіді:

$$Algorithm = \{A_1, A_2, \dots, A_s\} \quad (2.10)$$

де кожна A_j може бути реалізована як у вигляді процедур сучасних мов програмування, так і у вигляді формально перевірюваних перетворень у системі доказів. У межах точних наук множина Algorithm відіграє подвійну роль: з одного боку, вона задає спосіб обчислення або побудови відповіді, а з іншого – може бути об’єктом навчання як такий.

2.5.1.1 Складні задачі на побудову алгоритмів, виведення та доведення

Складними назвемо задачі, для яких недостатньо застосувати один локальний алгоритм або правило. Їх розв’язання потребує побудови багатокрокової стратегії, використання кількох взаємопов’язаних знань і, часто, формального обґрунтування коректності кожного кроку. До цього класу належать задачі на побудову алгоритмів, задачі на виведення формул, задачі на

доведення теорем, а також нетривіальні задачі на моделювання, оптимізацію або аналіз структури математичного об’єкта [4, 5, 22].

На відміну від простих задач, тут не можна вважати, що множина Algorithm містить один або кілька готових алгоритмів, які лише потрібно застосувати. У складних задачах сам алгоритм або доведення є об’єктом побудови. Тому для цього класу задач доцільно розширити загальну модель, ввівши об’єкти стратегії та формальної перевірки. Узагальнений запис має вигляд

$$ComplexTask = (R, Given, Input, Answer, Strategy, Verifier) \quad (2.16)$$

де $Strategy$ задає послідовність міркувань, перетворень або кроків побудови алгоритму, а $Verifier$ – процедуру встановлення коректності цієї послідовності. Якщо мова йде про задачу на доведення, $Strategy$ може бути представлена як трасою доказових станів; якщо мова йде про алгоритм, $Strategy$ може бути програмною або псевдокодовою схемою, специфікація якої підлягає перевірці.

Для задач на доведення доцільно ввести додатковий об’єкт P – формальний або напівформальний доказ:

$$ProofTask = (R, Given, Statement, P, Verifier) \quad (2.17)$$

Тут $Statement$ – твердження, яке підлягає доведенню, а $Verifier(P) = True$ означає, що доведення приймається системою як коректне. У традиційному навчанні роль $Verifier$ виконує викладач. У пропонованій інформаційній технології роль остаточного арбітра для формалізованого фрагмента може бути передана proof assistant, наприклад Lean. Такий підхід переносить контроль істинності від людини до малого довіреного ядра системи, але не усуває потреби у дружньому до користувача інтерфейсі й поетапному педагогічному поданні.

Для задач на побудову алгоритму зручно використовувати модель

$$AlgTask = (R, Given, Input, Output, Spec, A, Verifier) \quad (2.18)$$

де $Spec$ задає формальну або напівформальну специфікацію властивостей алгоритму A. У найпростішому варіанті це може бути предикат часткової коректності $Pre(x) \Rightarrow Post\big(A(x)\big)$; у більш складному – сукупність інваріантів, умов завершення, властивостей відмовостійкості або ресурсних обмежень. Таким чином, у складній задачі система перевіряє не лише результат виконання алгоритму на вибірці тестів, а відповідність алгоритму його логічній специфікації.

Складні задачі істотно пов'язані з поняттям формалізовності. Не кожна математична задача в поточних обчислювальних і бібліотечних межах може бути повністю автоматично перевірена. Тому корисно ввести градацію задач за рівнем формалізовності: від повністю автоматизовних до таких, що потребують суттєвої взаємодії студента або викладача з системою. Для потреб моделі позначимо рівень формалізовності задачі через [22]

$$F(Q) \in \{F0, F1, F2, F3, F4\} \tag{2.19}$$

де $F0$ відповідає задачам, що повністю перевіряються звичайними алгоритмами; $F1$–$F2$ – задачам, для яких доступна автоматизована або напівавтоматизована формальна перевірка окремих кроків; $F3$ – задачам, що вимагають розгорнутого доказового супроводу; $F4$ – задачам, які в межах поточних ресурсів не можуть бути повністю формалізовані та служать об'єктом лише часткової підтримки. Така градація дозволяє зв'язати зміст навчальної задачі з реальними можливостями її машинної перевірки.

Для складних задач особливого значення набуває аналіз проміжних станів розв'язання. Нехай σ_t – стан розв'язання на t-му кроці, тоді стратегія побудови алгоритму або доведення може бути подана як послідовність

$$\Pi = \langle \sigma_0, \sigma_1, \dots, \sigma_m \rangle \tag{2.20}$$

Правильність кінцевого результату в цьому випадку залежить від допустимості переходів між сусідніми станами. Якщо $next(\sigma_t) = \sigma_{t+1}$ відповідає дозволеному правилу перетворення, то крок вважається коректним. Для proof assistant це означає можливість зіставлення людського кроку з

формальною тактикою або proof term; для алгоритмічних задач – відповідність кроку специфікації чи інваріанту.

У межах навчання точним наукам складні задачі є ключовими, оскільки саме вони протидіють примітивізації освіти. Якщо навчання обмежується лише простими задачами, студент навчається відтворювати процедури. Якщо ж у систему вбудовано складні задачі на доведення, виведення та побудову алгоритмів, то вона починає працювати з концептуальним мисленням. Тому формальна модель складних задач не є допоміжним додатком до моделі простих задач, а необхідною умовою побудови повноцінної системи підтримки навчання точним наукам.

Рисунок 2.6 – Схема класифікації навчальних задач на прості та складні з прикладами для кожного класу

2.5.2 Діагностичні моделі встановлення причин помилкових відповідей

У системах підтримки навчання точним наукам важливо не лише встановити факт правильної або неправильної відповіді, а й визначити причину

помилки. У традиційних комп'ютерних навчальних системах оцінювання часто зводиться до бінарного результату: відповідь правильна або неправильна. Такий підхід є недостатнім для навчання точним наукам, оскільки одна й та сама неправильна відповідь може бути наслідком різних причин: помилки в обчисленнях, нерозуміння означення, неправильного застосування правила, порушення умов теореми або помилки у логічній структурі міркування. [7, 8, 9].

У межах цієї роботи діагностична модель розглядається як формалізована конструкція, що дозволяє за спостережуваними непрямими ознаками встановити ймовірну першопричину помилкової відповіді. Такими ознаками можуть бути: конкретне значення неправильної відповіді, спосіб її запису, структура математичного виразу, послідовність проміжних кроків, тип перетворення, час відповіді, кількість повторних спроб або звернень до підказок.

Таким чином, діагностична модель не є лише моделлю стану знань студента. Вона є механізмом інтерпретації помилки, що встановлює зв'язок між наслідком і можливою причиною. Якщо неправильна відповідь є спостережуваним наслідком, то діагностична модель дозволяє висунути гіпотезу щодо прихованої причини цієї помилки.

2.5.2.1 Компоненти знань та їх представлення

Нехай $S = \{s_1, s_2, \dots, s_n\}$ – множина компонентів знань, які відстежуються в системі. Кожний компонент s_i відповідає окремій змістовній одиниці: правилу, процедурі, схемі перетворення, типу аналізу, елементу теорії або алгоритмічному прийому. Тоді стан знань студента у момент часу t можна представити вектором імовірностей засвоєння

$$K_t = \big(P_t(L_1), P_t(L_2), \dots, P_t(L_n)\big) \in [0,1]^n \qquad (2.21)$$

де $P_t(L_i)$ – поточна оцінка ймовірності того, що компонент s_i засвоєний студентом до моменту t. Таке подання природно узгоджується з баєсівською інтерпретацією knowledge tracing і дозволяє надалі оновлювати кожну компоненту окремо [7, 8, 9].

Оскільки одна навчальна задача часто вимагає одночасного використання кількох взаємопов’язаних умінь, для кожної задачі Q вводиться відображення на підмножину компонентів знань:

$$\Lambda(Q) = \{s_{i_1}, s_{i_2}, \dots, s_{i_k}\} \subseteq S \tag{2.22}$$

Тут $\Lambda(Q)$ є набором компонентів, задіяних у розв’язанні задачі Q. Наприклад, для задачі на розв’язання диференціального рівняння другого порядку такими компонентами можуть бути: побудова характеристичного рівняння, аналіз типу коренів, запис загального розв’язку, підстановка початкових умов, елементарні алгебраїчні перетворення.

Щоб отримати узагальнену оцінку того, наскільки студент готовий до задачі Q, вводиться функція агрегування покомпонентних імовірностей:

$$P_t(Q) = Agg(\{P_t(L_i): s_i \in \Lambda(Q)\}) \tag{2.23}$$

Функція Agg може мати різну природу. Якщо задача потребує одночасної надійної наявності всіх компонентів, доцільним є мінімум; якщо окремі компоненти можуть частково компенсувати один одного – середнє значення; якщо деякі компоненти є важливішими – зважене середнє:

$$P_t(Q) = \sum_{s_i \in \Lambda(Q)} \omega_i^Q \cdot P_t(L_i), \Sigma\omega_i^Q = 1 \tag{2.24}$$

Таке представлення дозволяє перейти від оцінювання «задача/не задача» до оцінювання готовності до цілих класів задач, що є суттєво важливим для зовнішнього циклу навчання.

Однак для задач точних наук однієї лише покомпонентної ймовірнісної моделі недостатньо. Студент може дати неправильну відповідь не через повну відсутність концептуального розуміння, а через локальну структурну помилку: пропущений оператор, неправильний знак, некоректне спрощення, неправильну ідентифікацію типу коренів, помилковий вибір наступного кроку доведення. Для виявлення таких помилок у модель вводиться структурне подання відповіді у формі дерева виразу [23, 25].

Нехай a_t – відповідь студента на кроці t. Тоді після синтаксичного аналізу будується дерево

$$T_t = Tree(a_t) = (V_t, E_t, \lambda_t) \quad (2.25)$$

де V_t – множина вузлів (операндів та операцій), $E_t \subseteq V_t \times V_t$ – множина спрямованих зв'язків «батько–нащадок», $\lambda_t: V_t \to \Sigma$ – функція міток, яка ставить у відповідність вузлам символи з алфавіту Σ, наприклад «+», «−», «*», «exp», «sin», константи та змінні.

Порівнюючи студентське дерево T_t з еталонним деревом $T*_t$, можна визначити перше структурне розходження. Позначимо його індекс через δ:

$$\delta(T_t, T*_t) = min\{j: \lambda_t(u_j) \neq \lambda*_t(v_j)\} \quad (2.26)$$

Ця величина локалізує перший вузол, на якому розв'язання студента перестає узгоджуватися з еталонною структурою. На практиці це дозволяє ідентифікувати не лише помилковий результат, а й місце, де було зруйновано правильну лінію міркувань.

Нехай $DM = \{dm_1, dm_2, \ldots, dm_m\}$ – множина діагностичних моделей типових помилок. Тоді функція класифікації структурної помилки може бути задана як

$$\phi: (T_t, \delta) \to DM \quad (2.27)$$

де φ відображає дерево відповіді та локалізацію розходження у конкретний клас помилки. Кожен клас dm_k пов'язаний із підмножиною компонентів знань, які ця помилка активує як проблемні. Таким чином, вводиться відображення

$$\psi: DM \to 2^S \quad (2.28)$$

що ставить у відповідність діагностичній моделі множину компонентів знань, стан яких повинен бути оновлений як наслідок помилкового кроку.

Окрім предметного сигналу, система відстежує і поведінковий сигнал. Нехай $action_t$ – атомарна дія студента на кроці t. Їй відповідає вектор поведінкових ознак

$$B_t = [\tau_t, h_t, r_t, p_t, d_t, v_t] \quad (2.29)$$

де τ_t – нормалізований час відповіді, h_t – кількість запитаних підказок, r_t – кількість повторних спроб, p_t – тривалість паузи до відповіді, d_t – кількість змін введеного об'єкта, v_t – варіативність зусиль або інші поведінкові ознаки. Для одержання більш стійкої оцінки поведінкової якості використовується усереднений профіль

$$\bar{E}_t = (1/t) \sum_{k=1}^{t} B_k \quad (2.30)$$

Норма цього вектора інтерпретується як узагальнений поведінковий бал залученості. У запропонованій моделі він не замінює знання, а доповнює його, дозволяючи відрізнити випадкове вгадування від змістовної, хоч і не завжди безпомилкової роботи.

Для інтеграції покомпонентної ймовірності знань і якості поведінкового профілю вводиться гібридний бал компонента

$$HSS_t(s_i) = \alpha \cdot P_t(L_i) + \beta \cdot \|\bar{E}_t\|, \alpha, \beta \in [0,1], \alpha + \beta = 1 \quad (2.31)$$

Величина HSS_t не є прямою заміною предметної оцінки засвоєння. Її доцільно інтерпретувати як інтегральний показник навчальної цінності поточної взаємодії: наскільки студент не лише знає компонент, а й опрацьовує його у спосіб, що свідчить про осмислену залученість і потенціал переносу знання. Саме ця величина надалі використовується в мотиваційній моделі та при побудові окремих типів рекомендацій.

Таблиця 2.4 – Приклади компонентів знань та поведінкових ознак

Тип елемента	Позначення	Зміст
Компонент знань	$\boldsymbol{s_i}$	Окреме правило, поняття, уміння або підзадача
Задача	Q	Навчальна задача, що активує підмножину $\boldsymbol{\Lambda(Q)}$
Діагностична модель помилки	$\boldsymbol{dm_k}$	Типова помилка або група помилок
Поведінкова ознака	$\boldsymbol{\tau_t}$	Час відповіді
Поведінкова ознака	$\boldsymbol{h_t}$	Кількість підказок
Поведінкова ознака	$\boldsymbol{r_t}$	Кількість повторних спроб
Поведінкова ознака	$\boldsymbol{p_t}$	Пауза перед відповіддю
Поведінкова ознака	$\boldsymbol{d_t}$	Кількість змін введення

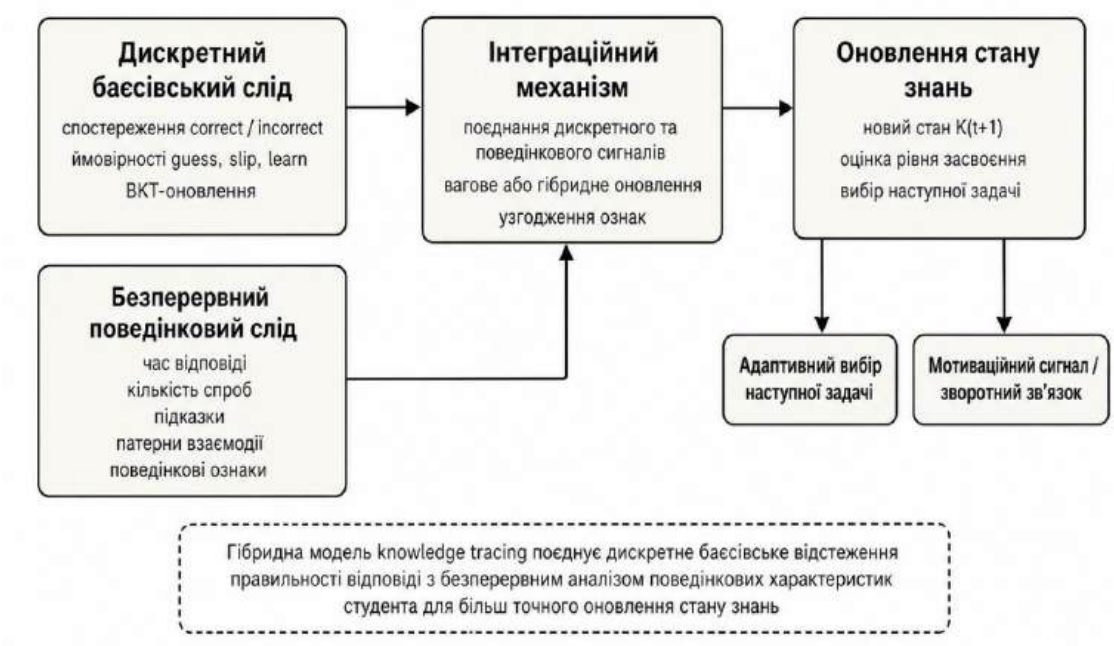


Рисунок 2.7 – Гібридна модель knowledge tracing: дискретний баєсівський слід, безперервний поведінковий слід і оновлення стану знань

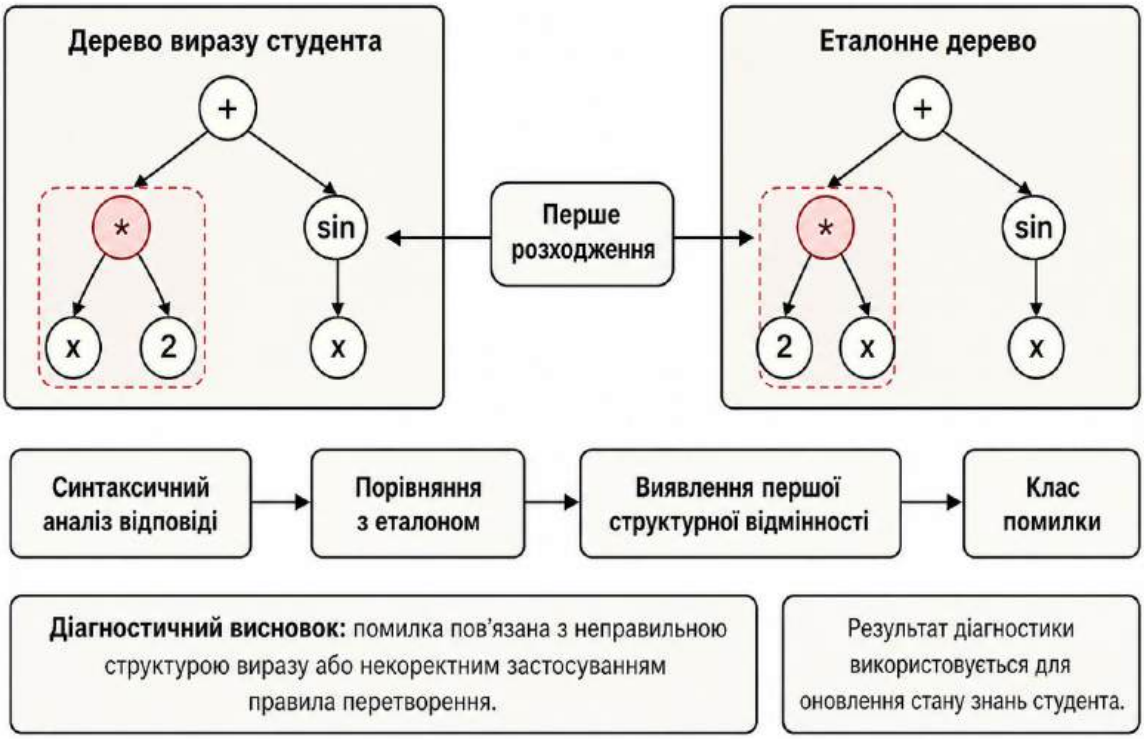


Рисунок 2.8 – Схема структурної діагностики помилок через дерево виразу та перше розходження з еталоном

2.5.2.2 Баєсівська модель оновлення стану знань

Нехай для кожного компонента знань s_i задано чотири базові параметри баєсівської моделі: $P(L_i^0)$ – початкова ймовірність засвоєння компонента; T_i – імовірність переходу від незасвоєного до засвоєного стану після навчальної взаємодії; S_i – параметр slip, що характеризує ймовірність помилкової відповіді за умови засвоєного компонента; G_i – параметр guess, що характеризує ймовірність правильної відповіді при незасвоєному компоненті. Ці параметри є стандартними для Bayesian Knowledge Tracing і забезпечують інтерпретованість оновлення моделі [7, 28, 29].

Позначимо через $O_t \in \{0,1\}$ спостереження на кроці t, де $O_t = 1$ означає правильний крок або відповідь, а $O_t = 0$ – помилковий. Тоді для одного компонента s_i класична баєсівська модель оновлення має вигляд. Якщо спостережено правильну відповідь, апостеріорна ймовірність засвоєння до навчального переходу визначається як

$$P_t(L_i|O_t = 1) = \big((1 - S_i) \cdot P_t(L_i)\big) / \Big((1 - S_i) \cdot P_t(L_i) + G_i \cdot \big(1 - P_t(L_i)\big)\Big) \quad (2.32)$$

Після цього враховується можливість навчального переходу під впливом виконаної дії:

$$P_{t+1}(L_i) = P_t(L_i|O_t = 1) + \big(1 - P_t(L_i|O_t = 1)\big) \cdot T_i \quad (2.33)$$

Якщо ж спостережено неправильну відповідь, апостеріорна ймовірність засвоєння обчислюється як

$$P_t(L_i|O_t = 0) = \big(S_i \cdot P_t(L_i)\big) / \Big(S_i \cdot P_t(L_i) + (1 - G_i) \cdot \big(1 - P_t(L_i)\big)\Big) \quad (2.34)$$

а потім також застосовується перехід навчання

$$P_{t+1}(L_i) = P_t(L_i|O_t = 0) + \big(1 - P_t(L_i|O_t = 0)\big) \cdot T_i \quad (2.35)$$

Наведені співвідношення є базовим ядром діагностичної моделі. Їх перевага полягає в прозорості та інтерпретованості: викладач може явно задавати початкові оцінки, імовірність навчання, параметри slip і guess для кожного компонента або їх групи. Однак у навчанні точним наукам цього недостатньо, оскільки одна навчальна задача часто активує кілька компонентів знань одночасно [7, 9, 25].

Нехай задача Q_t пов’язана з підмножиною компонентів $\Lambda(Q_t)$. Тоді оновлення повинно виконуватися не для одного компонента, а для всіх $s_i \in \Lambda(Q_t)$. Найпростіший варіант – незалежне оновлення всіх залучених компонентів. Проте більш адекватною є зважена схема, коли внесок задачі у зміну конкретного компонента визначається коефіцієнтом ω_i^Q. Тоді

$$P_{t+1}(L_i) = U_i\big(P_t(L_i), O_t, T_i, S_i, G_i, \omega_i^Q\big), s_i \in \Lambda(Q_t) \quad (2.36)$$

де U_i позначає відповідне баєсівське оновлення з урахуванням ваги компонента в задачі. Така схема є особливо корисною для задач на вищу математику, де

помилка в допоміжному алгебраїчному перетворенні не повинна з тим самим ступенем знижувати оцінку засвоєння всієї теми, що й концептуальна помилка в побудові розв'язку.

Щоб врахувати структурно локалізовані помилки, використаємо відображення $\psi: DM \to 2^S$, введене в попередньому підрозділі. Якщо дерево виразу породило діагностичну модель dm_k, то оновлення виконується лише для компонентів $\psi(dm_k)$, а не для всіх компонентів задачі $\Lambda(Q_t)$:

$$A_t = \psi(dm_k) \subseteq \Lambda(Q_t) \tag{2.37}$$

Тоді баєсівське оновлення уточнюється таким чином: для компонентів із множини A_t спостереження інтерпретується як failure, для решти компонентів – або не виконується оновлення, або виконується ослаблене оновлення з меншим штрафом. Формально це можна записати як

$$P_{t+1}(L_i) = U_i\big(P_t(L_i), O_t^i\big), O_t^i = 0 \text{if} s_i \in A_t \tag{2.38}$$

де для $s_i \notin A_t$ значення O_t^i може бути задане як нейтральне або скориговане з меншим коефіцієнтом впливу. Такий підхід реалізує принцип локалізованої діагностики: система не «каже», що студент не знає все, а уточнює, в якому саме фрагменті відбулося порушення.

У багатьох реальних ситуаціях правильність або неправильність кроку має доповнюватися даними про характер взаємодії студента. Нехай $\|\bar{E}_t\|$ – агрегована поведінкова оцінка. Тоді для уникнення надмірно жорстких штрафів за осмислені, але помилкові спроби можна вводити модифіковане спостереження

$$\bar{O}_t^i = \gamma_i \cdot O_t^i + (1 - \gamma_i) \cdot f(\|\bar{E}_t\|) \tag{2.39}$$

де f($\|\bar{E}_t\|$) – нормалізована функція поведінкового профілю, а $\gamma_i \in [0,1]$ визначає баланс між предметною правильністю та якістю навчальної взаємодії. Якщо студент робить помилку, але демонструє високий рівень змістовної залученості, модель може зменшувати силу негативного оновлення. Це не означає, що помилка вважається правильною; йдеться лише про те, що діагностика стає більш педагогічно чутливою.

Гібридність моделі проявляється у поєднанні двох траєкторій: дискретної та неперервної. Дискретна траєкторія задається баєсівськими оновленнями стану знань і структурною діагностикою. Неперервна траєкторія задається часовими та поведінковими ознаками взаємодії. Їх поєднання може бути формалізоване як оператор злиття [8, 9, 11]

$$K_{t+1} = Fusion(BKT(K_t, O_t, DM_t), \bar{E}_t) \tag{2.40}$$

де BKT(K_t, O_t, DM_t) означає результат дискретного баєсівського оновлення з урахуванням структурної помилки DM_t, а $Fusion$ – оператор, який коригує або збагачує знання поведінковою інформацією. У найпростішому випадку роль $Fusion$ відіграє гібридний бал HSS; у складніших реалізаціях він може бути замінений нейромережевою або квантовою моделлю траєкторії, однак базова система при цьому зберігає інтерпретованість.

Окремий практичний інтерес становлять багатокомпонентні задачі, для яких одна помилка може стосуватися лише частини активованих навичок. У такому випадку агрегована ймовірність готовності до задачі після спостереження може бути записана як

$$P_{t+1}(Q_t) = Agg(\{P_{t+1}(L_i): s_i \in \Lambda(Q_t)\}) \tag{2.41}$$

Якщо ж діагностична модель вказує на концептуальне відновлення після помилки, важливо оцінити не лише нове значення $P_{t+1}(L_i)$, а й приріст знання

$$\Delta P_t(L_i) = P_{t+1}(L_i) - P_t(L_i) \tag{2.42}$$

Саме величина $\Delta P_t(L_i)$ надалі використовується в мотиваційній моделі для визначення подій типу recovery sense. Вона фіксує не просто факт правильності, а позитивну динаміку оволодіння знанням.

Таким чином, баєсівська модель оновлення стану знань у системі підтримки навчання точним наукам має багаторівневий характер. На базовому рівні вона реалізує класичний BKT. На другому рівні вона враховує багатокомпонентну структуру задач. На третьому рівні вона інтегрує локалізовані структурні помилки, а на четвертому – поєднує знання з

поведінковим профілем взаємодії. Саме така багаторівнева побудова робить діагностичну модель придатною для задач вищої математики, де поверхневе оцінювання правильності не дає достатньої педагогічної інформації [7, 8, 9, 25].

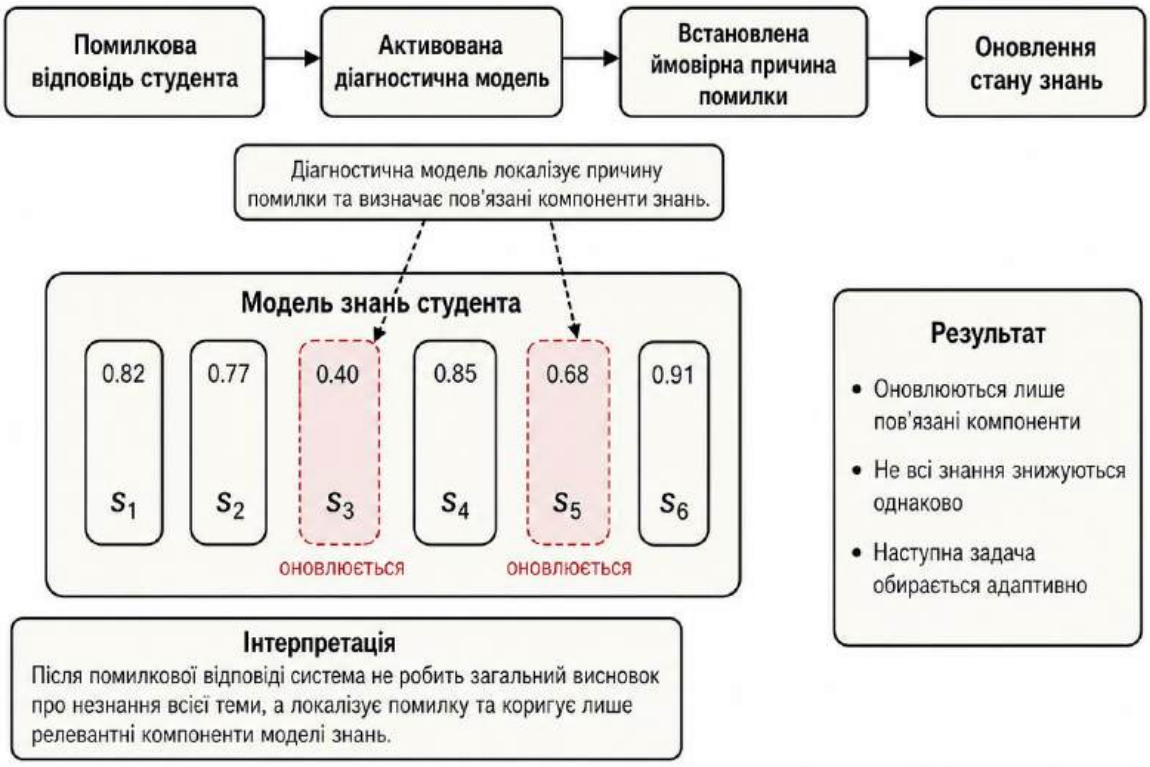


Рисунок 2.9 – Стан моделі знань студента з активованою діагностичною моделлю після помилкової відповіді

2.6 Модель мотиваційного регулювання навчального процесу

Мотиваційне регулювання в системі підтримки навчання точним наукам розглядається не як зовнішній декоративний шар, а як повноцінний елемент керованого навчального процесу. Його завдання полягає в узгодженні навчальних досягнень із такими стимулами, які підтримують не лише короткострокову активність, а й довготривалу внутрішню мотивацію до оволодіння складними предметними структурами. У традиційних освітніх системах мотиваційний механізм здебільшого зводиться до оцінки, рейтингу або формальної відзнаки. У більшості гейміфікованих платформ зовнішнє підкріплення реалізується у формі балів, значків, рівнів та таблиць лідерів. Однак такі стимули часто не узгоджені з реальним когнітивним поступом студента і

можуть призводити до ефекту «полювання за балами» замість осмисленого навчання [13, 14, 15].

У пропонованій системі мотиваційна модель повинна спиратися на діагностичні сигнали про реальний характер навчальної діяльності. Це означає, що об'єктом винагородження стає не лише правильна відповідь як така, а значущі прояви інтелектуальної роботи: упізнавання структурного шаблону, самостійне виправлення помилки, наполегливе просування без надмірної кількості підказок, виявлення нестандартного, але коректного способу розв'язання, внесок у пояснювальний шар або допомогу іншим студентам. Такий підхід далі називатимемо сенсо-економічним, оскільки він пов'язує навчальну винагороду з освітнім смислом виконаної дії [12, 14, 37].

2.6.1. Система цифрових винагород

Нехай SENS – множина всіх типів значущих навчальних подій, що можуть бути виявлені системою. У загальному вигляді [15, 37]

$$SENS = \{sens_{str}, sens_{eff}, sens_{rec}, sens_{soc}, sens_{cre}, sens_{ref}\} \quad (2.43)$$

де $sens_{str}$ – structural sense (структурне впізнавання), $sens_{eff}$ – effort sense (зусилля і наполегливість), $sens_{rec}$ – recovery sense (відновлення після помилки), $sens_{soc}$ – social sense (соціально значущий внесок), $sens_{cre}$ – creative sense (коректний нестандартний метод), $sens_{ref}$ – reflective sense (самокорекція і рефлексивний перегляд дії).

Кожний елемент множини SENS інтерпретується як подія, що має власний тип, часову мітку, джерело, пов'язані компоненти знань і числове значення освітньої цінності. Якщо $sens_t$ – подія у момент t, то її можна подати кортежем

$$sens_t = \langle type_t, time_t, source_t, A_t, meta_t \rangle \quad (2.44)$$

де $type_t$ визначає тип значущої події, $source_t$ вказує на навчальну дію або її результат, A_t – множину пов'язаних компонентів знань, а $meta_t$ – додаткові метадані, наприклад складність задачі, величину приросту знання, кількість самостійних кроків або відсутність підказок.

Для відображення події у цифрову цінність вводиться функція вартості

$$Value: SENS \to R_{+} \tag{2.45}$$

У найпростішій формі ця функція може бути задана як

$$Value(sens_t) = w(type_t) \cdot \rho(sens_t, HSS_t, \Delta P_t, diff_t) \tag{2.46}$$

де $w(type_t)$ – базова вага типу події, $diff_t$ – складність задачі, ΔP_t – приріст імовірності засвоєння пов'язаного компонента знань, а ρ – контекстний множник, який коригує оцінку залежно від педагогічної ситуації. Наприклад, для recovery sense значення ρ може зростати, якщо студент виправив концептуальну помилку без підказки, а для effort sense – якщо тривалий час роботи супроводжувався високою когнітивною залученістю, а не пасивним очікуванням.

Щоб забезпечити формалізоване зберігання та подальшу циркуляцію таких винагород, вводиться єдиний освітній токен SenseCoin. Баланс студента у момент часу t позначимо через C_t. Тоді оновлення балансу після нарахування винагород і, за потреби, списання частини токенів для отримання освітніх сервісів задається співвідношенням [16, 17, 37]

$$C_{t+1} = C_t + \Sigma Value(sens_t) - Burn_t \tag{2.47}$$

де $Burn_t$ позначає кількість токенів, що списується на отримання певного сервісу – додаткової консультації, доступу до поглибленого модуля, участі у спеціальному проєкті, відкриття розширеного набору задач тощо. Важливо підкреслити, що SenseCoin у базовій моделі не є фінансовим активом у класичному сенсі. Це освітній токен, що функціонує всередині навчального середовища та відображає педагогічно значущі досягнення.

Щоб винагороди не перетворювалися на непрозору внутрішню метрику системи, доцільно використовувати керований журнал подій або permissioned blockchain. У цьому випадку кожне нарахування токена супроводжується записом

$$tx_t = \langle id_t, user_t, sens_t, Value(sens_t), sign_t \rangle \quad (2.48)$$

де $sign_t$ – цифровий підпис або інший механізм підтвердження походження транзакції. Такий запис забезпечує перевірюваність, незмінність історії та можливість аудиту системи винагород. Університет або освітня установа можуть розгорнути приватну Ethereum-сумісну мережу з алгоритмом консенсусу Proof-of-Authority, що дозволяє усунути комісії публічних мереж і зберегти повний інституційний контроль над токеном.

Структурно система цифрових винагород повинна включати щонайменше три рівні: діагностичний, який виявляє значущу подію; оракульний, який перетворює педагогічний сигнал у команду нарахування; блокчейн-рівень або журнал незмінних записів, який виконує власне реєстрацію токену. Таке розмежування важливе, оскільки логіка оцінювання смислу належить навчальній системі, а логіка незмінності та обліку – інфраструктурному шару.

Таблиця 2.5 – Типи значущих навчальних подій у сенсо-економічній моделі

Тип події	Зміст	Типовий тригер
Structural sense	Упізнавання глибокої структури або схеми	Суттєве зростання HSS після кроку розпізнавання
Effort sense	Наполеглива осмислена робота	Тривалий час, кілька спроб, мінімум підказок
Recovery sense	Самостійне виправлення концептуальної помилки	Позитивний ΔP після попередньої помилки
Social sense	Допомога іншому студенту або внесок у базу пояснень	Підтверджене використання іншими
Creative sense	Коректний нестандартний метод	Верифіковане альтернативне рішення
Reflective sense	Самокорекція та перегляд некоректної дії	Відкликання помилкового кроку з подальшим правильним

Рисунок 2.10 – Порівняльна схема традиційної, гейміфікаційної та сенсо-економічної мотиваційної моделі

2.6.1 Інтеграція мотиваційних механізмів зі станом знань

Інтеграція мотиваційного механізму зі станом знань є принциповою відмінністю запропонованої моделі від традиційної гейміфікації. У звичайних системах бали нараховуються за факт виконання дії: проходження тесту, відвідування заняття, завершення модулю. У нашому випадку мотиваційний сигнал виникає лише тоді, коли навчальна дія має формально підтверджений зв'язок зі зміною когнітивного стану або з педагогічно значущим внеском. Отже, мотивація не є окремим конвеєром балів, а функціонує як другий рівень інтерпретації змін у стані знань [15, 37].

Нехай для компонента s_i у момент t розраховано приріст знання

$$\Delta P_t(L_i) = P_{t+1}(L_i) - P_t(L_i) \quad (2.49)$$

Тоді базова умова виникнення події типу recovery sense може бути подана як

$$Trigger_{rec}(s_i, t) = 1, \text{if} O_t = 1 \wedge O_{t-1} = 0 \wedge \Delta P_t(L_i) \geq \theta_r ec \quad (2.50)$$

де θ_{rec} – поріг, після перевищення якого приріст знання вважається достатнім для винагородження відновлення після помилки. Аналогічно подія structural sense може породжуватися тоді, коли структурний аналіз відповіді зафіксував правильне розпізнавання схеми, а відповідний HSS перевищив поріг

$$Trigger_{str}(s_i, t) = 1, \text{if } HSS_t(s_i) \geq \theta_{str} \quad (2.51)$$

Effort sense може бути пов’язаний з поведінковими показниками, а не лише з фактами правильності. Наприклад, якщо студент не досяг одразу правильної відповіді, але працював осмислено і без зловживання підказками, подія effort sense може породжуватися за правилом

$$Trigger_eff(t) = 1, \text{if} \|\bar{E}_t\| \geq \theta_{eff} \wedge h_t \leq h_{max} \quad (2.52)$$

Загальний індикатор мотиваційної активації можна описати як об’єднання булевих тригерів

$$Trigger_t = Trigger_{rec} \vee Trigger_{str} \vee Trigger_{eff} \vee Trigger_{cre} \vee Trigger_{soc} \vee Trigger_{ref} \quad (2.53)$$

Після активації $Trigger_t$ викликається функція оцінювання Value($sens_t$), а її результат додається до балансу токенів. Завдяки такій схемі мотиваційний модуль інтегрується зі станом знань без втрати педагогічної інтерпретованості: кожне нарахування має пояснення, яке можна показати студенту мовою змістовного зворотного зв’язку.

Поведінковий і знаннєвий контури можуть бути інтегровані також на рівні мотиваційного стану M_t. У найпростішому варіанті

$$M_t = \langle C_t, R_t, HSS_t \rangle \quad (2.54)$$

де C_t – поточний баланс токенів, R_t – журнал винагород, HSS_t – інтегральна оцінка якості поточної навчальної взаємодії. Тоді розширений стан системи студента набуває вигляду

$$X_t = \langle K_t, B_t, M_t, H_t \rangle \quad (2.55)$$

що узгоджується з функціональною моделлю розділу 2.4. Звідси випливає, що мотиваційний стан не є зовнішнім додатком до системи, а входить до внутрішнього стану студента і, принаймні опосередковано, може впливати на вибір наступних задач, інтенсивність рекомендацій чи режими подання підказок.

З теоретичної точки зору така інтеграція відкриває можливість створення нових мотиваційних систем та економічних механізмів у навчальних середовищах, орієнтованих на внутрішню мотивацію, а не на формальні атрибути досягнення. Якщо винагороджується не зовнішня ознака завершення, а значуща когнітивна дія, то цифровий стимул перестає бути «псевдооцінкою» і стає засобом підтримки академічної гідності, автономії та довгострокового залучення.

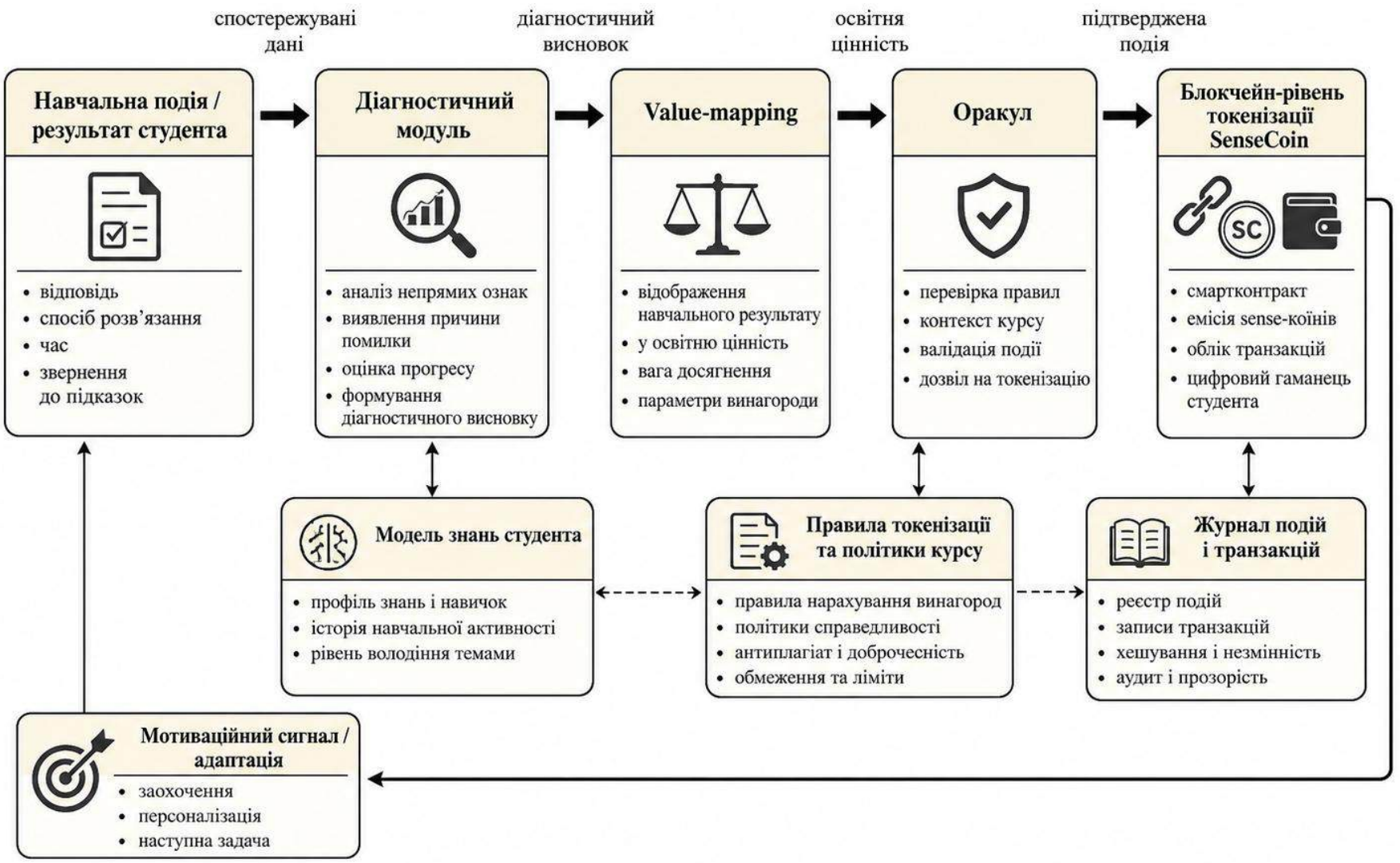


Рисунок 2.11 – Схема інтеграції діагностичного модуля, value-mapping, оракула та блокчейн-рівня токенізації SenseCoin

2.7 Модель простору параметричних навчальних задач

Для забезпечення масштабованості системи підтримки навчання точним наукам навчальна задача має існувати не лише як окремий статичний об'єкт, а як елемент простору параметризованих варіантів. Це особливо важливо для математичних дисциплін, де одна й та сама структура задачі може породжувати велику кількість коректних індивідуальних варіантів за рахунок зміни коефіцієнтів, початкових умов, типів функцій, розмірностей матриць, геометричних конфігурацій або умов обмежень. Тому необхідно ввести формальну модель простору параметричних задач, яка описує множину допустимих варіантів та правила переходу від шаблону до конкретної навчальної інстанції [5, 36].

Нехай $C = \{c_1, c_2, \dots, c_m\}$ – множина класів або шаблонів навчальних задач. Кожному шаблону c_j відповідає власний простір параметрів Ω_j. Загальний простір параметризованих задач можна представити як диз'юнктне об'єднання

$$\Omega = \bigcup_{j=1}^{m} \left(\{c_j\} \times \Omega_j\right) \tag{2.56}$$

Елементом цього простору є пара (c_j, θ), де c_j – шаблон задачі, а θ – конкретний вектор параметрів, який визначає один варіант задачі. Для більшості математичних задач параметри θ містять цілі або дійсні числа, але в загальному випадку можуть включати також булеві перемикачі, категорії формулювань, типи функцій, обмеження на домени, а для складних задач – і структурні параметри доказового шаблону.

Для кожного шаблону c_j вводиться вектор параметрів

$$\theta = (\theta_1, \theta_2, \dots, \theta_r) \in \Omega_j \tag{2.57}$$

Разом із простором параметрів необхідно визначити предикат коректності $\chi_j(\theta)$, який перевіряє, чи утворює вибраний набір параметрів математично допустимий варіант задачі. У найзагальнішому випадку

$$\chi_j : \Omega_j \to \{0,1\} \tag{2.58}$$

де $\chi_j(\theta)=1$ означає, що параметри задовольняють усі обмеження: немає заборонених значень, задача не вироджується, правильна відповідь належить допустимому класу, а обчислення еталонного розв'язку не приводить до суперечності. Наприклад, для задачі на розв'язання системи лінійних рівнянь методом Крамера необхідно забезпечити невиродженість матриці коефіцієнтів; для задачі на квадратне рівняння з цілими коренями – цілочисельність дискримінанта; для задачі на обчислення інтеграла – відсутність розривів на заданому проміжку.

Тоді множина допустимих варіантів шаблону c_j визначається як

$$\Omega_j^+ = \{\theta \in \Omega_j : \chi_j(\theta) = 1\} \tag{2.59}$$

а конкретна навчальна задача q породжується відображенням генерації

$$G_j : \Omega_j^+ \to Q_j \tag{2.60}$$

де Q_j – множина всіх коректних інстанцій задач класу c_j. Функція G_j формує текст умови, еталонну відповідь, службові об'єкти перевірки та прив'язку до компонентів знань. У найпростішому випадку вона має вигляд

$$G_j(\theta) = \langle Given_j(\theta), Input_j(\theta), Answer_j(\theta), Meta_j(\theta) \rangle \tag{2.61}$$

де $Meta_j(\theta)$ містить метадані: активовані компоненти знань, оцінку складності, рівень формалізовності, допустимі типи перевірки, шаблони типових помилок тощо.

Оскільки далеко не всі набори параметрів є допустимими, практичне формування задачі часто реалізується за схемою повторної генерації. У такому випадку випадково обраний θ перевіряється предикатами коректності параметрів та коректності отриманої відповіді:

$$\text{while} \chi_j(\theta) = 0 or \nu_j\left(Answer_j(\theta)\right) = 0 do regenerate \theta \tag{2.62}$$

де ν_j – предикат допустимості еталонного результату. Він потрібний, коли навіть математично допустимий набір параметрів породжує педагогічно небажаний варіант, наприклад надто громіздкий вираз, нетривіальну для поточного рівня складність або неінтерпретований формат відповіді.

Для адаптивної системи недостатньо просто генерувати будь-які допустимі варіанти. Необхідно вміти оцінювати складність конкретного варіанта задачі. Тому вводиться функція складності [22, 36]

$$Diff: \Omega \to R_+ \tag{2.63}$$

яка може залежати від кількості активованих компонентів знань, глибини дерева виразу, обсягу обчислень, рівня формалізовності, числа кроків очікуваного розв'язання, а також від історичних статистичних даних про те, як аналогічні варіанти проходили інші студенти. Узагальнено можна записати

$$Diff(c_j, \theta) = \lambda_1 \cdot \left|\Lambda\left(G_j(\theta)\right)\right| + \lambda_2 \cdot depth\left(T_j(\theta)\right) + \lambda_3 \cdot F\left(G_j(\theta)\right) + \lambda_4 \cdot comp(\theta) \tag{2.64}$$

де depth($T_j(\theta)$) – глибина структурного дерева задачі або еталонного перетворення, F($G_j(\theta)$) – рівень формалізовності варіанта, $comp(\theta)$ – оцінка обчислювальної складності, а λ_k – вагові коефіцієнти. Така функція не претендує на єдиний універсальний опис складності, але створює придатну основу для адаптивного добору варіантів.

Простір параметризованих задач повинен підтримувати також вимогу новизни. Якщо студент уже розв'язував надто схожі варіанти, система має або виключати частину області Ω_j, або вводити штраф за повторення. Формально це можна задати через функцію близькості

$$Sim(q_a, q_b) \in [0,1] \tag{2.65}$$

і вимогу відбору варіанта

$$q_t = \operatorname{argmax}_{q \in \Omega^+} U(q|K_t, H_t), U = PedagogicalValue - \mu \cdot RepeatPenalty \tag{2.66}$$

де RepeatPenalty зростає зі збільшенням Sim(q, q_prev), а PedagogicalValue враховує відповідність рівню знань, складність та покриття необхідних компонентів. Така модель робить простір задач не просто геометричним простором параметрів, а педагогічно структурованим простором варіантів.

Для складних задач на доведення та алгоритми параметризація має власну специфіку. Тут варіюватися можуть не лише числові коефіцієнти, а й структура самої задачі: вибір теореми, тип допоміжних лем, конфігурація припущень, форма представлення доведення, рівень допустимого автоматизму. У такому випадку параметричний простір набуває комбінаторного характеру, а елемент θ містить дискретні ознаки:

$$\theta = \left(\theta_{num}, \theta_{struct}, \theta_{lang}, \theta_{form}\right) \tag{2.67}$$

де θ_{num} – числові параметри, θ_{struct} – структурні параметри задачі, θ_{lang} – параметри лінгвістичного подання, θ_{form} – параметри формального режиму перевірки. Саме для таких просторів особливо актуальними стають оптимізаційні та квантово-прискорені процедури пошуку, розглянуті в наступному підрозділі.

Отже, модель простору параметризованих навчальних задач задає формальну основу для генерації великої кількості коректних, контрольованих за складністю та змістом варіантів задач. Вона поєднує математичну коректність, педагогічну придатність і вимогу новизни, а тому є необхідною ланкою між предметною моделлю задач і адаптивною логікою навчальної системи.

Таблиця 2.6 – Основні елементи моделі простору параметризованих задач

Елемент	Позначення	Призначення
Шаблон задачі	$\boldsymbol{c_j}$	Визначає клас задач і правила генерації
Простір параметрів	$\boldsymbol{\Omega_j}$	Множина всіх можливих параметрів шаблону
Предикат коректності	$\boldsymbol{\chi_j}$	Відсікає недопустимі параметри
Генератор варіанта	$\boldsymbol{G_j}$	Формує конкретну інстанцію задачі
Функція складності	Diff	Оцінює складність конкретного варіанта
Функція близькості	Sim	Контролює новизну і неповторюваність

2.8 Теоретична модель квантового прискорення пошуку в просторі навчальних задач

Побудована в підрозд. 2.7 модель простору параметричних навчальних задач природним чином приводить до задачі пошуку в скінченних або обмежених комбінаторних просторах. На практиці генерація чергової навчальної задачі не зводиться до простого випадкового вибору параметрів. Система повинна знайти такий вектор параметрів, який одночасно задовольняє умовам коректності, належить до потрібного класу складності, відповідає поточному стану знань студента та, за можливості, максимізує навчальну корисність. У класичній реалізації такі процедури виконуються методами перебору, відсікання некоректних варіантів, евристичного пошуку або послідовного випробування параметрів до отримання допустимого екземпляра [30, 31].

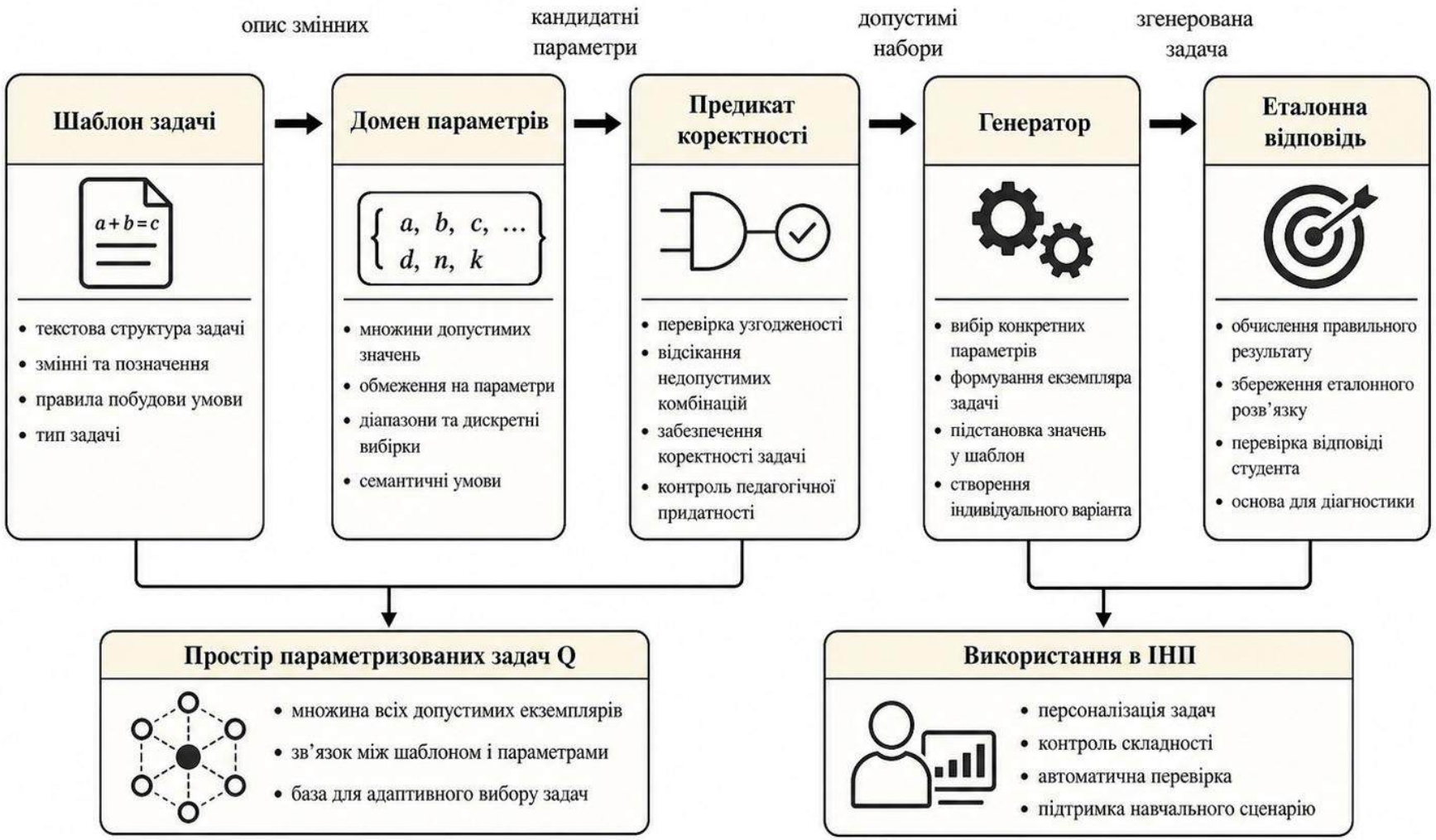


Рисунок 2.12 – Схема простору параметризованих задач: шаблон, домен параметрів, предикат коректності, генератор, еталонна відповідь

Зі зростанням розмірності параметричного простору Ω та ускладненням множини обмежень $\chi_j(\theta)$ вартість такого пошуку зростає. Особливо це помітно у випадках, коли система повинна не просто знайти будь-який допустимий варіант, а знайти варіант із заданими дидактичними властивостями: належністю до певної теми, рівня складності, типу формалізованості або підмножини компонентів знань. Тому в довгостроковій перспективі доцільно розглядати можливість використання квантових алгоритмів як прискорювачів окремих підзадач генерації, відбору та оцінювання навчальних екземплярів.

У цій роботі квантове прискорення не розглядається як самостійна заміна класичної інформаційної технології підтримки навчання. Йдеться лише про теоретичне обґрунтування можливості використання окремих квантових підпроцедур усередині класичної системи, збереження її архітектури та незмінності дидактичної логіки. Такий підхід є принципово важливим: зовнішній та внутрішній цикли навчання, моделі знань, мотиваційне регулювання й механізми формальної верифікації залишаються класичними, а квантові обчислення розглядаються як потенційний інструмент прискорення обчислювально затратних фрагментів.

Нехай для певного класу задач Q_j визначено параметричний простір Ω_j, предикат допустимості χ_j та множину допустимих параметрів Ω_j^+, як у формулах (2.56)–(2.59). Введемо також узагальнений предикат придатності

$$\chi(\theta): \Omega \to \{0,1\}, \chi(\theta) = 1 \Leftrightarrow \theta \in \Omega^+ \tag{2.68}$$

де Ω^+ є множиною всіх параметричних векторів, що породжують коректні навчальні задачі потрібного типу. Якщо параметри обираються рівноймовірно з Ω, то ймовірність прийняття випадкового вектора дорівнює

$$p = |\Omega^+|/|\Omega|. \tag{2.69}$$

У класичній схемі генерації на основі відбракування середня кількість випробувань до знаходження допустимого екземпляра визначається величиною

$$E[N_c lass] = 1/p, \tag{2.70}$$

а відповідна асимптотична складність за кількістю звернень до предиката χ становить $O(1/p)$. Якщо ж p є малим, наприклад через велику кількість структурних обмежень або вимогу до конкретного рівня складності, то процедура генерації може ставати вузьким місцем інформаційної технології.

Квантовий алгоритм амплітудного підсилення дозволяє замінити класичний пошук допустимого параметричного вектора на квантову процедуру, яка підсилює амплітуди правильних станів у суперпозиції параметрів. Нехай A – оператор підготовки стану, який формує суперпозицію векторів параметрів [30]

$$A\left|0\right\rangle = \sum_{\theta\in\Omega}\alpha_\theta\left|\theta\right\rangle, \sum_{\theta\in\Omega}|\alpha_\theta|^2 = 1. \tag{2.71}$$

Для простоти подальшого аналізу припустимо рівномірний розподіл, тобто $|\alpha_\theta|^2 = 1/|\Omega|$. Тоді можна ввести оракул O_χ, який змінює фазу станів, що відповідають допустимим параметрам:

$$O_\chi\left|\theta\right\rangle = (-1)^{\chi(\theta)}\left|\theta\right\rangle. \tag{2.72}$$

Композиція відбиття відносно початкового стану та фазового оракула породжує стандартний оператор амплітудного підсилення. У результаті очікувана кількість звернень до оракула зменшується з $O(1/p)$ до

$$O\left(1/\sqrt{p}\right), \tag{2.73}$$

що відповідає квадратичному прискоренню порівняно з класичною схемою відбракування. Для інформаційної технології підтримки навчання це означає, що при генерації складних параметризованих задач виграш може бути досягнутий вже на рівні підготовки коректних екземплярів, не змінюючи логіки подальшої перевірки, діагностики чи пояснення.

Другий важливий аспект полягає не лише у знаходженні допустимого параметра, а у попередньому оцінюванні частки допустимих параметрів для заданого класу задач або заданого кошика складності. Така оцінка може бути використана для автоматичного калібрування генератора, визначення очікуваної густини придатних екземплярів та адаптації зовнішнього циклу навчання. У

класичному випадку оцінювання p здійснюється методами Монте-Карло і вимагає $O(1/\varepsilon^2)$ вибірок для досягнення адитивної похибки ε. Квантове оцінювання амплітуди дає теоретичну можливість скоротити це число до

$$O(1/\varepsilon), \tag{2.74}$$

що також відповідає майже квадратичному прискоренню. Для навчальних систем це означає прискорене оцінювання насиченості простору задач екземплярами певної складності, а отже – швидше налаштування порогів адаптації та стабільнішу роботу механізмів зовнішнього циклу.

Окрім генерації, простір задач може використовуватися для пошуку екземплярів із заданими дидактичними властивостями. Нехай для фіксованого моменту часу t сформовано скінченну множину кандидатних задач [32, 33]

$$Q_t = \{q_1, q_2, \dots, q_N\}, \tag{2.75}$$

де кожна задача q_i вже є коректною, але не всі кандидати однаково корисні для поточного студента. Введемо булевий предикат навчальної придатності

$$f_t(q_i) = 1 \Leftrightarrow U(q_i, K_t, M_t, H_t) \geq \tau_U, \tag{2.76}$$

де τ_U – поріг корисності, а $U(q_i, K_t, M_t, H_t)$ – функція корисності, яка враховує стан знань K_t, мотиваційний стан M_t та історію навчання H_t. Якщо M з N кандидатів задовольняють нерівність (2.76), то класичний пошук першого придатного екземпляра в найгіршому випадку має складність $O(N)$, тоді як квантовий пошук Гровера дає оцінку

$$O\left(\sqrt{N/M}\right). \tag{2.77}$$

Це дозволяє розглядати квантовий пошук як перспективний інструмент для швидкого відбору задач із великого заздалегідь згенерованого банку, зокрема тоді, коли функція корисності включає складні структурні або формалізаційні критерії.

Ще одне перспективне застосування пов’язане з автоматичним добором діагностичних моделей та правил аналізу помилок. У попередніх підрозділах було показано, що неправильно подана студентом математична конструкція може бути представлена деревом виразу $T = (V, E, \lambda)$, а класифікаційна функція

φ відносить знайдене розходження до одного з діагностичних класів dm_k. Якщо припустити, що система повинна шукати найкращу підмножину діагностичних ознак, правил або фрагментів дерева для конкретного класу задач, то ця задача може бути зведена до комбінаторної оптимізації. У такому разі вона допускає представлення у вигляді квадратичної бінарної моделі оптимізації (QUBO), яка, у свою чергу, може розглядатися як об’єкт для QAOA або квантового відпалу. У межах цієї дисертації така постановка не доводиться до алгоритмічної реалізації, проте важлива як теоретичне продовження структурної та діагностичної моделей системи.

Не менш важливим є обмеження безпечного використання квантових підпроцедур. Усі операції, пов’язані з формальною верифікацією математичних доведень або алгоритмів, повинні зберігати класичне довірене ядро перевірки. Тому навіть у випадку використання квантового прискорення для пошуку лем, контрприкладів або кандидатів на параметричні екземпляри остаточне рішення про коректність повинне прийматися класичним верифікатором. Для задач, пов’язаних із доведеннями, таким верифікатором виступає trusted kernel систем Lean, Coq/Rocq або Isabelle/HOL. Отже, квантовий компонент не замінює логічне ядро, а лише постачає йому кандидатні об’єкти для перевірки [17, 18, 34, 35].

З педагогічної точки зору це означає, що квантове підсилення не змінює структуру взаємодії студента із системою. Студент, як і раніше, працює з тими самими задачами, поясненнями, підказками та механізмами верифікації; змінюється лише обчислювальна ефективність внутрішніх процедур генерації, добору й оцінювання. Завдяки цьому квантові методи можуть бути інтерпретовані як інфраструктурний рівень прискорення, а не як нова педагогічна парадигма.

Узагальнюючи викладене, теоретична модель квантового прискорення у просторі параметричних навчальних задач може бути представлена як композиція трьох класів процедур: 1) прискорення генерації коректних екземплярів на основі предиката допустимості χ; 2) прискорення оцінювання

щільності допустимих параметрів для калібрування складності; 3) прискорення пошуку придатних задач у множині кандидатів відповідно до функції корисності U. Формально це можна записати у вигляді відображення

$$QAcc: (\Omega, \chi, U) \mapsto (AA, QAE, Grover), \quad (2.78)$$

де AA, QAE та Grover позначають класи квантових підпроцедур, що можуть бути вкладені у класичний контур інформаційної технології підтримки навчання.

Разом з тим, ця модель має чіткі межі застосовності. Вона спирається на припущення про наявність квантового доступу до предикатів χ та f_t, а також на можливість побудови ефективних оракулів для перевірки допустимості й навчальної корисності. У реальних системах ці припущення не завжди виконуються, а отже очікувані виграші мають розглядатися як теоретично можливі, а не гарантовані у найближчій практиці. Додатковими факторами, що обмежують прикладну реалізацію, є експериментальний характер QRAM, чутливість QAOA до параметрів ініціалізації та залежність квантового відпалу від топології конкретної задачі.

Таблиця 2.7 – Порівняння класичних і квантових схем пошуку у просторі навчальних задач

Задача	Класичний підхід	Квантовий підхід	Теоретичний виграш
Генерація коректного параметричного екземпляра	Відбракування, $O(1/p)$	Амплітудне підсилення, $O\left(1/\sqrt{p}\right)$	Квадратичний
Оцінювання частки допустимих параметрів	Монте-Карло, $O(1/\varepsilon^2)$	QAE, $O(1/\varepsilon)$	Майже квадратичний
Пошук придатної задачі серед N кандидатів	Перебір, $O(N)$	Пошук Гровера, $O(\sqrt{(N/M)})$	Квадратичний за $M \ll N$
Комбінаторний добір структур/правил	Евристичний перебір	QUBO/QAOA/ annealing	Евристичне прискорення

Попри зазначені обмеження, включення теоретичної квантової моделі до системного опису підтримки навчання є доцільним. Воно дає змогу вже на етапі проєктування архітектури інформаційної технології передбачити точки майбутньої інтеграції нових обчислювальних парадигм, не порушуючи цілісності самої навчальної системи.

Рисунок 2.13 – Схема можливих точок інтеграції квантових підпроцедур у класичну інформаційну технологію підтримки навчання: генерація задач, оцінювання складності, пошук задач, добір діагностичних моделей

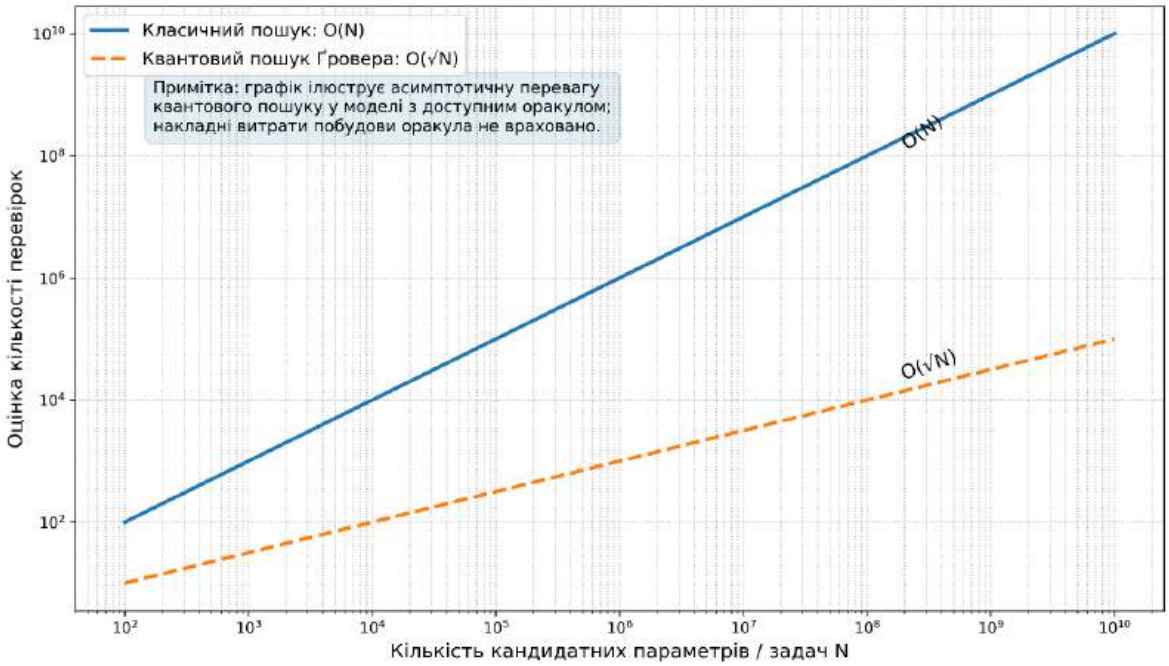


Рисунок. 2.14. Діаграма порівняння класичної та квантової складності пошуку у просторі параметричних задач

Висновки до розділу 2

У другому розділі виконано системне моделювання підтримки навчання точним наукам та побудовано взаємопов’язаний комплекс вербальних, структурних, функціональних і математичних моделей, що становлять теоретичну основу розроблюваної інформаційної технології.

1. Сформульовано мету та постановку задачі моделювання навчального процесу як складної керованої системи, у межах якої взаємодіють студент, навчальні задачі, компоненти знань, механізми адаптації, формальної верифікації та мотиваційного регулювання.

2. Побудовано вербальну модель системи підтримки навчання точним наукам, що описує її основні сутності, зв’язки між ними та логіку функціонування в межах зовнішнього та внутрішнього циклів навчання.

3. Розроблено структурну модель системи, у якій виділено модулі взаємодії зі студентом, генерації задач, перевірки відповідей, діагностики знань, формальної верифікації, мотиваційного регулювання та збереження історії навчання; визначено інформаційні потоки між цими компонентами.

4. Запропоновано функціональну модель системи, що формалізує послідовність основних навчальних дій: вибір або генерацію задачі, виконання відповіді, перевірку результату, діагностику помилок, оновлення стану знань та формування наступного навчального впливу.

5. Побудовано математичну модель навчальної задачі у вигляді кортежу $Q = (R, Given, Input, Answer, Algorithm)$, що дозволяє розрізняти прості задачі на застосування алгоритмів та складні задачі на побудову алгоритмів, виведення формул і доведення тверджень.

6. Розроблено формалізовану діагностичну модель стану знань студента, яка поєднує баєсівське подання ймовірностей засвоєння компонентів знань з аналізом структурних помилок і поведінкових ознак навчальної активності.

7. Побудовано модель мотиваційного регулювання навчального процесу на основі системи цифрових винагород, у якій стимули пов’язано не лише з правильністю результату, а й зі змістовними когнітивними подіями: структурним розпізнаванням, зусиллям, відновленням після помилки, творчим або соціальним внеском.

8. Запропоновано модель простору параметричних навчальних задач, яка задає формальні правила генерації коректних екземплярів, оцінювання складності та добору задач залежно від стану знань і навчальної мети.

9. Теоретично обґрунтовано можливість квантового прискорення окремих процедур пошуку в просторі параметричних навчальних задач, зокрема генерації допустимих екземплярів, оцінювання щільності коректних параметрів та пошуку придатних задач серед множини кандидатів.

10. Сукупність побудованих моделей створює цілісну основу для розроблення методів підтримки навчання точним наукам, які розглядаються в наступному розділі, та безпосередньо визначає вимоги до програмної реалізації інформаційної технології.

РОЗДІЛ 3 МЕТОДИ ПІДТРИМКИ НАВЧАННЯ ТОЧНИМ НАУКАМ

У другому розділі було побудовано систему моделей, яка описує підтримку навчання точним наукам на вербальному, структурному, функціональному, математичному та машинному рівнях. Третій розділ переходить від моделювання до методів, тобто до способів практичного використання побудованих моделей у навчальному процесі. Якщо моделі відповідають на питання, з яких сутностей складається система та як вони формально пов’язані, то методи визначають, яким чином ці сутності мають бути використані для діагностики, адаптації, формальної перевірки та генерації навчального контенту.

Основою запропонованих методів є поєднання двох підходів: інтелектуального тьюторингу та формальної верифікації. Перший підхід забезпечує адаптивність навчальної траєкторії, діагностику стану знань і добір наступного навчального впливу [70–78]. Другий підхід забезпечує логічну строгість математичних доведень і алгоритмічних рішень, що особливо важливо для точних наук, де правильна відповідь без правильного міркування не є достатнім доказом засвоєння матеріалу [53–60].

Методи, що розглядаються у цьому розділі, не зводяться до окремих програмних процедур. Вони задають педагогічно та математично обґрунтовану послідовність дій, яка може бути реалізована у програмній системі: від вибору формального артефакту або шаблону задачі до формування навчального сценарію, перевірки відповіді, локалізації причини помилки і оновлення стану знань студента. Такий підхід узгоджується з методологією навчання алгоритмам, у якій центральне місце займає перехід від змістовної задачі до формалізованої процедури її розв’язання [51, 52].

Розділ побудовано за логікою наростання формальної строгості. Спочатку розглядається метод інтеграції формальної верифікації математичних доведень у навчальний процес. Далі описується метод інтеграції формальної верифікації

алгоритмів, що має особливе значення для підготовки фахівців з програмної інженерії, відмовостійких та критичних до безпеки систем. Після цього подано метод адаптивного вибору задач, алгоритмічну модель параметричної генерації задач і метод формування поетапного навчального контенту на основі формально верифікованих артефактів.

3.1 Метод інтеграції формальної верифікації математичних доведень у навчальний процес

Навчання математичних доведень є однією з найскладніших складових навчання точним наукам. У традиційному навчальному процесі студент, як правило, спочатку сприймає готове доведення, потім відтворює його фрагменти, а далі намагається застосовувати аналогічні міркування до нових задач. Проблема полягає в тому, що комп’ютерна система без формального механізму перевірки не може надійно відрізнити логічно правильне доведення від тексту, який лише зовні нагадує доведення. Саме тому інтеграція proof-checking у навчання математичних доведень є ключовим елементом запропонованої інформаційної технології [53–56].

Метод інтеграції формальної верифікації математичних доведень полягає у використанні формально доведених тверджень як джерела навчальних сценаріїв. У цьому підході система Lean/Mathlib розглядається не як заміна викладача і не як інструмент примусового навчання синтаксису формальної мови, а як довірене ядро коректності, з якого видобувається структура доведення, набір залежностей, ключові леми та логічні переходи [53, 54, 56].

На відміну від звичайного автоматизованого тестування, формальна верифікація дозволяє перевіряти не тільки результат, але й саму логіку переходу від припущень до висновку. Це дає можливість формувати навчальні матеріали, у яких кожен крок пояснення має зв’язок із формально коректним твердженням. Такий підхід зменшує ризик появи некоректних або надмірно евристичних пояснень, що особливо актуально в умовах використання генеративних мовних моделей для підготовки навчального контенту [55, 87–90].

Загальна ідея методу подана на рис. 3.1. Формальне твердження у Lean/Mathlib використовується як початковий артефакт, після чого воно перетворюється на дерево залежностей, а далі - на поетапний навчальний сценарій. У цьому процесі важливо розділяти формальне ядро і дидактичний шар: формальне ядро відповідає за коректність, а дидактичний шар - за доступність пояснення для студента.

Вхідними даними методу є навчальна тема, цільове математичне твердження, перелік попередніх понять і набір формальних артефактів, що можуть бути використані для побудови навчального сценарію. Вихідними даними є структурований сценарій доведення, який містить кроки, пояснення, перевірювані твердження, можливі помилки та підказки. Такий сценарій може бути використаний у LMS або в окремому HTML-модулі навчання.

Нехай математичне твердження, що підлягає засвоєнню, позначено через T. Формальний артефакт, який підтверджує його коректність у середовищі Lean, подамо як пару

$$A_T = (\mathrm{Stmt}_T, \mathrm{Proof}_T) \tag{3.1}$$

де Stmt_T – формальне подання твердження, а Proof_T – формальна побудова доведення, прийнята системою перевірки. У навчальному процесі важливим є не лише сам факт існування Proof_T, а й можливість перетворити його на навчально зрозумілу структуру.

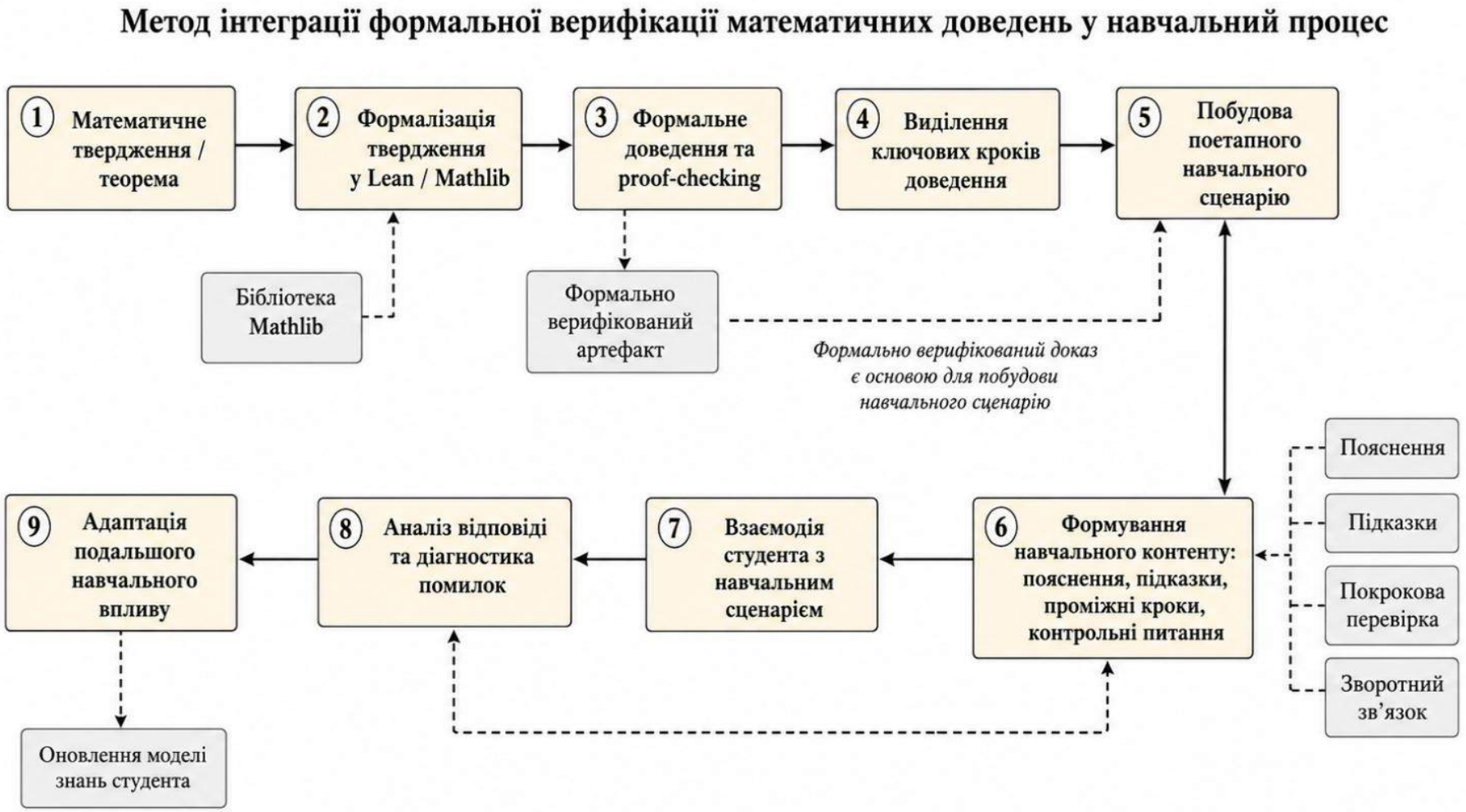


Рисунок 3.1 – Метод інтеграції формальної верифікації математичних доведень у навчальний процес

Для цього вводиться граф доведення

$$G_T = (V_T, E_T, \lambda_T, \tau_T) \tag{3.2}$$

де V_T – множина кроків доведення, E_T – відношення залежності між кроками, λ_T – функція змістовного маркування кроку, τ_T – тип кроку. Типами кроків можуть бути: введення означення, застосування леми, перетворення виразу, розгляд випадків, застосування індукції, узагальнення, побудова контрприкладу або завершення доведення. Подібне графове представлення дозволяє працювати з доведенням як із навчальним об’єктом, а не як із суцільним текстом.

Дидактичне перетворення формального графа доведення задається відображенням

$$\Phi: V_T \to \mathrm{Step}_T \tag{3.3}$$

де Step_T – множина навчальних кроків, кожен з яких містить формальний фрагмент, змістовне пояснення, очікувану дію студента, можливі помилки та підказки. Відображення Ф є ключовим, оскільки саме воно перетворює формальну структуру на навчальний сценарій.

Узагальнено метод складається з таких етапів: вибір цільового твердження; встановлення навчальної мети; пошук або підготовка формального артефакту; побудова графа залежностей; дидактична декомпозиція; формування кроків; перевірка відповідності кроків формальному артефакту; інтеграція сценарію у навчальне середовище. Основні етапи методу наведено в табл. 3.1.

Ключова відмінність запропонованого методу від простого використання Lean у навчанні полягає в тому, що студент не обов’язково працює безпосередньо з повним синтаксисом системи доказів. Система може приховувати складні формальні конструкції за математичним інтерфейсом, показуючи студенту саме ту структуру доведення, яка потрібна для навчання. При цьому формальна коректність зберігається через зв’язок навчального кроку з перевіреним артефактом [56, 57].

Таблиця 3.1 – Етапи методу інтеграції формальної верифікації математичних доведень

Етап	Зміст дії	Результат
1. Вибір твердження	Визначення теореми або властивості, яка повинна бути засвоєна студентом	Цільове твердження T
2. Формалізація	Пошук або підготовка Lean/Mathlib-артефакту	Формальний об’єкт A_T
3. Декомпозиція	Виділення означень, лем, залежностей і логічних переходів	Граф доведення G_T
4. Дидактична інтерпретація	Перетворення формальних кроків у навчальні пояснення	Множина навчальних кроків Step_T
5. Інтеграція	Подання сценарію в LMS або HTML-модулі	Навчальний матеріал з верифікованим ядром

Таким чином, метод інтеграції формальної верифікації математичних доведень виконує подвійну функцію. З одного боку, він підвищує якість навчального контенту за рахунок його узгодження з формальним доведенням. З іншого боку, він формує новий тип навчального сценарію, у якому студент бачить не лише результат, а логічну структуру переходу від припущень до висновку.

3.1.1 Представлення доведення як поетапного навчального сценарію

Поетапне представлення доведення є необхідним для того, щоб формальна верифікація стала педагогічно корисною. Формальний доказ у системі Lean може бути коротким для досвідченого користувача, але надто щільним і синтаксично складним для студента. Тому навчальна система повинна будувати проміжний

рівень представлення: між формальним доказом і звичайним поясненням природною мовою.

У запропонованій моделі навчальний сценарій доведення подається як скінченна послідовність кроків

$$S_T = \langle \text{step}_1, \text{step}_2, \ldots, \text{step}_n \rangle \tag{3.4}$$

де кожен крок step_i є структурованим об’єктом. На відміну від абзацу в підручнику, крок навчального сценарію повинен містити не лише текст пояснення, а й посилання на формальну основу, очікувану дію студента та діагностичні ознаки можливих помилок. Тому крок визначається як

$$\text{step}_i = (\text{claim}_i, \text{basis}_i, \text{action}_i, \text{check}_i, \text{hint}_i, \text{diag}_i) \tag{3.5}$$

де claim_i – локальне твердження або підціль, basis_i – означення, лема або попередній крок, action_i – очікувана дія студента, check_i – спосіб перевірки, hint_i – підказка, diag_i – множина діагностичних моделей, які можуть бути активовані у разі помилки.

Такий опис дозволяє поєднати формальну структуру доведення з внутрішнім циклом інтелектуальної навчальної системи. Якщо студент правильно виконує крок, система переходить до наступного. Якщо крок виконано помилково, система не просто повідомляє про помилку, а визначає її характер: невірно застосовано означення, пропущено умову теореми, некоректно виконано перетворення, неправильно вибрано лему або порушено логіку імплікації [71, 72, 84].

Для навчання точним наукам важливо виділяти різні типи кроків доведення. У табл. 3.2 наведено приклад класифікації кроків і відповідних навчальних дій.

Кроки сценарію можуть мати різний рівень відкритості. На початковому рівні система надає студенту майже повну структуру доведення і пропонує заповнити окремі переходи. На середньому рівні система вимагає самостійно вибирати леми й означення. На високому рівні студент сам формує стратегію

доведення, а система лише перевіряє формальну коректність окремих переходів або фінальної структури.

Таблиця 3.2 – Типи кроків доведення та відповідні навчальні дії

Тип кроку	Формальна роль	Навчальна дія студента	Можлива діагностика
Означення	Розкриття змісту поняття	Вибрати або записати означення	Нерозуміння базового поняття
Лема	Використання попередньо доведеного твердження	Вказати потрібну лему	Невміння встановити залежність
Перетворення	Зміна форми виразу	Виконати алгебраїчний або логічний перехід	Помилка правила перетворення
Розгляд випадків	Поділ задачі на підзадачі	Визначити повну множину випадків	Неповнота міркування
Завершення	Отримання цільового твердження	Пояснити, чому ціль доведена	Невміння зв’язати підціль з результатом

Для керування рівнем відкритості введемо параметр автономності $a_i \in [0,1]$ для кожного кроку. Якщо $a_i = 0$, студенту надається повна підказка. Якщо $a_i = 1$, крок виконується повністю самостійно. Тоді складність сценарію можна оцінити як

$$C(S_T) = \sum_{i=1}^{n} w_i \cdot a_i \tag{3.6}$$

де w_i – ваговий коефіцієнт складності конкретного кроку. Така оцінка може використовуватися для адаптивного добору сценарію: студент з низькою

впевненістю отримує більш деталізований сценарій, а студент із високим рівнем засвоєння – сценарій із більшою часткою відкритих кроків.

Важливо, що сценарій доведення не є простою послідовністю текстових підказок. Він має бути пов'язаний із моделлю знань студента. Кожен крок активує певні компоненти знань, і помилка в ньому оновлює не всю тему, а лише релевантні компоненти. Саме це забезпечує інтеграцію методу формальної верифікації з діагностичними моделями, розглянутими у другому розділі [84–86].

3.1.2 Формування навчального матеріалу на основі Lean/Mathlib

Формування навчального матеріалу на основі Lean/Mathlib передбачає побудову дидактичного шару над формальним артефактом. Формальна бібліотека Mathlib містить велику кількість перевірених тверджень, однак сама по собі вона не є навчальним матеріалом. Тому необхідний метод трансляції формальної структури у форму, придатну для студента [53–56].

Пропонований метод складається з п'яти етапів: вибору формального твердження; нормалізації математичних позначень; побудови дерева залежностей; генерації поетапного пояснення; вбудовування результату в HTML або LMS-сценарій. Під час цього процесу принципово важливо зберігати межу між формально перевіреним ядром і текстовим поясненням. Якщо пояснення автоматично генерується мовною моделлю, воно не повинно розглядатися як джерело коректності; коректність забезпечує лише формальний артефакт [55, 87–90].

У загальному вигляді метод можна подати оператором

$$\mathrm{Materialize}(A_T,\ L,\ R) \rightarrow \mathrm{Content}_T \tag{3.7}$$

де A_T – формальний артефакт, L – набір мовних і дидактичних правил пояснення, R – правила подання матеріалу в інтерфейсі, $\mathrm{Content}_T$ – навчальний контент, що містить формальне ядро, пояснення, кроки, підказки і контрольні точки.

Під час формування навчального матеріалу доцільно використовувати розділення на три шари: формальний, математичний і дидактичний. Формальний шар містить Lean/Mathlib-код або посилання на нього. Математичний шар містить звичні позначення і твердження у формі, прийнятній для курсу. Дидактичний шар містить пояснення, запитання до студента, підказки, варіанти помилок і фрагменти зворотного зв'язку.

Алгоритмічно процес формування навчального матеріалу можна описати так.

Алгоритм 3.1 – Формування навчального матеріалу на основі Lean/Mathlib

Крок	Опис операції
1	Вибрати математичне твердження T і визначити його навчальну мету.
2	Знайти або підготувати формальний артефакт A_T у Lean/Mathlib.
3	Виділити означення, леми, залежності та типи кроків доведення.
4	Побудувати граф G_T і сценарій S_T.
5	Згенерувати математичне пояснення кожного кроку.
6	Додати контрольні питання, підказки і діагностичні ознаки помилок.
7	Перевірити відповідність кожного кроку формальному артефакту.
8	Сформувати HTML/LMS-подання навчального матеріалу.

Прикладом застосування методу може бути навчальний матеріал з методу Крамера. Формальна бібліотека надає можливість працювати з матрицями, визначниками і твердженнями про розв'язування лінійних систем. У навчальному сценарії це перетворюється на послідовність кроків: постановка системи, умова невиродженості матриці, означення визначника, формула для

компоненти розв’язку, пояснення ролі заміни стовпця, перевірка отриманого виразу.

Головна перевага такого підходу полягає в тому, що студент отримує не лише правильну формулу, а логічну структуру її обґрунтування. При цьому система може адаптувати рівень деталізації: для одних студентів показувати повний ланцюг означень і лем, а для інших - лише ключові кроки з можливістю розкриття деталей за потреби.

Отже, метод формування навчального матеріалу на основі Lean/Mathlib є не просто технологічним способом створення HTML-сторінки. Він задає принцип побудови навчального контенту, у якому пояснення підпорядковане формально перевіреній структурі, а кожен крок може бути пов’язаний із компонентами знань і діагностичними моделями.

3.2 Метод інтеграції формальної верифікації алгоритмів у навчання побудові алгоритмічних рішень

Другим напрямом використання формальної верифікації є навчання побудові алгоритмічних рішень. Якщо у випадку математичних доведень центральним об’єктом є твердження та його доказ, то у випадку алгоритмів центральним об’єктом є програма або алгоритмічна процедура разом із її специфікацією. Навчальна мета полягає не лише в тому, щоб студент отримав правильний результат на тестових прикладах, а в тому, щоб він умів обґрунтовувати коректність алгоритму для всіх допустимих входів [61–65].

Традиційне навчання алгоритмів часто спирається на тестування: студент пише програму або псевдокод, після чого система перевіряє його на наборі прикладів. Такий підхід є корисним, але він не доводить коректність алгоритму. Навіть велика кількість тестів не гарантує відсутності помилки, особливо в граничних випадках, при складних умовах або в критичних системах. Формальна верифікація доповнює тестування доведенням властивостей алгоритму [61, 62, 66].

У навчанні фахівців з відмовостійких і критичних до безпеки систем питання коректності алгоритмів має особливе значення. Алгоритм керування, діагностики, відновлення або прийняття рішення повинен бути не лише працездатним у типових ситуаціях, а й обґрунтовано правильним за визначених припущень. Тому метод інтеграції формальної верифікації алгоритмів розглядається як інженерний компонент підтримки навчання точним наукам і алгоритмізовних дисциплін [63–69].

Загальна схема методу наведена на рис. 3.2. Вона починається з навчальної задачі на побудову алгоритму, далі формуються специфікація, передумови, постумови та інваріанти, після чого студентське рішення перетворюється на набір доказових зобов'язань. Результати перевірки використовуються для навчального висновку: які властивості доведені, які умови порушені, де потрібно уточнити інваріант або структуру алгоритму.

Рисунок 3.2 – Метод інтеграції формальної верифікації алгоритмів у навчання побудові алгоритмічних рішень

Формально алгоритмічну задачу можна подати як четвірку

$$\text{AlgTask} = (P,\ A,\ Q,\ \Omega) \tag{3.8}$$

де P – передумова, A – алгоритм або програмна процедура, Q – постумова, Ω – множина додаткових властивостей: інваріантів, обмежень ресурсів, властивостей завершення, відсутності помилок доступу або збереження узгодженості даних.

Коректність алгоритму у найпростішому випадку задається триплетом Гоара

$$\{P\}\,A\,\{Q\} \tag{3.9}$$

Цей запис означає: якщо перед виконанням алгоритму A істинна передумова P і алгоритм завершується, то після його завершення істинна постумова Q. Для повної коректності додатково потрібно довести завершуваність алгоритму. У навчальному контексті важливим є не лише кінцевий факт доведення, а процес побудови передумов, постумов та інваріантів [61, 62].

Метод інтеграції формальної верифікації алгоритмів включає такі етапи: формулювання задачі; визначення специфікації; побудова алгоритму; виділення інваріантів; формування доказових зобов’язань; перевірка; діагностика порушень; формування навчального зворотного зв’язку. На відміну від звичайного автотестування, результатом є не лише verdict pass/fail, а пояснення того, яка саме властивість алгоритму не доведена.

У запропонованій системі тестування та формальна верифікація не протиставляються. Тестування може використовуватися як первинний швидкий фільтр, що виявляє очевидні помилки. Формальна верифікація застосовується до ключових фрагментів, де потрібно довести властивість або навчити студента працювати з інваріантами. Такий комбінований підхід є педагогічно доцільним, оскільки повна формалізація кожної навчальної задачі може бути надмірною, але

її використання для ключових алгоритмів має високий навчальний ефект [66–69].

Таблиця 3.3 – Порівняння тестування і формальної верифікації алгоритмів у навчанні

Критерій	Тестування	Формальна верифікація
Об’єкт перевірки	Результат на скінченній множині прикладів	Властивість алгоритму для класу допустимих входів
Тип висновку	Емпіричний	Логічно обґрунтований
Діагностична цінність	Виявляє наявність помилки у прикладі	Вказує порушену умову, інваріант або властивість
Навчальний ефект	Формує навички налагодження	Формує навички специфікації та доказу коректності
Обмеження	Не гарантує повної коректності	Потребує формалізації та більш високого рівня абстракції

3.2.1 Коректність алгоритмів як навчальний об’єкт

Коректність алгоритму в навчальному процесі слід розглядати як самостійний навчальний об’єкт. Студент повинен засвоїти не лише те, як працює алгоритм, а й те, чому він працює правильно. Для цього необхідно перейти від інтуїтивного пояснення до формальних категорій: передумова, постумова, інваріант, варіант циклу, умова завершення, допустимий вхід, граничний випадок.

Нехай алгоритм A має область допустимих входів X і множину виходів Y. Його специфікація задається відношенням

$$\mathrm{Spec}_A \subseteq X \times Y \tag{3.10}$$

Алгоритм A є частково коректним відносно Spec_A, якщо для кожного $x \in X$ у разі завершення алгоритму результат $y = A(x)$ задовольняє $(x, y) \in Spec_A$.

Якщо додатково доведено завершуваність A для всіх $x \in X$, алгоритм є тотально коректним. Саме різниця між частковою і тотальною коректністю повинна бути предметом навчального пояснення [61, 65].

У навчанні алгоритмів студенти часто роблять помилки не в операторному записі, а у специфікації. Наприклад, вони можуть правильно реалізувати сортування для типових масивів, але не визначити поведінку для порожнього масиву, повторюваних елементів або вже відсортованого входу. З точки зору формальної верифікації це означає неповну або некоректну специфікацію, а з точки зору навчання - прогалину в розумінні задачі.

Для алгоритмів із циклами ключовим є поняття інваріанта. Інваріант I повинен бути істинним до початку циклу, зберігатися після кожної ітерації та разом з умовою завершення давати постумову. У навчальному сценарії це може бути подано як послідовність питань: що залишається незмінним після кожного кроку, яку частину задачі вже розв’язано, що зменшується або наближається до завершення.

$$P \Rightarrow I, \quad \{I \wedge B\}\,\mathrm{body}\,\{I\}, \quad I \wedge \neg B \Rightarrow Q \tag{3.11}$$

Тут B – умова продовження циклу, body – тіло циклу. Запис (3.11) відображає три базові доказові зобов’язання для інваріанта. У навчальній системі кожне з цих зобов’язань може бути перетворене на окремий крок сценарію: встановити інваріант, довести його ініціалізацію, довести збереження, зв’язати його із постумовою.

Таким чином, коректність алгоритмів як навчальний об’єкт має багаторівневу структуру: рівень специфікації, рівень побудови алгоритму, рівень інваріантів, рівень доказових зобов’язань і рівень діагностики помилок. Саме ця структура відрізняє метод формальної верифікації алгоритмів від звичайного навчання програмуванню через приклади.

3.2.2 Верифікація алгоритмів у прикладних та інженерних задачах

В інженерних дисциплінах алгоритми часто використовуються не як абстрактні навчальні об’єкти, а як частини систем керування, діагностики, планування або обробки даних. У таких задачах помилка алгоритму може мати не лише навчальні, а й практичні наслідки. Тому навчання формальній коректності алгоритмів є особливо важливим для підготовки фахівців, які працюють з відмовостійкими, вбудованими та критичними до безпеки системами [63–69].

Метод інтеграції верифікації в інженерні задачі передбачає, що навчальний приклад повинен містити не тільки програмний фрагмент, а й контекст його використання. Наприклад, алгоритм відновлення після збою, алгоритм голосування в надлишковій системі, алгоритм контролю граничного параметра або алгоритм вибору безпечного режиму роботи повинні розглядатися через специфікацію умов, за яких вони є коректними.

Для інженерної задачі введемо модель

$$\text{EngAlgTask} = (\text{Env}, P, A, Q, \text{Fail}, \text{Safe}) \tag{3.12}$$

де Env – модель середовища, P – передумови, A – алгоритм, Q – цільова постумова, Fail – множина відмов або порушень, Safe – властивість безпечного стану. У цьому випадку перевірка алгоритму повинна враховувати не лише функціональну правильність, а й збереження безпечного стану за наявності допустимих відмов.

У навчальному сценарії це може бути подано як поступове ускладнення. Спочатку студент перевіряє алгоритм на типових даних. Потім формулює передумови. Далі визначає інваріанти або властивості безпеки. Після цього система формує доказові зобов’язання, а результати перевірки використовуються для діагностики. Якщо студент неправильно задав Safe, система вказує на проблему в інженерній інтерпретації вимоги. Якщо неправильно побудовано A, система вказує на конкретний фрагмент алгоритму.

Для задач із відмовостійкості корисним є розділення помилок на три групи: помилки специфікації, помилки алгоритмічної логіки і помилки обробки граничних або аварійних випадків. Така класифікація дозволяє формувати не лише технічний, а й дидактичний зворотний зв'язок: студент бачить, чи проблема в нерозумінні вимоги, в неправильній логіці алгоритму або в неповному аналізі середовища.

Отже, метод інтеграції формальної верифікації алгоритмів у навчання побудові алгоритмічних рішень забезпечує перехід від емпіричного навчання через тестування до навчання через специфікацію, доказ і діагностику. Це дозволяє формувати у студентів не лише навички програмування, а й інженерну культуру побудови коректних алгоритмічних рішень.

3.3 Метод адаптивного вибору навчальних задач на основі оцінки стану знань

Адаптивний вибір навчальних задач є центральним механізмом зовнішнього циклу інтелектуальної навчальної системи. Його призначення полягає в тому, щоб на кожному кроці обирати таку задачу, яка найбільшою мірою відповідає поточному стану знань студента, навчальній меті, історії помилок і мотиваційному стану [70–78].

На відміну від лінійного навчального сценарію, де всі студенти виконують одну й ту саму послідовність задач, адаптивний метод працює з індивідуальною траєкторією. Така траєкторія не повинна бути випадковою або побудованою лише за результатом останньої відповіді. Вона має враховувати модель знань, складність задачі, діагностичні гіпотези, бажаний рівень автономності та мотиваційні обмеження.

Нехай стан студента на кроці t подано як

$$S_t = (K_t, M_t, H_t, D_t) \tag{3.13}$$

де K_t – вектор оцінок засвоєння компонентів знань, M_t – мотиваційний стан, H_t – історія взаємодії, D_t – множина активних діагностичних гіпотез. Множину

кандидатних задач позначимо Q_t. Кожна задача q ∈ Q_t має власний профіль: компоненти знань, складність, тип, можливість формальної перевірки, час виконання, очікуваний навчальний ефект.

Метод адаптивного вибору полягає у визначенні функції корисності U(q,S_t), яка оцінює доцільність запропонувати задачу q студенту в стані S_t. Загальний вигляд вибору має форму

$$q_t^* = \text{argmax}_{q \in Q_t} U(q, S_t) \tag{3.14}$$

за умови, що задача задовольняє обмеженням допустимості: не є надто легкою, не є надто складною, відповідає темі, має коректний шаблон, може бути перевірена і пов'язана з релевантними компонентами знань.

Складова корисності може бути подана як зважена сума кількох критеріїв:

$$U(q, S_t) = \alpha \cdot \text{Gap}(q, K_t) + \beta \cdot \text{Diag}(q, D_t) + \gamma \cdot \text{Mot}(q, M_t) + \delta \cdot \text{Var}(q, H_t) - \eta \cdot \text{Cost}(q) \tag{3.15}$$

Тут Gap(q, K_t) оцінює відповідність задачі прогалинам у знаннях; Diag(q, D_t) – релевантність задачі активним діагностичним гіпотезам; Mot(q, M_t) – мотиваційну придатність задачі; Var(q, H_t) – різноманітність відносно попередньої історії; Cost(q) – очікувану складність або часові витрати. Коефіцієнти α, β, γ, δ, η задають пріоритети системи.

Змістовно метод адаптивного вибору задач зображено на рис. 3.3. Система поєднує оцінку знань, мотиваційний стан, множину кандидатних задач і навчальні цілі у функцію корисності, після чого обирає наступну задачу.

Критерії функції корисності мають різну природу. Gap(q, K_t) можна визначити як відстань між бажаним рівнем засвоєння компонентів і поточними оцінками студента. Якщо C(q) - множина компонентів знань, задіяних у задачі, то

$$\text{Gap}(q, K_t) = \sum_{s_i \in C(q)} w_i \cdot \left(K_i^{\text{target}} - K_{t,i}\right)_+ \tag{3.16}$$

Ця величина є більшою для задач, які активують саме ті компоненти, що потребують доопрацювання. Якщо компонент уже добре засвоєний, він не повинен надмірно підвищувати корисність задачі.

Діагностична релевантність Diag(q, D_t) визначається тим, наскільки задача здатна перевірити активну гіпотезу про причину помилки. Наприклад, якщо попередня помилка пов’язана з неправильним застосуванням правила обчислення визначника, система повинна запропонувати задачу, яка локально перевіряє саме цей компонент, а не всю тему лінійної алгебри. Це безпосередньо пов’язує адаптивний вибір із діагностичними моделями [84–86].

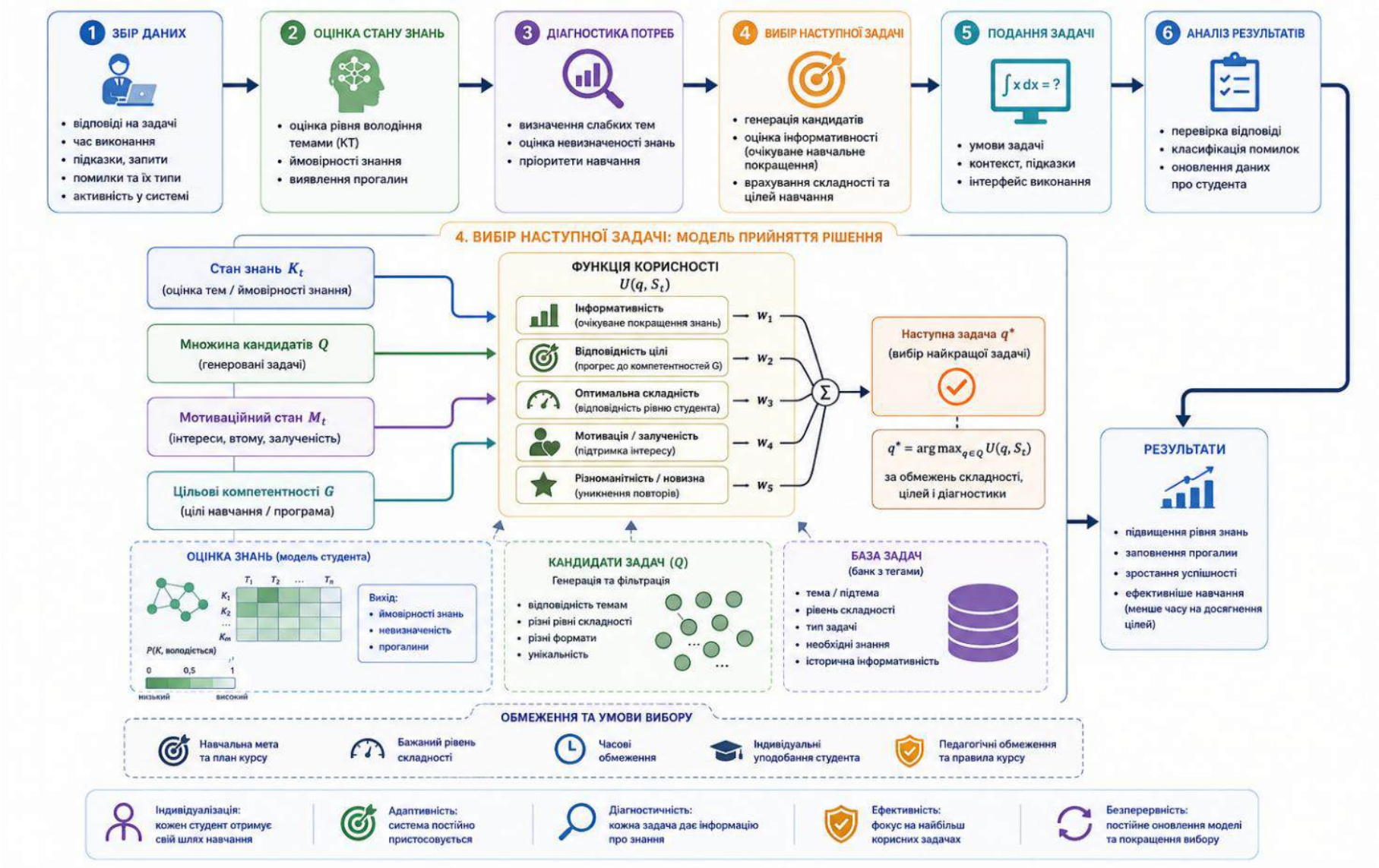


Рисунок 3.3 – Метод адаптивного вибору навчальних задач на основі оцінки стану знань

Мотиваційна складова Mot(q, M_t) виконує роль обмежувача та підсилювача. Якщо студент демонструє втому або серію невдач, система може обрати задачу з нижчим рівнем складності, але з високою діагностичною цінністю. Якщо студент демонструє стійкий прогрес, система може підвищити рівень автономності або запропонувати складніший сценарій [87–90].

Таблиця 3.4 – Основні критерії адаптивного вибору навчальної задачі

Критерій	Зміст	Приклад використання
Gap	Відповідність задачі прогалинам у знаннях	Добір задачі на компонент з низькою оцінкою засвоєння
Diag	Зв’язок із активною діагностичною гіпотезою	Перевірка ймовірної причини попередньої помилки
Mot	Мотиваційна придатність задачі	Зменшення складності після серії невдач або підвищення автономності після прогресу
Var	Різноманітність навчальної траєкторії	Уникнення повторення однотипних задач без дидактичної потреби
Cost	Часова і когнітивна вартість задачі	Недопущення перевантаження студента

Алгоритм адаптивного вибору задачі можна подати як послідовність дій. Спочатку система оновлює стан знань після останньої відповіді. Далі формує множину кандидатів. Потім для кожної задачі обчислює корисність і перевіряє обмеження. Після цього обирається задача з максимальною корисністю або одна з кількох задач із високою корисністю з урахуванням різноманітності.

Метод адаптивного вибору задач повинен підтримувати як зовнішній, так і внутрішній цикл навчання. На рівні зовнішнього циклу він визначає, яку задачу або тему запропонувати наступною. На рівні внутрішнього циклу він може визначати, яку підказку, мікрокрок або додатковий формально верифікований фрагмент пояснення надати студенту під час розв’язання [71, 72, 77].

Алгоритм 3.2 – Адаптивний вибір навчальної задачі

Крок	Опис операції
1	Отримати поточний стан $S_t = (K_t, M_t, H_t, D_t)$.
2	Сформувати множину кандидатних задач Q_t.
3	Відкинути задачі, що не задовольняють умовам допустимості.
4	Для кожної задачі q обчислити U(q, S_t).
5	Обрати q $*_t$ = argmax U(q, S_t) або сформувати короткий список задач.
6	Подати задачу студенту і зафіксувати навчальну взаємодію.
7	Після відповіді виконати перевірку, діагностику та оновлення стану знань.

Таким чином, адаптивний вибір задач у запропонованій системі не є простим правилом типу “після помилки дати легшу задачу”. Це багатокритеріальний метод, який пов’язує модель знань, діагностику помилок, мотиваційний стан, складність задач і цільові компетентності в єдину процедуру добору навчального впливу.

3.4 Алгоритмічна модель параметричної генерації навчальних задач

Параметрична генерація навчальних задач є одним із практичних механізмів масштабування інтелектуальної навчальної системи. Її призначення полягає в тому, щоб на основі шаблону задачі та домену параметрів автоматично формувати велику кількість коректних і дидактично придатних варіантів. На відміну від випадкової генерації текстів, параметрична генерація повинна гарантувати математичну коректність умови, існування еталонної відповіді та можливість автоматичної перевірки [79–83].

У другому розділі було сформульовано модель простору параметризованих задач. У цьому підрозділі ця модель перетворюється на метод. Для кожного шаблону T визначається домен параметрів Θ_T, предикат коректності χ_T і функція генерації Gen_T. Задача вважається допустимою лише

тоді, коли вибраний набір параметрів задовольняє всі математичні, технічні та педагогічні обмеження.

Загальна схема параметричної генерації задач подана на рис. 3.4. Шаблон і домен параметрів утворюють простір можливих екземплярів, предикат коректності відсікає недопустимі комбінації, генератор формує конкретну задачу, а еталонна відповідь використовується для перевірки результату студента.

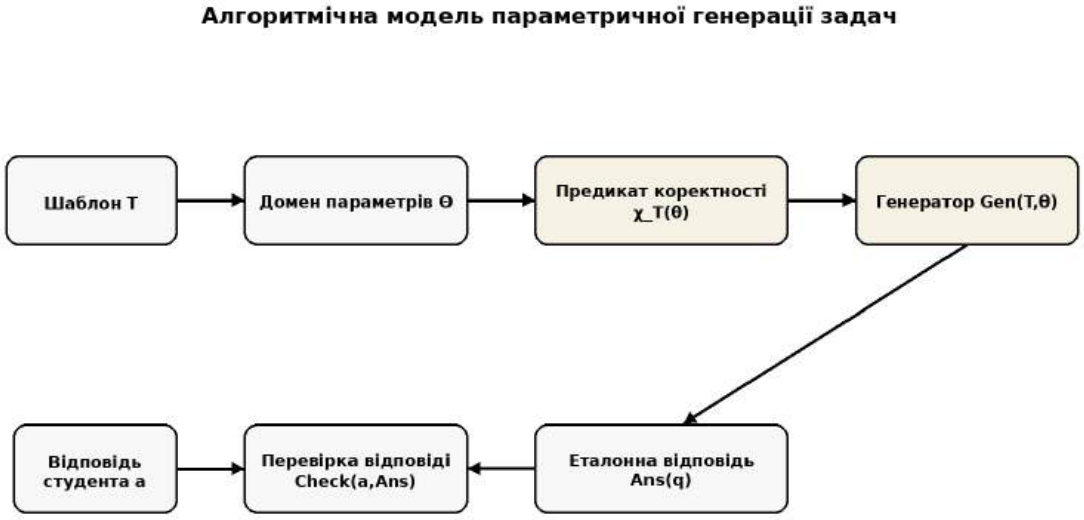


Рисунок 3.4 – Алгоритмічна модель параметричної генерації навчальних задач

Формально шаблон задачі можна подати як

$$T = (\mathrm{Text}_T,\ \Theta_T,\ \chi_T,\ \mathrm{Ans}_T,\ \mathrm{Check}_T) \tag{3.17}$$

де Text_T – текстова або структурна форма умови, Θ_T - простір параметрів, χ_T – предикат коректності, Ans_T – процедура обчислення еталонної відповіді, Check_T – процедура перевірки відповіді студента.

Екземпляр задачі q формується як результат підстановки допустимого набору параметрів $\theta \in \Theta_T$ у шаблон T:

$$q = \mathrm{Gen}_T(\theta), \quad \text{де} \quad \chi_T(\theta) = 1 \tag{3.18}$$

Якщо χ_T (θ)=0, набір параметріввідкидається, оскільки він породжує некоректну, вироджену або педагогічно непридатну задачу. Наприклад, у задачах лінійної алгебри предикат коректності може забороняти нульовий визначник, якщо задача передбачає застосування методу Крамера; у задачах математичного аналізу він може забороняти значення, що порушують область визначення функції.

Алгоритмічна модель параметричної генерації повинна забезпечувати три властивості: коректність, варіативність і контроль складності. Коректність означає, що задача має математичний сенс і правильну відповідь. Варіативність означає, що система може створювати багато екземплярів одного типу. Контроль складності означає, що параметри впливають на рівень обчислювальної або концептуальної складності задачі.

3.4.1 Генерація параметрів задачі

Генерація параметрів є першим етапом створення екземпляра задачі. Вона не повинна бути довільною. Кожен параметр має домен допустимих значень, а група параметрів має спільні обмеження. Тому генерація параметрів подається як вибір θ з домену Θ_T з урахуванням обмежень χ_T.

Нехай

$$\Theta_T = \Theta_1 \times \Theta_2 \times ... \times \Theta_m \tag{3.19}$$

де Θ_i – домен і-го параметра. Набір θ = (Θ_1, Θ_2, ..., Θ_m) є кандидатним набором параметрів. Він стає допустимим лише після перевірки предиката χ_T. При цьому домени можуть бути числовими, символьними, структурними або змішаними. Наприклад, для задачі з матрицями параметрами можуть бути розмірність, елементи матриці, тип правої частини, обмеження на визначник та бажаний рівень складності.

Метод генерації параметрів може використовувати кілька стратегій: випадковий вибір із фільтрацією; конструктивну генерацію, за якої параметри одразу будуються коректними; генерацію за складністю; генерацію з

урахуванням стану знань студента. Для навчальної системи особливо важливо, щоб генерація не лише створювала коректні задачі, а й підтримувала адаптивний вибір.

Для керування складністю введемо функцію

$$\mathrm{diff}_T: \Theta_T \to \mathbb{R}_+ \tag{3.20}$$

яка ставить набору параметрів θ у відповідність оцінку складності майбутньої задачі. Тоді генерація може виконуватися не з усього домену, а з підмножини

$$\Theta_T(d_1, d_2) = \{\theta \in \Theta_T \mid d_1 \leq \mathrm{diff}_T(\theta) \leq d_2\} \tag{3.21}$$

Це дозволяє добирати параметри відповідно до поточного рівня студента. Якщо система виявила прогалину в базовій процедурі, вона може обрати простіші параметри, які ізолюють потрібний компонент знань. Якщо студент демонструє стійкий прогрес, система може ускладнювати параметри, збільшуючи кількість кроків або вводячи менш очевидні випадки.

3.4.2 Перевірка коректності параметрів

Перевірка коректності параметрів є необхідною умовою використання генерації в навчанні точним наукам. Некоректно згенерована задача може мати невизначений результат, суперечливу умову, кілька несподіваних відповідей або вимагати знань, які не відповідають цільовій темі. Тому предикат χ_T має враховувати не лише математичні, а й дидактичні обмеження.

У загальному вигляді предикат коректності можна подати як кон'юнкцію кількох груп умов:

$$\chi_T(\theta) = \chi_{\mathrm{math}}(\theta) \wedge \chi_{\mathrm{domain}}(\theta) \wedge \chi_{\mathrm{did}}(\theta) \wedge \chi_{\mathrm{check}}(\theta) \tag{3.22}$$

Тут χ_{math} перевіряє математичну узгодженість, χ_{domain} - належність параметрів допустимим областям, χ_{did} - педагогічну придатність, χ_{check} - можливість автоматичної перевірки відповіді. Наприклад, задача може бути математично коректною, але непридатною для конкретного навчального сценарію, якщо її розв'язання потребує ще не вивчених методів.

Перевірка параметрів має виконуватися до подання задачі студенту. Це дозволяє уникнути ситуацій, у яких система створює задачу з некоректною умовою або відповіддю. Для задач точних наук це принципово: навіть незначна помилка в домені параметрів може призвести до неправильної відповіді, а отже – до помилкової діагностики знань студента.

Алгоритм перевірки коректності параметрів наведено нижче.

Алгоритм 3.3 – Перевірка коректності параметрів задачі

Крок	Опис операції
1	Згенерувати кандидатний набір параметрів θ.
2	Перевірити належність θ до домену Θ_T.
3	Перевірити математичні обмеження χ_math.
4	Перевірити дидактичні обмеження χ_did.
5	Перевірити можливість обчислення еталонної відповіді Ans_T.
6	Якщо всі умови виконано, передати θ до генератора; інакше відкинути θ або змінити параметри.

У практичній реалізації частина перевірок може бути задана явно в коді, частина – через символьні перетворення, а частина – через формальні або напівформальні обмеження. Наприклад, для задач на визначники можна явно перевірити ненульовість det(A), для задач на інтеграли – належність функції потрібному класу, для задач на диференціальні рівняння – тип рівняння і форму коренів характеристичного многочлена.

3.4.3 Перевірка правильності результату

Після генерації задачі система повинна мати можливість перевірити відповідь студента. Для простих задач перевірка може бути числовою або символьною; для складних задач вона може включати структурне порівняння, діагностику помилок або формальну верифікацію.

Нехай Ans_T (θ) – еталонна відповідь для задачі $q = Gen_T(\theta)$, а a – відповідь студента. Тоді перевірка задається функцією

$$\mathrm{Check}_T\big(a,\ \mathrm{Ans}_T(\theta)\big) \to \{0,1\} \times Z \qquad (3.23)$$

де перша компонента результату означає правильність або неправильність відповіді, а Z – множина спостережуваних ознак, які можуть бути використані для діагностики. Таке визначення принципово відрізняється від простої перевірки «правильно/неправильно»: система зберігає додаткові дані про характер помилки.

Для числових відповідей перевірка може враховувати допустиму похибку. Для символьних відповідей необхідно використовувати нормалізацію виразів, приведення до канонічної форми або порівняння дерев виразів. Для доведень і алгоритмів застосовується формальна перевірка або перевірка доказових зобов'язань [53–57, 82, 84].

Загальне правило полягає в тому, що еталонна відповідь не повинна зберігатися лише як текстовий рядок. Вона має бути структурованим об'єктом, придатним для порівняння, пояснення і діагностики. Наприклад, для математичного виразу корисно зберігати дерево виразу; для алгоритму – специфікацію та набір тестових і доказових умов; для доведення - послідовність кроків або посилання на формальний артефакт.

Отже, алгоритмічна модель параметричної генерації задач забезпечує не лише створення варіативних умов, а й повний цикл навчальної взаємодії: формування коректної задачі, обчислення еталонної відповіді, перевірку студентського результату та передачу ознак помилки до діагностичного ядра.

3.5 Метод формування поетапного навчального контенту з формально верифікованих артефактів

Останній метод, що розглядається у цьому розділі, пов'язує формальну верифікацію, параметричну генерацію та адаптивне навчання в єдиний процес створення навчального контенту. Його призначення полягає в тому, щоб формальні артефакти - теореми, доведення, алгоритми, специфікації, інваріанти

– перетворювалися на поетапні навчальні матеріали, придатні для використання в інтерфейсі системи.

На відміну від звичайного навчального тексту, поетапний контент має бути керованим. Він повинен підтримувати різні рівні деталізації, можливість приховування або розкриття підказок, перевірку проміжних дій, локалізацію помилок і адаптацію наступного кроку. Це особливо важливо для студентів, які засвоюють не лише формули, а логіку побудови доведення або алгоритму [71, 72, 85, 86].

Загальна схема формування такого контенту наведена на рис. 3.5. Формальний артефакт нормалізується, декомпонується на кроки, перетворюється на дидактичні пояснення і далі вбудовується в HTML або LMS-сценарій. При цьому формально верифіковане ядро залишається джерелом коректності.

Рисунок 3.5 – Метод формування поетапного навчального контенту з формально верифікованих артефактів

Формальний артефакт F у цьому методі розглядається як структурований об’єкт

$$F = (\text{Obj}, \text{Spec}, \text{Proof}, \text{Dep}, \text{Meta}) \tag{3.24}$$

де Obj – математичний або алгоритмічний об’єкт, Spec – специфікація або твердження, Proof – формальне доведення або перевірка, Dep – залежності від інших означень і тверджень, Meta – метадані навчального використання. На основі F будується навчальний об’єкт Content(F).

Формування контенту виконується через послідовність перетворень:

$$F \rightarrow \mathrm{Normalize}(F) \rightarrow \mathrm{Decompose}(F) \rightarrow \mathrm{Explain}(F) \rightarrow \mathrm{Render}(F) \quad (3.25)$$

Етап Normalize приводить позначення та структуру до форми, зрозумілої в межах навчального курсу. Етап Decompose розкладає артефакт на кроки. Етап Explain створює пояснення, питання і підказки. Етап Render формує кінцеве подання: HTML-сторінку, LMS-блок, інтерактивний сценарій або фрагмент навчального модуля.

Кожен крок навчального контенту має містити мінімально необхідну інформацію: ціль кроку, математичне твердження, пояснення, очікувану дію студента, спосіб перевірки, типові помилки, підказку і зв’язок із компонентами знань. Такий формат дозволяє інтегрувати контент у діагностичну й адаптивну логіку системи.

Таблиця 3.5 – Структура поетапного навчального контенту

Елемент кроку	Призначення
Ціль кроку	Пояснює, що саме має бути засвоєно або виконано
Формальне твердження	Забезпечує зв’язок із верифікованим артефактом
Математичне пояснення	Перекладає формальний зміст у навчальну форму
Очікувана дія	Визначає, що повинен зробити студент
Перевірка	Фіксує критерій правильності кроку
Типові помилки	Пов’язує крок із діагностичними моделями
Підказка	Надає допомогу без повного розкриття відповіді
Компоненти знань	Вказує, які елементи моделі знань оновлюються

Метод формування поетапного контенту може використовувати генеративні мовні моделі як допоміжний інструмент для підготовки текстових пояснень, але не як джерело математичної істини. Пояснення, згенеровані автоматично, повинні перевірятися відносно формального артефакту або експертного шаблону. Це дозволяє поєднати гнучкість природномовного пояснення із строгістю формальної верифікації [55, 87–90].

Особливо важливим є питання рівня деталізації. Один і той самий формальний артефакт може породжувати різні навчальні матеріали: коротке нагадування для сильного студента, детальний покроковий сценарій для студента з прогалинами, діагностичний сценарій для перевірки конкретної помилки або демонстраційний матеріал для лекції. Тому метод повинен підтримувати параметризацію контенту за рівнем автономності й деталізації.

Нехай $r \in [0,1]$ - параметр деталізації, де $r=0$ означає мінімальне пояснення, а $r=1$ – максимально деталізований сценарій. Тоді генерація навчального контенту може бути подана як

$$\mathrm{Content}_r(F) = \mathrm{Render}\Big(\mathrm{Explain}_r\big(\mathrm{Decompose}(F)\big)\Big) \qquad (3.26)$$

У цьому записі $Explain_r$ означає, що кількість пояснень, підказок і проміжних кроків залежить від r. Значення r може задаватися викладачем або адаптивно визначатися системою на основі стану знань студента.

Метод формування поетапного контенту також забезпечує зв’язок між теоретичним і практичним навчанням. Наприклад, після пояснення формально верифікованої теореми система може автоматично запропонувати параметризовану задачу, яка перевіряє застосування ключового кроку. Якщо студент припускається помилки, діагностична модель встановлює її причину, а система повертає його до відповідного фрагмента поетапного пояснення.

Таким чином, запропонований метод дозволяє побудувати навчальний цикл нового типу: формально верифікований артефакт → поетапний навчальний сценарій → задача → відповідь студента → діагностика → адаптивне повернення

до релевантного фрагмента. Саме цей цикл поєднує формальну строгість і педагогічну адаптивність.

Висновки до розділу 3

1. Запропоновано метод інтеграції формальної верифікації математичних доведень у навчальний процес, який використовує Lean/Mathlib як довірене джерело формальної коректності та перетворює формальні артефакти на поетапні навчальні сценарії.

2. Розроблено підхід до представлення доведення як графа залежностей і послідовності навчальних кроків, у яких кожен крок має формальну основу, дидактичне пояснення, очікувану дію студента, механізм перевірки та діагностичні ознаки можливих помилок.

3. Розроблено метод інтеграції формальної верифікації алгоритмів у навчання побудові алгоритмічних рішень, що дозволяє перейти від емпіричного тестування до навчання через специфікацію, інваріанти, доказові зобов’язання і діагностику порушених властивостей алгоритму.

4. Обґрунтовано значення формальної верифікації алгоритмів для підготовки фахівців з відмовостійких, вбудованих та критичних до безпеки систем, де коректність алгоритмічних рішень має критичний характер.

5. Запропоновано метод адаптивного вибору навчальних задач на основі функції корисності, що враховує стан знань, мотиваційний стан, історію взаємодії, діагностичні гіпотези, складність та різноманітність навчальної траєкторії.

6. Розроблено алгоритмічну модель параметричної генерації навчальних задач, яка включає шаблон, домен параметрів, предикат коректності, генератор, еталонну відповідь та процедуру перевірки студентського результату.

7. Показано, що перевірка правильності результату повинна повертати не лише бінарну оцінку, а й множину спостережуваних ознак, придатних для подальшої діагностики причин помилок.

8. Розроблено метод формування поетапного навчального контенту з формально верифікованих артефактів, що дозволяє поєднати формальну строгість, адаптивність і дидактичну доступність у межах єдиного навчального циклу.

РОЗДІЛ 4 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ТА ПРОТОТИП ІНТЕЛЕКТУАЛЬНОЇ НАВЧАЛЬНОЇ ПРОГРАМИ

4.1 Вимоги до інформаційної технології підтримки навчання точним наукам

У попередніх розділах було сформовано систему моделей і методів, що описують процес підтримки навчання точним наукам через взаємодію задач, знань, діагностики, формальної верифікації, генерації варіантів і мотиваційного регулювання. Розділ 4 присвячено переходу від цих моделей до інформаційної технології та її програмного прототипу. У цьому контексті інформаційна технологія розглядається не як окрема програмна реалізація, а як упорядкована сукупність моделей, методів, алгоритмів, структур даних, програмних компонентів і організаційних процедур, які забезпечують підтримку навчального процесу в цифровому середовищі. [91, 92]

Інформаційна технологія підтримки навчання точним наукам повинна зберігати зв’язок із системним підходом, покладеним в основу дисертаційного дослідження. Це означає, що програмний прототип має відображати не лише зовнішній інтерфейс користувача, а й машинну форму реалізації вербальної, структурної, функціональної та математичної моделей. Відповідно, кожна сутність, введена на рівні моделей, повинна мати програмне відображення: студент – у вигляді профілю й історії взаємодій; задача – у вигляді шаблону, параметрів і еталонної відповіді; знання – у вигляді компонентів і станів; діагностика – у вигляді правил, сигналів і оновлень; мотивація – у вигляді подій і винагород.

Ключовою вимогою до такої технології є підтримка адаптивного навчального циклу: система повинна зберігати інформацію про попередні дії студента, формувати або добирати наступну задачу, перевіряти відповідь, оновлювати модель знань і формувати зворотний зв’язок. На відміну від звичайної LMS, що переважно забезпечує доступ до навчальних матеріалів і

фіксацію результатів, інтелектуальна навчальна програма має підтримувати прийняття навчальних рішень на основі формалізованих моделей [93–95].

З огляду на предметну область точних наук, технологія повинна відповідати вимогам коректності, інтерпретованості та відтворюваності. Коректність означає, що система не повинна генерувати некоректні задачі або еталонні відповіді, а також не повинна давати студенту математично помилкові пояснення. Інтерпретованість означає, що рішення системи щодо добору задачі або оновлення стану знань повинні бути пояснюваними для викладача. Відтворюваність означає, що за однакових вхідних параметрів система повинна формувати однаковий або контрольовано варіативний результат.

У роботі як прототип інформаційної технології розглядається інтелектуальна навчальна програма «Мирна», побудована з використанням сучасного стеку програмної інженерії: клієнтська частина реалізована на Angular, серверна логіка ґрунтується на Java 17 і Spring, збереження даних передбачається в PostgreSQL, а процеси складання, контролю змін і розгортання підтримуються GitHub та GitHub Actions. Публічний інтерфейс прототипу розміщується за адресою https://khai-edu.github.io/its-myrna/ і використовується як демонстраційний вхід до системи [96–101].

Вимоги до інформаційної технології доцільно поділити на функціональні, нефункціональні, педагогічні, діагностичні, архітектурні й експлуатаційні. Функціональні вимоги визначають, що саме повинна виконувати система. Нефункціональні вимоги задають обмеження щодо продуктивності, надійності, масштабованості, підтримуваності та безпеки. Педагогічні вимоги визначають відповідність системи цілям навчання, а діагностичні вимоги – можливість встановлення причин помилкових відповідей за непрямими ознаками. Архітектурні вимоги забезпечують модульність системи та можливість розвитку її окремих компонентів.

Таблиця 4.1 – Основні вимоги до інформаційної технології підтримки навчання точним наукам

Група вимог	Зміст вимоги	Реалізація у прототипі «Мирна»
Функціональні	Генерація параметризованих навчальних задач, перевірка відповідей, збереження результатів, підтримка адаптивних переходів між навчальними елементами.	Модулі генерації задач, перевірки відповідей, журнал взаємодій, логіка переходів між задачами та темами.
Діагностичні	Виявлення причин помилкових відповідей за непрямими ознаками: відповідь, спосіб запису, тип помилки, історія спроб.	Діагностичне ядро, зв’язування задач із компонентами знань, локалізоване оновлення стану студента.
Педагогічні	Забезпечення навчального сценарію, підтримка поступового ускладнення, можливість формування підказок і пояснень.	Завдання подаються як частина сценарію; передбачається використання поетапного контенту й формально перевірених пояснень.

Архітектурні	Модульність, відокремлення клієнтського, серверного та рівня даних, можливість незалежного розвитку компонентів.	Angular-клієнт, Java/Spring-сервер, PostgreSQL-сховище, API-рівень взаємодії.
Експлуатаційні	Повторюваність складання, контроль версій, можливість розгортання в освітньому середовищі.	GitHub, GitHub Actions, GitHub Pages для демонстраційного розгортання клієнтського рівня.
Безпекові	Контроль доступу, захист персональних даних, обмеження адміністративних операцій, аудит подій.	Передбачається розмежування ролей, журналювання дій і можливість подальшої інтеграції механізмів автентифікації.

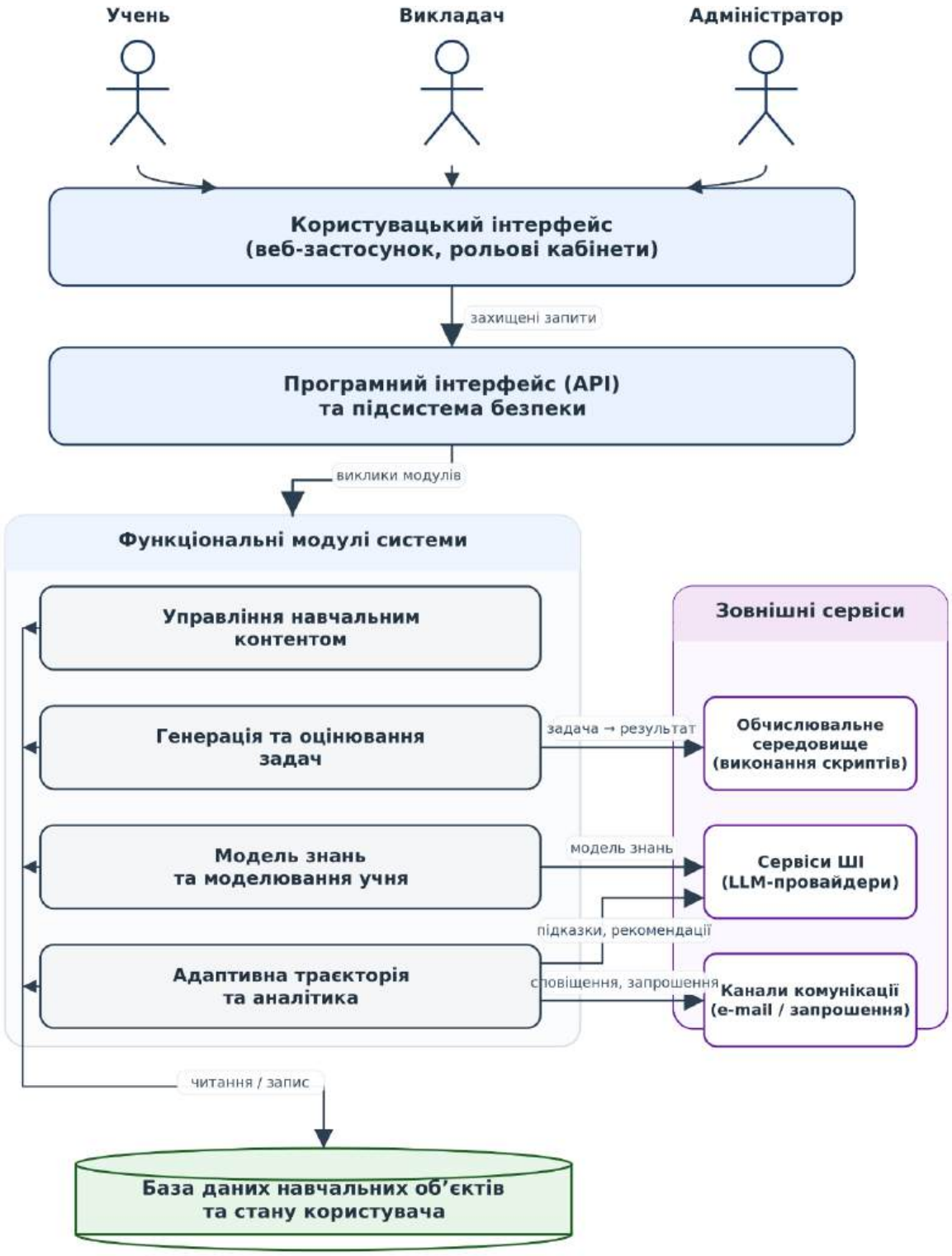


Рисунок 4.1 – Узагальнена схема інформаційної технології підтримки навчання точним наукам

Важливо, що інформаційна технологія не обмежується механізмом автоматичної перевірки. Її призначення полягає у підтримці повного циклу навчання: постановка задачі, взаємодія студента з системою, отримання відповіді, діагностика, оновлення стану знань, формування подальшого навчального впливу та мотиваційного сигналу. Такий цикл відповідає загальним принципам інтелектуальних навчальних систем, але у цій роботі він адаптований до задач точних наук, де особливе значення мають формальна коректність і алгоритмічна відтворюваність [102, 103].

Система повинна також підтримувати подальше розширення. До базового прототипу може бути додано повноцінний модуль формальної перевірки доведень, інтеграцію з Lean/Mathlib, модуль токенізації освітніх подій, розширений профіль студента, а також модуль дослідницької аналітики для викладача. Тому архітектура має будуватися з урахуванням принципів слабкого зв’язування компонентів, явного API та розділення відповідальностей між рівнями системи [104–106].

4.2 Машинна модель системи

Машинна модель є рівнем подання системи, на якому вербальні, структурні, функціональні та математичні моделі перетворюються на програмно реалізовні об’єкти. Якщо математична модель задає множини, функції, предикати, оператори оновлення та правила діагностики, то машинна модель визначає, у яких структурах даних, програмних модулях і алгоритмах ці об’єкти реалізуються. Таким чином, машинна модель є мостом між теоретичною частиною дисертації та практичним прототипом інформаційної технології.

У машинній моделі система розглядається як сукупність взаємодіючих програмних компонентів, кожен із яких виконує обмежену роль. Центральними об’єктами є користувач, навчальна тема, шаблон задачі, параметризований екземпляр задачі, відповідь студента, еталонна відповідь, компонент знань, діагностичний висновок, навчальна подія та мотиваційна подія. На рівні

реалізації ці об’єкти можуть бути подані класами, DTO, сутностями бази даних, API-контрактами або записами журналу подій.

Загально машинну модель можна подати як кортеж:

$$M = < U, T, Q, A, K, D, R, V, DB, API >, \quad (4.1)$$

де U – множина користувачів; T – множина навчальних тем; Q – множина задач; A – множина відповідей студентів; K – множина компонентів знань і станів їх засвоєння; D – множина діагностичних моделей; R – множина правил генерації та перевірки; V – множина формально верифікованих артефактів; DB – сховище даних; API – множина інтерфейсів взаємодії між клієнтським і серверним рівнями.

Таке подання дозволяє узгодити системну модель з програмною реалізацією. Наприклад, елемент Q у машинній моделі не є лише текстом умови задачі. Він включає шаблон, набір допустимих параметрів, предикат коректності, функцію генерації, еталонну відповідь, рівень складності й зв’язок із компонентами знань. Відповідно, задача у системі є не статичною одиницею навчального контенту, а параметризованим програмним об’єктом.

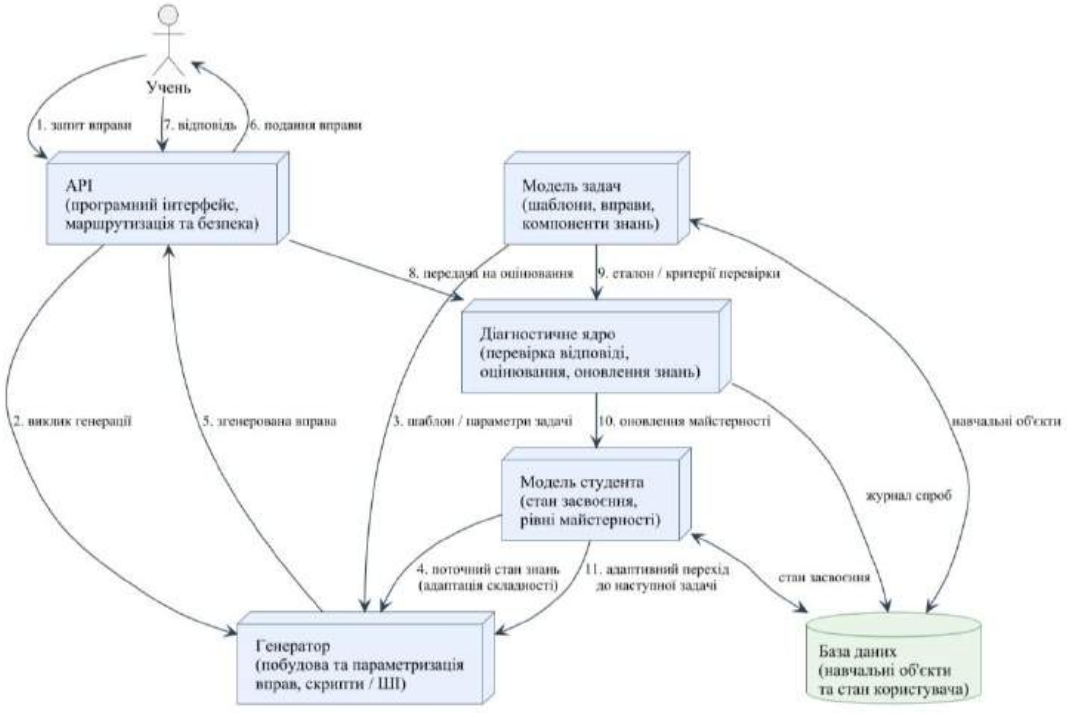


Рисунок 4.2 – Машинна модель системи підтримки навчання точним наукам

4.2.1 Логіка функціонування системи

Логіка функціонування системи реалізує замкнений навчальний цикл. На початку циклу студент обирає тему або переходить до наступного завдання за рекомендацією системи. Після цього генератор задач формує екземпляр задачі з урахуванням шаблону, параметрів і обмежень коректності. Задача подається через клієнтський інтерфейс, після чого студент вводить відповідь. Система виконує перевірку, формує результат, оновлює історію взаємодії та, за наявності діагностичної логіки, уточнює пов’язані компоненти знань.

Логіка системи має подвійну природу. З одного боку, вона реалізує класичну серверно-клієнтську взаємодію: інтерфейс формує запити до API, сервер обробляє бізнес-логіку, а база даних зберігає стани. З іншого боку, вона реалізує навчальний цикл, у якому кожна дія студента впливає на стан системи. Тому машинну логіку не можна звести лише до набору CRUD-операцій; вона повинна містити правила генерації, перевірки, діагностики й адаптації [107, 108].

Узагальнено цикл взаємодії можна описати послідовністю подій:

1. ідентифікація студента або вибір навчального профілю;
2. отримання поточного стану теми, задачі або навчального сценарію;
3. генерація або добір задачі відповідно до шаблону та поточного стану;
4. подання задачі у клієнтському інтерфейсі;
5. введення студентом відповіді або проміжного кроку;
6. перевірка відповіді за еталонним результатом або формальним артефактом;
7. формування діагностичного сигналу;
8. оновлення історії взаємодії, стану знань і мотиваційних подій;
9. вибір наступного навчального впливу.

У випадку простих задач основний акцент робиться на алгоритмічній генерації й перевірці результату. У випадку складних задач система повинна підтримувати поетапне подання навчального матеріалу, включно з поясненнями, проміжними твердженнями, контрольними питаннями та, за потреби, формально

верифікованими фрагментами. Саме тому в машинній логіці передбачено розділення між модулем генерації задач, модулем перевірки відповідей і модулем формально верифікованого контенту.

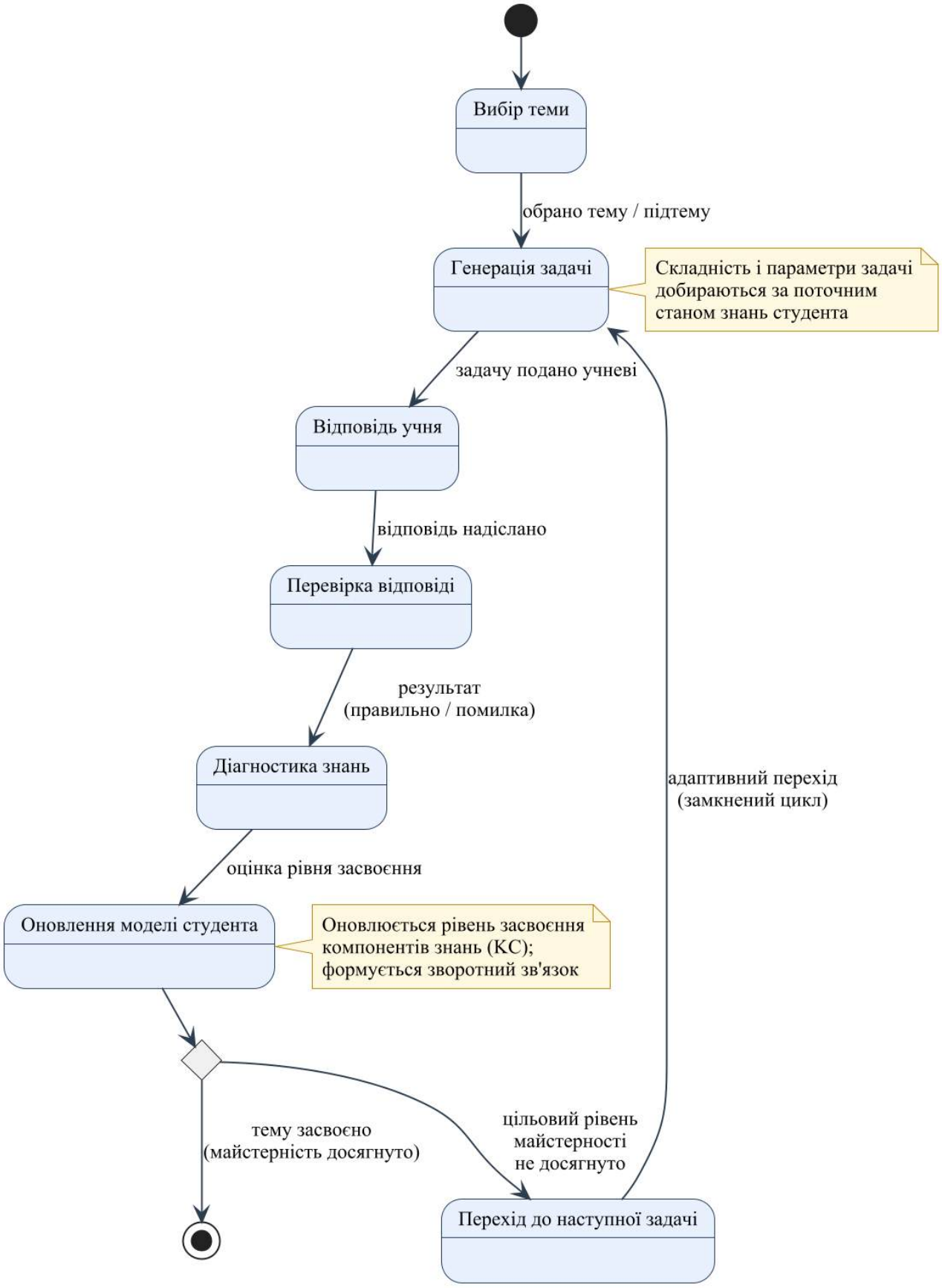


Рисунок 4.3 – Логіка функціонування системи «Мирна» як замкненого навчального циклу

4.2.2 Алгоритмічна реалізація

Алгоритмічна реалізація системи ґрунтується на модульному представленні основних операцій. Кожна операція має чітко визначені вхідні й вихідні дані. Наприклад, генерація задачі отримує шаблон і параметричний домен, а повертає екземпляр задачі й еталонну відповідь. Перевірка відповіді отримує відповідь студента та еталонний результат, а повертає статус, величину похибки або діагностичний сигнал. Оновлення стану студента отримує результат перевірки й пов’язані компоненти знань, а повертає новий стан профілю.

Алгоритм 4.1 – Узагальнений алгоритм обробки навчальної дії студента

```
Input: userId, taskId, studentAnswer, interactionData
1. userState := loadStudentState(userId)
2. task := loadTask(taskId)
3. expected := getExpectedAnswer(task)
4. checkResult := verifyAnswer(studentAnswer, expected, task.tolerance)
5. diagnosticSignal := diagnose(checkResult, studentAnswer,
task.knowledgeComponents)
6. newState := updateKnowledgeState(userState, diagnosticSignal)
7. rewardEvent := computeMotivationEvent(newState, checkResult,
interactionData)
8. saveInteraction(userId, taskId, studentAnswer, checkResult, diagnosticSignal,
rewardEvent)
9. nextTask := selectNextTask(newState, task.topic, scenarioPolicy)
Output: checkResult, feedback, rewardEvent, nextTask
```

Для задач із параметричною генерацією основний алгоритм складається з трьох етапів: вибору параметрів, перевірки їх допустимості й обчислення еталонного результату. Якщо параметри не задовольняють предикат коректності, генератор повторює вибір або використовує заздалегідь підготовлений набір допустимих комбінацій. Такий підхід дозволяє уникати

некоректних задач, вироджених випадків і ситуацій, у яких відповідь є неоднозначною.

Алгоритм 4.2 – Генерація параметризованої задачі

```
Input: taskTemplate, parameterDomain, correctnessPredicate
1. repeat
2.     params := sample(parameterDomain)
3. until correctnessPredicate(params) = true
4. problemText := instantiate(taskTemplate.text, params)
5. expectedAnswer := calculateAnswer(taskTemplate.answerFunction, params)
6. metadata := buildTaskMetadata(taskTemplate, params)
Output: problemText, expectedAnswer, metadata
```

Алгоритмічна реалізація має бути достатньо простою для подальшого супроводу і водночас відкритою до розширення. Тому в прототипі доцільно розділяти доменну логіку, рівень API, сховище даних і клієнтський інтерфейс. У подальшому це дозволяє замінювати або доповнювати окремі компоненти: наприклад, підключити повноцінний proof-checking сервіс, додати модуль токенізації або розширити діагностику без повної перебудови системи [104–106].

4.3 Архітектура інтелектуальної навчальної програми «Мирна»

Прототип інтелектуальної навчальної програми «Мирна» реалізує прикладний рівень запропонованої інформаційної технології. Його призначення полягає у демонстрації того, що розроблені моделі та методи можуть бути доведені до програмної форми: студент взаємодіє з інтерфейсом, система подає задачі, виконує перевірку відповідей, зберігає результати й підтримує логіку переходів між навчальними елементами. Архітектура прототипу орієнтована на модульність, відтворюваність і можливість поетапного нарощування функціональності.

Технологічна основа «Мирни» відповідає сучасній практиці веб-розробки освітніх систем. Клієнтська частина побудована на Angular, що дозволяє

формувати компонентний інтерфейс, маршрутизацію, сторінки задач і взаємодію з API. Серверна частина реалізується на Java 17 із використанням Spring/Spring Boot, що забезпечує побудову REST-сервісів, доменної логіки, сервісного шару та інтеграцію з базою даних. Для збереження структурованих даних використовується PostgreSQL. Контроль версій, автоматизоване складання та розгортання підтримуються GitHub і GitHub Actions, а клієнтський рівень демонстраційно розгортається на GitHub Pages за адресою https://khai-edu.github.io/its-myrna/ [96–101].

З архітектурної точки зору «Мирна» складається з чотирьох основних рівнів: клієнтського, інтеграційного, прикладного та рівня даних. Клієнтський рівень відповідає за інтерфейс користувача, відображення задач, приймання відповідей і подання зворотного зв’язку. Інтеграційний рівень реалізує API-контракти між клієнтом і сервером. Прикладний рівень містить модулі генерації, перевірки, діагностики, мотивації й роботи з навчальними сценаріями. Рівень даних забезпечує збереження профілів, задач, результатів, параметрів і журналу взаємодій.

Таблиця 4.2 – Технологічний стек прототипу «Мирна»

Рівень	Технологія	Призначення
Клієнтський рівень	Angular	Побудова інтерфейсу користувача, компонентів задач, маршрутизації, сторінок тем і взаємодії з API.
Серверний рівень	Java 17, Spring/Spring Boot	Реалізація доменної логіки, сервісів генерації й перевірки, REST API та бізнес-правил.
Рівень даних	PostgreSQL	Збереження користувачів, задач, шаблонів, параметрів, відповідей, результатів і діагностичних подій.
CI/CD	GitHub, GitHub Actions	Автоматизація складання, перевірки та розгортання компонентів системи.

Демонстраційне розгортання	GitHub Pages	Публікація клієнтського інтерфейсу прототипу для демонстрації та тестування.
Формально верифікований контент	Lean/Mathlib, HTML-матеріали	Підготовка навчальних матеріалів на основі формально доведених тверджень.

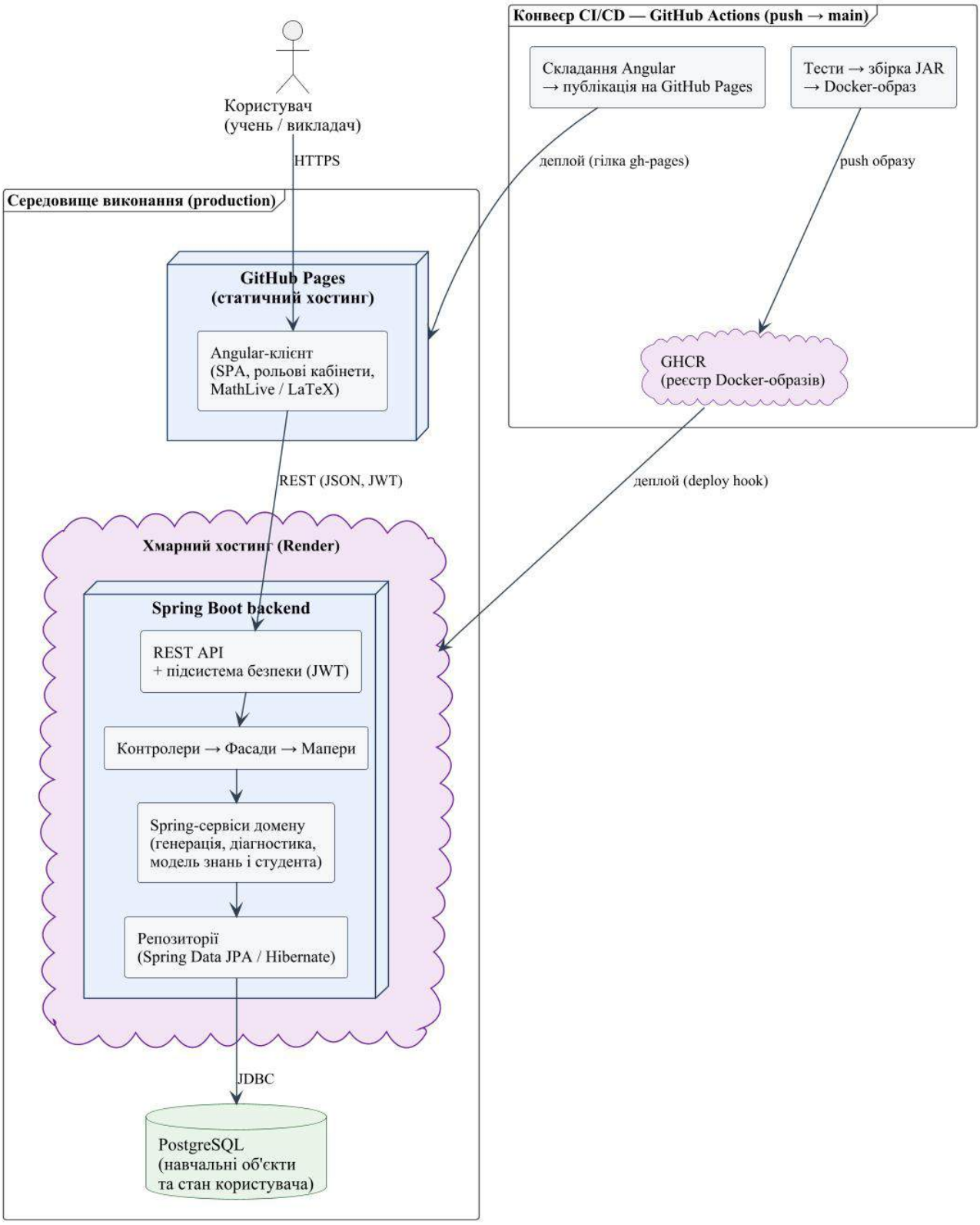


Рисунок 4.4 – Загальна архітектура інтелектуальної навчальної програми «Мирна»

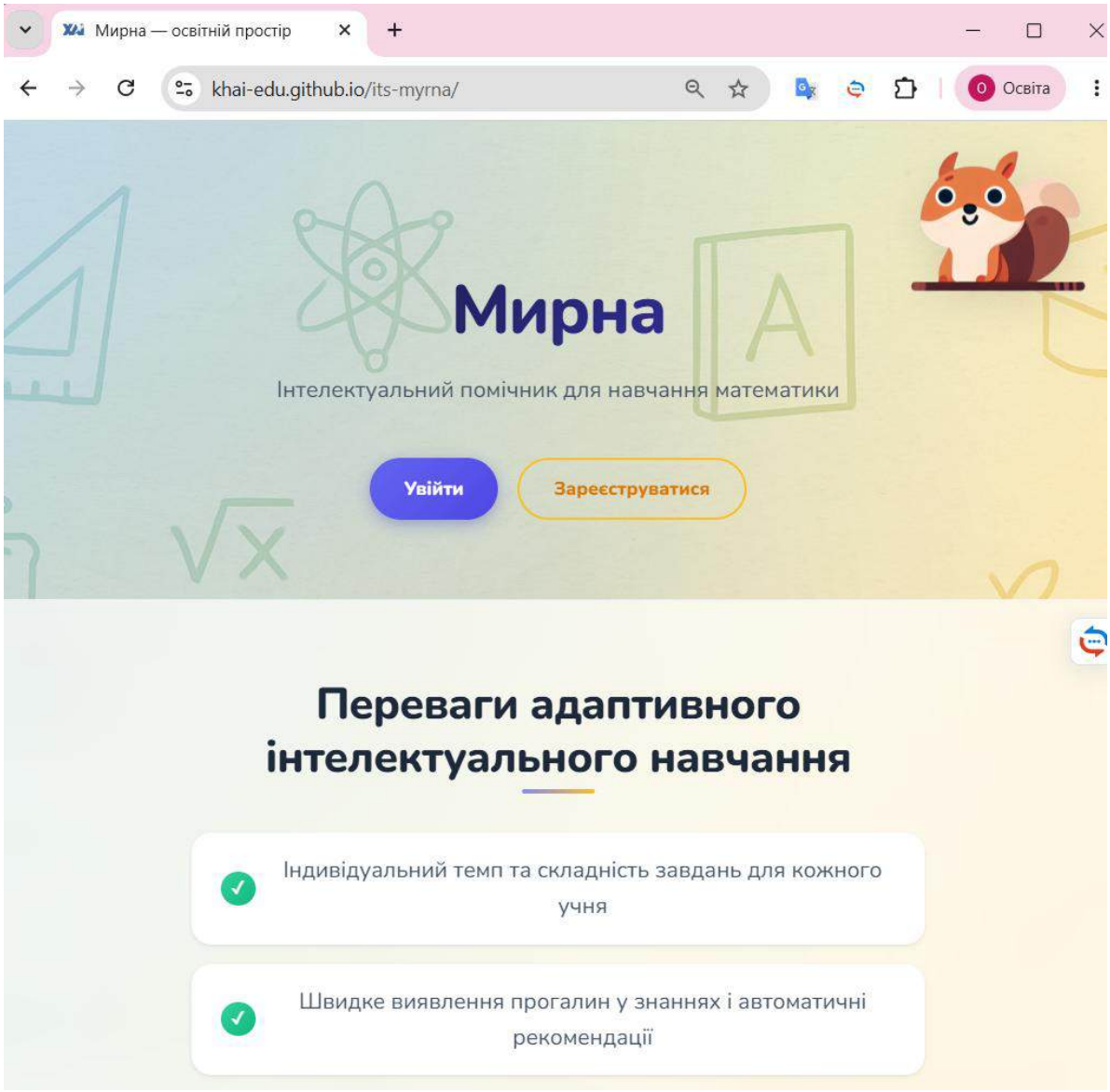


Рисунок 4.5 – Скріншот головної сторінки прототипу «Мирна» за адресою https://khai-edu.github.io/its-myrna/

4.3.1 Загальна структура системи

Загальна структура системи визначається ролями користувачів і функціональними модулями. Основною роллю є студент, який отримує задачі, виконує навчальні дії, вводить відповіді й отримує результат перевірки. Додатковою роллю є викладач або адміністратор, який може наповнювати систему шаблонами задач, контролювати структуру тем, аналізувати результати або коригувати навчальні сценарії. У базовому прототипі частина таких функцій

може бути реалізована у спрощеній формі, але архітектура повинна передбачати їх розвиток.

Система має модульну структуру. Кожен модуль працює з власною підмножиною даних, але всі модулі пов’язані через загальний стан навчального процесу. Генератор задач працює з шаблонами і параметрами; модуль перевірки – з еталонними відповідями; діагностичний модуль – з результатами, помилками й компонентами знань; мотиваційний модуль – з подіями досягнення й винагородами; модуль формально верифікованого контенту – з Lean/Mathlib-артефактами та HTML-матеріалами. Така декомпозиція дозволяє уникнути монолітного змішування навчальної логіки й технічної реалізації.

Таблиця 4.3 – Основні компоненти загальної структури системи

Компонент	Основні функції	Взаємодія
Інтерфейс користувача	Подання задач, приймання відповідей, відображення результатів, підказок і пояснень.	Взаємодіє з API, отримує дані задач і передає відповіді студента.
API-рівень	Приймання запитів, маршрутизація до сервісів, повернення результатів клієнту.	Зв’язує клієнтський рівень із сервісами генерації, перевірки, діагностики та даними.
Генератор задач	Формування індивідуальних варіантів задач на основі шаблонів і параметрів.	Отримує шаблон, перевіряє допустимість параметрів, повертає умову та еталон.
Модуль перевірки	Порівняння відповіді студента з еталоном або допустимою множиною результатів.	Передає результат перевірки до діагностичного ядра й журналу взаємодій.

Діагностичне ядро	Встановлення ймовірних причин помилок і визначення компонентів знань, що потребують оновлення.	Взаємодіє з моделлю студента, журналом відповідей і мотиваційним модулем.
Мотиваційний модуль	Формування винагород, подій прогресу, сигналів підтримки активності.	Отримує навчальні події та формує мотиваційні повідомлення або цифрові винагороди.
База даних	Збереження структурованих даних прототипу.	Використовується всіма серверними модулями.

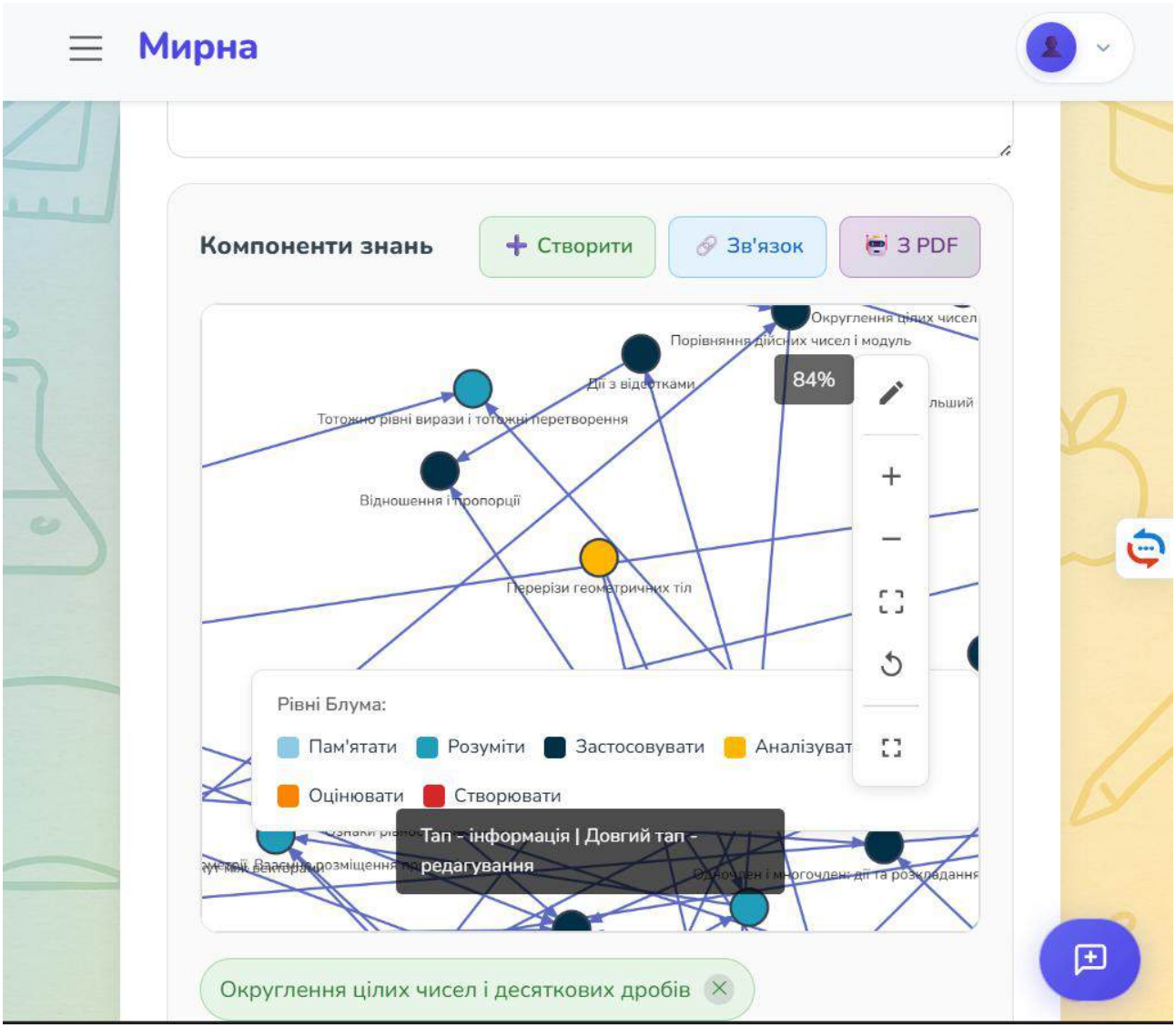


Рисунок 4.6 – Скріншот переліку навчальних тем і задач у прототипі «Мирна»

4.3.2 Клієнтська та серверна частини

Клієнтська частина системи виконує функції подання навчального контенту й організації взаємодії студента з системою. Angular забезпечує компонентний підхід до побудови інтерфейсу: сторінка задачі, список тем, форма відповіді, панель результатів, повідомлення зворотного зв’язку й навігаційні елементи можуть бути реалізовані як окремі компоненти. Це дає змогу підтримувати розширення інтерфейсу без переписування всієї клієнтської логіки [96].

Клієнтський інтерфейс повинен забезпечувати мінімум чотири режими роботи: перегляд навчальних тем, отримання задачі, введення відповіді, перегляд результату перевірки. У перспективі до цих режимів можуть бути додані поетапні доведення, Lean-орієнтовані пояснення, покрокові підказки та панель мотиваційних подій. Важливо, що інтерфейс повинен залишатися зрозумілим для студента, який не зобов'язаний знати внутрішню структуру діагностичних моделей або proof-checking систем.

Серверна частина забезпечує реалізацію прикладної логіки. Java 17 використовується як мова загального призначення з підтримкою сучасних можливостей платформи, а Spring/Spring Boot – як основа для побудови серверних сервісів, REST-контролерів, залежностей і конфігурації. Такий стек є придатним для побудови навчальних веб-систем, оскільки забезпечує модульність, підтримку багаторівневої архітектури та інтеграцію з реляційними базами даних [97–99].

API-рівень повинен бути достатньо явним, щоб клієнтська частина не залежала від внутрішньої реалізації серверних модулів. Наприклад, клієнт може надсилати запит на отримання задачі, не знаючи, чи вона була взята з бази, згенерована параметрично або сформована з формально верифікованого матеріалу. Аналогічно, відповідь студента передається у стандартному форматі, а сервер вирішує, який спосіб перевірки застосувати: пряме порівняння, числову перевірку з допуском, символьне порівняння або звернення до формально перевіреного артефакту [107, 108].

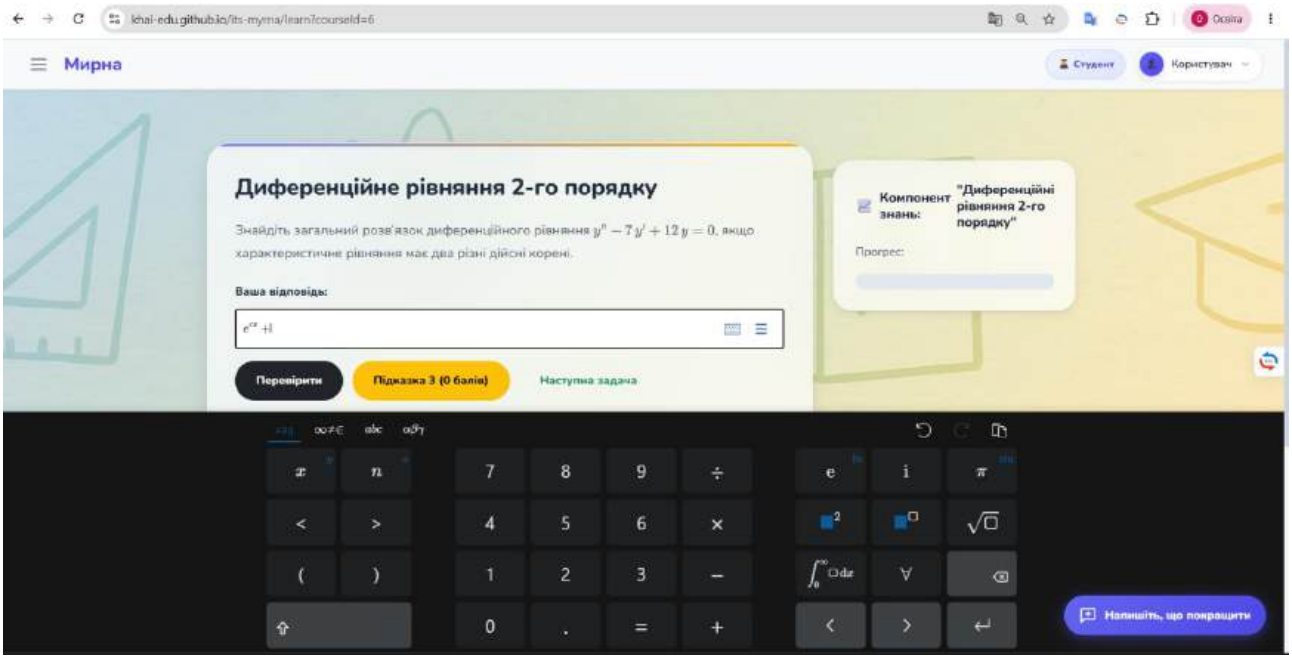


Рисунок 4.7 – Скріншот інтерфейсу задачі та форми введення відповіді у «Мирна»

4.3.3 Модель даних

Модель даних прототипу повинна забезпечувати збереження як навчального контенту, так і стану взаємодії студента із системою. На відміну від простих тестових систем, де достатньо зберігати питання, варіанти відповідей і оцінки, інтелектуальна навчальна програма потребує більш розгорнутої структури: шаблони задач, параметри, еталонні відповіді, компоненти знань, профілі студентів, спроби відповідей, діагностичні сигнали, події мотивації та, за потреби, формальні артефакти.

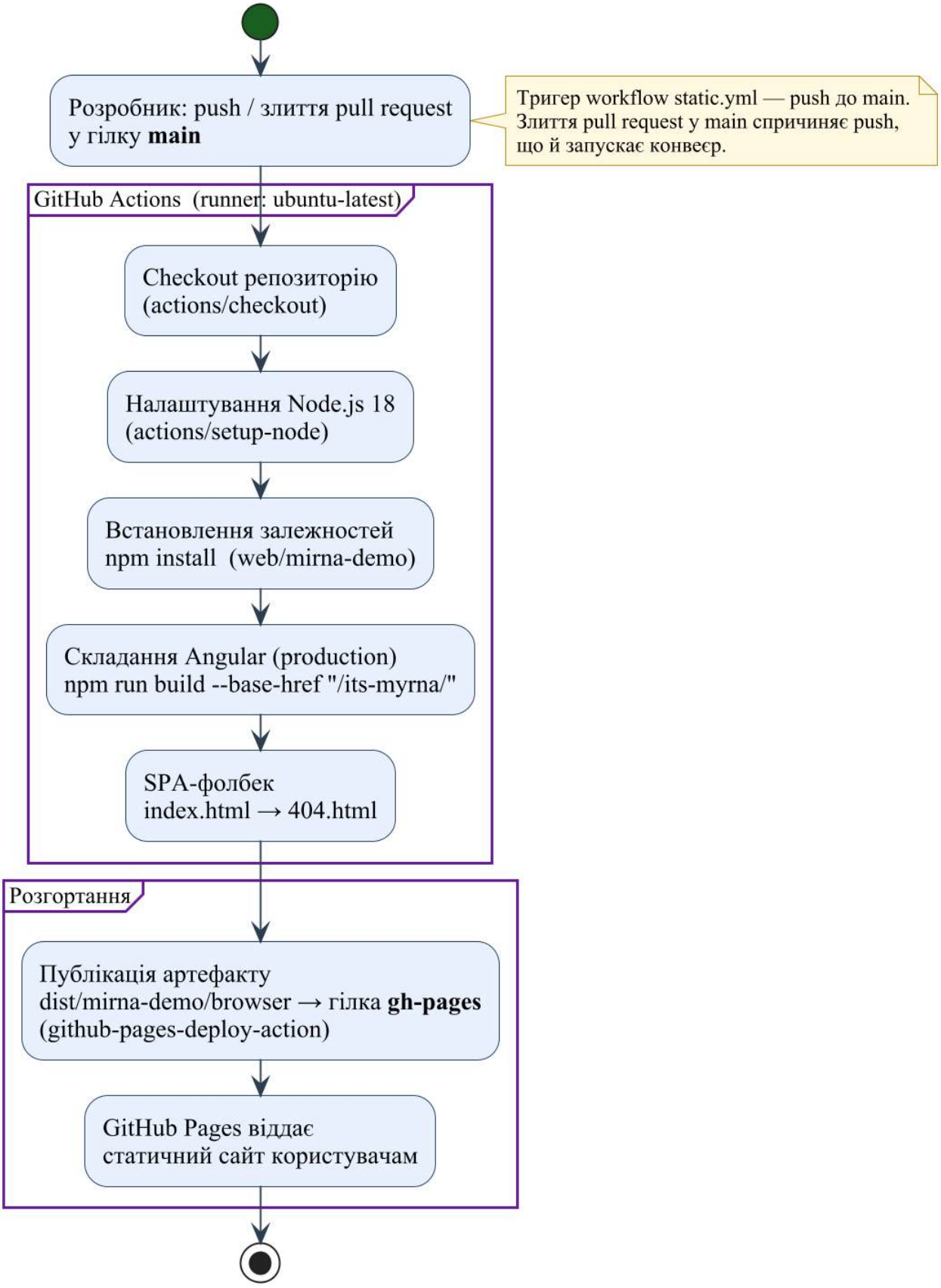


Рисунок 4.8 – Схема CI/CD-процесу розгортання клієнтської частини за допомогою GitHub Actions і GitHub Pages

PostgreSQL є доцільним вибором для зберігання таких даних, оскільки дозволяє поєднувати реляційну структуру з можливістю зберігання додаткових JSON-полів, коли окремі параметри задач або діагностичні ознаки мають змінну структуру. Це важливо для параметризованих задач, де різні шаблони можуть мати різні набори змінних, обмежень і форматів еталонної відповіді [100].

Таблиця 4.4 – Основні сутності моделі даних прототипу

Сутність	Приклади полів	Призначення
User / Student	id, name, role, profile, createdAt	Ідентифікація користувача, збереження ролі й базових параметрів профілю.
Topic	id, title, description, orderIndex	Групування задач за навчальними темами та сценаріями.
TaskTemplate	id, topicId, textTemplate, parameterSchema, answerFunction, difficulty	Опис шаблону параметризованої задачі та правил формування екземпляра.
TaskInstance	id, templateId, parameters, text, expectedAnswer, generatedAt	Конкретний згенерований варіант задачі з параметрами та еталоном.
StudentAnswer	id, userId, taskInstanceId, answer, submittedAt, attemptNumber	Фіксація відповіді студента, часу та номера спроби.
CheckResult	id, answerId, status, errorValue, feedback	Результат перевірки: правильно/неправильно, похибка, повідомлення.
Knowledge Component	id, code, title, description	Компонент знань або вмінь, що пов’язується із задачами та помилками.

DiagnosticEvent	id, answerId, componentId, errorType, confidence	Діагностичний сигнал про можливу причину помилки.
RewardEvent	id, userId, eventType, value, createdAt	Подія мотиваційного регулювання або цифрової винагороди.
FormalArtifact	id, title, system, sourceRef, htmlRef	Формально верифікований матеріал, пов’язаний із Lean/Mathlib або HTML-контентом.

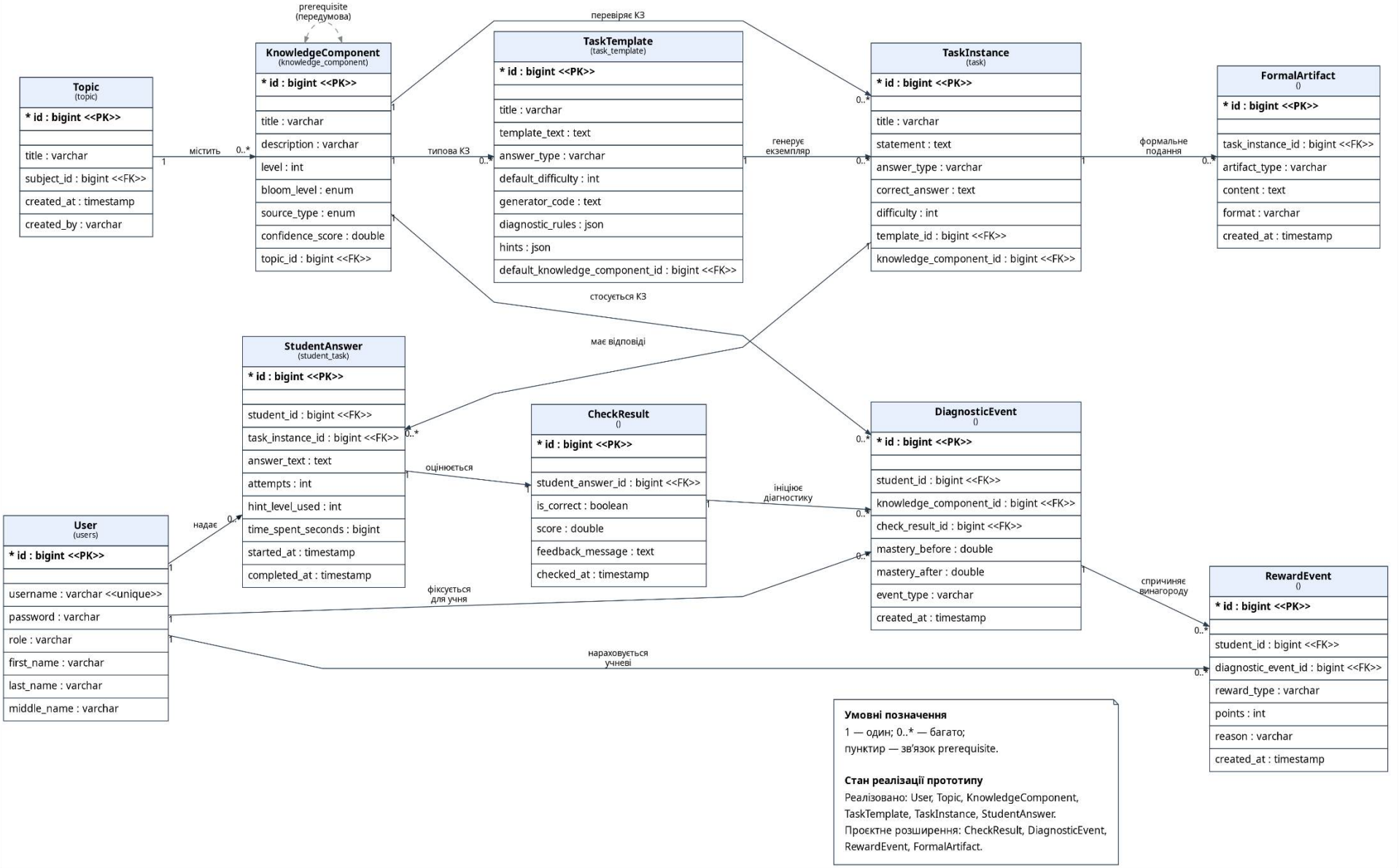


Рисунок 4.9 – ER-модель бази даних прототипу «Мирна»

Важливо, що модель даних повинна підтримувати трасованість навчальної дії. Кожна відповідь студента має бути пов’язана з конкретним екземпляром задачі, параметрами, еталонною відповіддю, результатом перевірки та можливим діагностичним висновком. Саме така трасованість дозволяє у майбутньому проводити аналіз помилок, уточнювати діагностичні моделі, оцінювати ефективність задач і формувати персоналізовані траєкторії навчання.

4.4 Реалізація основних програмних модулів

Реалізація основних програмних модулів прототипу «Мирна» відповідає модульній архітектурі інформаційної технології. Кожен модуль виконує окрему функцію, але всі модулі працюють у межах спільного навчального циклу. Такий підхід дозволяє розглядати прототип не як завершений промисловий продукт, а як інженерну реалізацію ключових наукових положень дисертації, доведену до рівня програмних компонентів.

Найбільш розвиненим практичним компонентом є генерація задач і перевірка відповідей. Саме ці функції утворюють базовий операційний рівень системи, який дозволяє студенту виконувати навчальні дії, а системі – автоматично фіксувати результат. Діагностичний модуль, модуль формально верифікованого контенту та мотиваційний модуль розглядаються як розширення базового циклу, що додають інтелектуальність, педагогічну глибину й можливість подальшої персоналізації

4.4.1 Модуль генерації задач

Модуль генерації задач відповідає за створення індивідуальних варіантів навчальних задач на основі шаблонів і параметрів. У прототипі така генерація є алгоритмічною: система не просто випадково змінює числа в умові, а повинна враховувати допустимість параметрів, уникати вироджених випадків і забезпечувати наявність еталонної відповіді. Це принципово відрізняє генерацію навчальних задач від генерації довільного тексту, оскільки для точних наук коректність є обов’язковою умовою.

Шаблон задачі задає текстову структуру, змінні, правила підстановки й функцію обчислення відповіді. Домен параметрів визначає допустимі значення. Предикат коректності відсікає комбінації, що призводять до некоректної або педагогічно непридатної задачі. Наприклад, у задачах лінійної алгебри слід уникати нульового визначника, якщо задача передбачає застосування методу Крамера; у задачах на дроби – уникати нульового знаменника; у задачах на логарифми – контролювати область визначення.

У програмній реалізації шаблон задачі може бути представлений як об’єкт, що містить текстову частину, схему параметрів, функцію генерації, функцію перевірки параметрів і функцію обчислення еталонної відповіді. Така структура дозволяє створювати нові типи задач без зміни загальної логіки системи.

Алгоритм 4.3 – Структура роботи модуля генерації задач

```
Input: topicId, studentState, generationPolicy
templates := findTemplatesByTopic(topicId)
template := selectTemplate(templates, studentState, generationPolicy)
params := generateParameters(template.parameterSchema)
if not isCorrect(params, template.correctnessPredicate) then repeat step 3
text := renderTemplate(template.textTemplate, params)
expected := calculateExpectedAnswer(template.answerFunction, params)
instance := saveTaskInstance(template, params, text, expected)
Output: instance
```

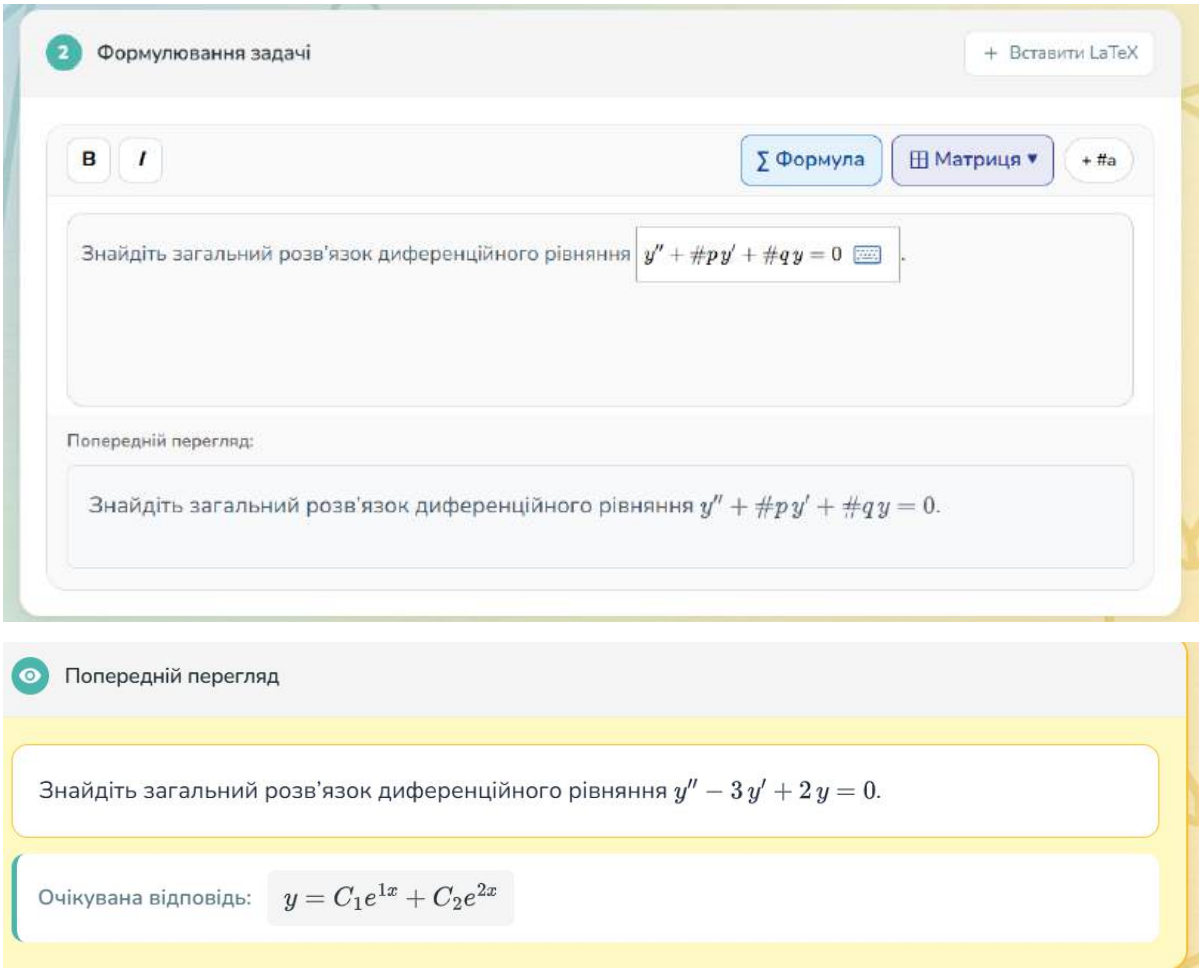


Рисунок 4.10 – Скріншот згенерованої задачі у прототипі «Мирна»

4.4.2 Модуль перевірки відповідей

Модуль перевірки відповідей є базовим механізмом зворотного зв'язку в системі. Його завдання полягає не лише в тому, щоб повідомити студенту, правильна чи неправильна відповідь, а й у тому, щоб сформувати структурований результат перевірки, придатний для подальшої діагностики. Для простих задач перевірка може виконуватися прямим порівнянням із еталоном або порівнянням із допустимою похибкою. Для задач із символьними відповідями потрібні додаткові правила нормалізації.

Під час перевірки числових відповідей система повинна враховувати формат введення, розділювачі, округлення, допустиму похибку та можливі еквівалентні форми запису. Для точних наук така деталізація є важливою, оскільки технічна помилка введення не завжди тотожна предметній помилці. Саме тому результат перевірки повинен містити не лише статус, а й тип

невідповідності: неправильне значення, неправильний формат, перевищення допустимої похибки, відсутність відповіді або помилка синтаксису.

Результат перевірки передається до діагностичного ядра. Якщо відповідь правильна, система може підвищити оцінку пов’язаних компонентів знань і сформувати позитивний мотиваційний сигнал. Якщо відповідь неправильна, система повинна спробувати встановити причину помилки. Таким чином, модуль перевірки є входом до діагностичного процесу.

Таблиця 4.5 – Типи результатів перевірки відповідей

Тип результату	Опис	Подальша дія системи
Correct	Відповідь збігається з еталоном або потрапляє в допустимий інтервал.	Фіксація успіху, оновлення стану знань, мотиваційний сигнал.
IncorrectValue	Формат відповіді правильний, але значення не збігається з еталоном.	Передача до діагностичного ядра для встановлення можливої причини.
InvalidFormat	Відповідь не може бути інтерпретована системою через формат або синтаксис.	Повідомлення про помилку введення, можливе повторне введення без штрафу.
OutOfTolerance	Числовий результат близький до правильного, але перевищує допустиму похибку.	Формування уточнювального зворотного зв’язку.
NoAnswer	Відповідь не подана або спроба завершена без результату.	Фіксація незавершеної взаємодії, можливість підказки або повторної спроби.

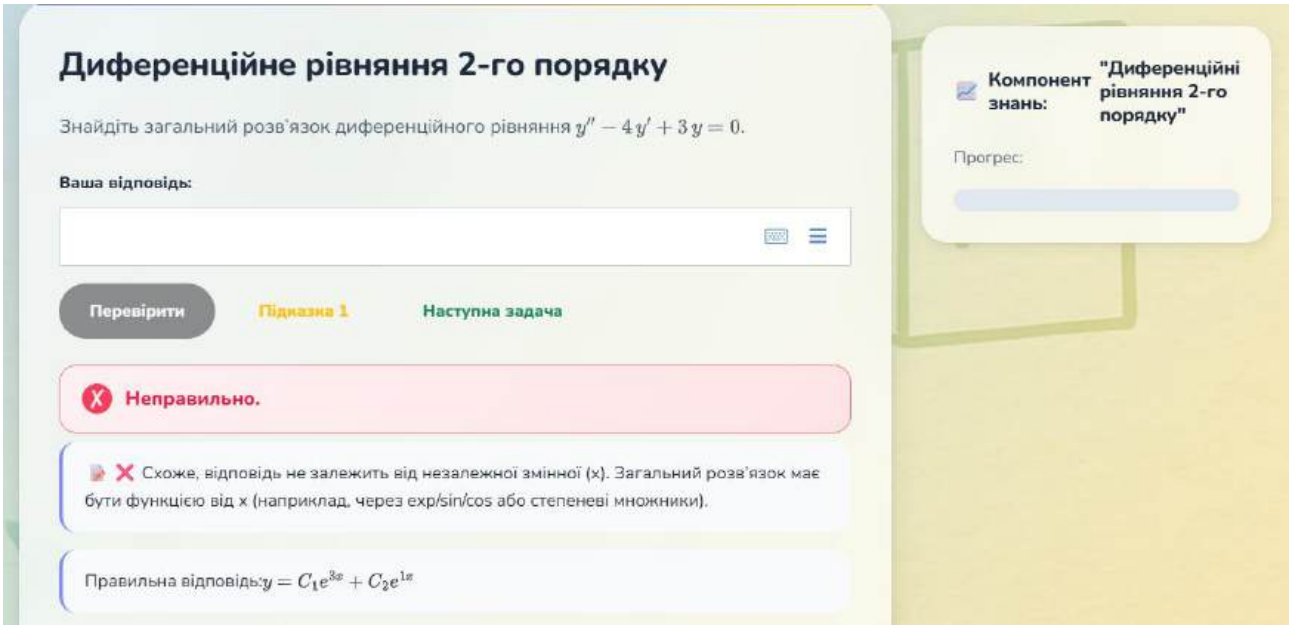


Рисунок 4.11 – Скріншот результату перевірки відповіді у системі «Мирна»

4.4.3 Модуль діагностики знань

Модуль діагностики знань реалізує перехід від факту помилки до її можливої причини. На відміну від звичайної перевірки, діагностика працює з непрямими ознаками: значенням відповіді, способом запису, типом відхилення, історією попередніх спроб, часом виконання та зв’язком задачі з компонентами знань. У базовому прототипі діагностика може бути реалізована спрощено – через прив’язування задачі до одного або кількох компонентів знань. У розширеній версії вона повинна працювати з набором діагностичних моделей типових помилок.

Діагностичний модуль не повинен робити надмірно загальних висновків. Неправильна відповідь не означає, що студент не знає всю тему. Вона може свідчити про локальну помилку: неправильний знак, некоректне застосування правила, помилку округлення, нерозуміння окремого поняття або збій у проміжному кроці. Тому результат діагностики має бути локалізованим і пов’язаним із конкретними компонентами знань.

У машинній реалізації діагностичний модуль може працювати у три етапи. Спочатку він отримує результат перевірки та контекст задачі. Далі визначає множину можливих причин помилки. Після цього формує діагностичний сигнал,

який передається до модуля оновлення стану знань. Такий сигнал може містити ідентифікатор компонента знань, тип помилки, рівень упевненості та рекомендовану дію.

Алгоритм 4.4 – Формування діагностичного сигналу

Input: checkResult, taskMetadata, studentAnswer, interactionHistory
1. if checkResult.status = Correct then return positiveDiagnosticSignal
2. features := extractErrorFeatures(studentAnswer, checkResult, interactionHistory)
3. candidateCauses := matchDiagnosticModels(features, taskMetadata.knowledgeComponents)
4. cause := selectMostProbableCause(candidateCauses)
5. signal := buildDiagnosticSignal(cause, taskMetadata.knowledgeComponents, confidence)
Output: signal

4.4.4 Модуль формально верифікованого навчального контенту

Модуль формально верифікованого навчального контенту призначений для роботи зі складними задачами, де важливим є не лише кінцевий результат, а й структура доведення або коректність алгоритму. У межах прототипу цей модуль представлено через підготовку HTML-матеріалів на основі формально перевірених тверджень у Lean/Mathlib. Його призначення полягає в тому, щоб перетворити формальний артефакт на педагогічно зрозумілий поетапний сценарій.

Основна ідея полягає в тому, що формально перевірений доказ або фрагмент математичної теорії може бути використаний як джерело надійного навчального матеріалу. Формальна система гарантує, що твердження та зв’язки між ними не суперечать формальній логіці. Навчальна система, у свою чергу, перетворює цей матеріал на пояснення, послідовність кроків, запитання для самоперевірки й проміжні підказки [111–115].

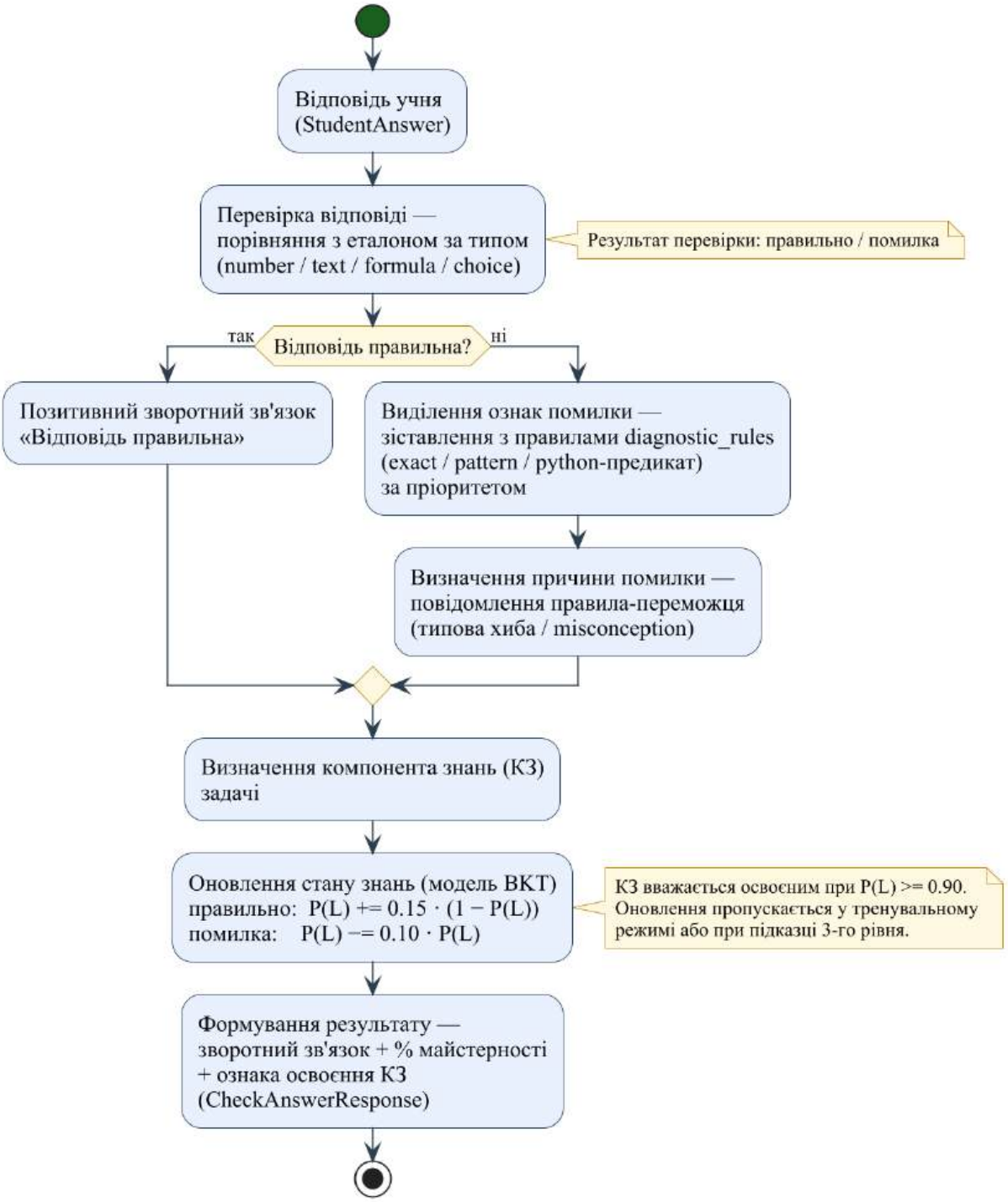


Рисунок 4.12 – Схема діагностичного модуля у прототипі «Мирна»

У практичному кейсі для ілюстрації такого підходу використано метод Крамера. Формальна база Mathlib забезпечує коректність відповідних математичних понять, а навчальний HTML-матеріал подає студенту поетапне пояснення: постановка системи лінійних рівнянь, умова невиродженості матриці, роль визначника, побудова матриць із заміною стовпців, формула для

невідомих та логіка доведення. Важливо, що науковий внесок полягає не у повторному доведенні відомої теореми, а в інтеграції формально верифікованого артефакту в навчальний процес.

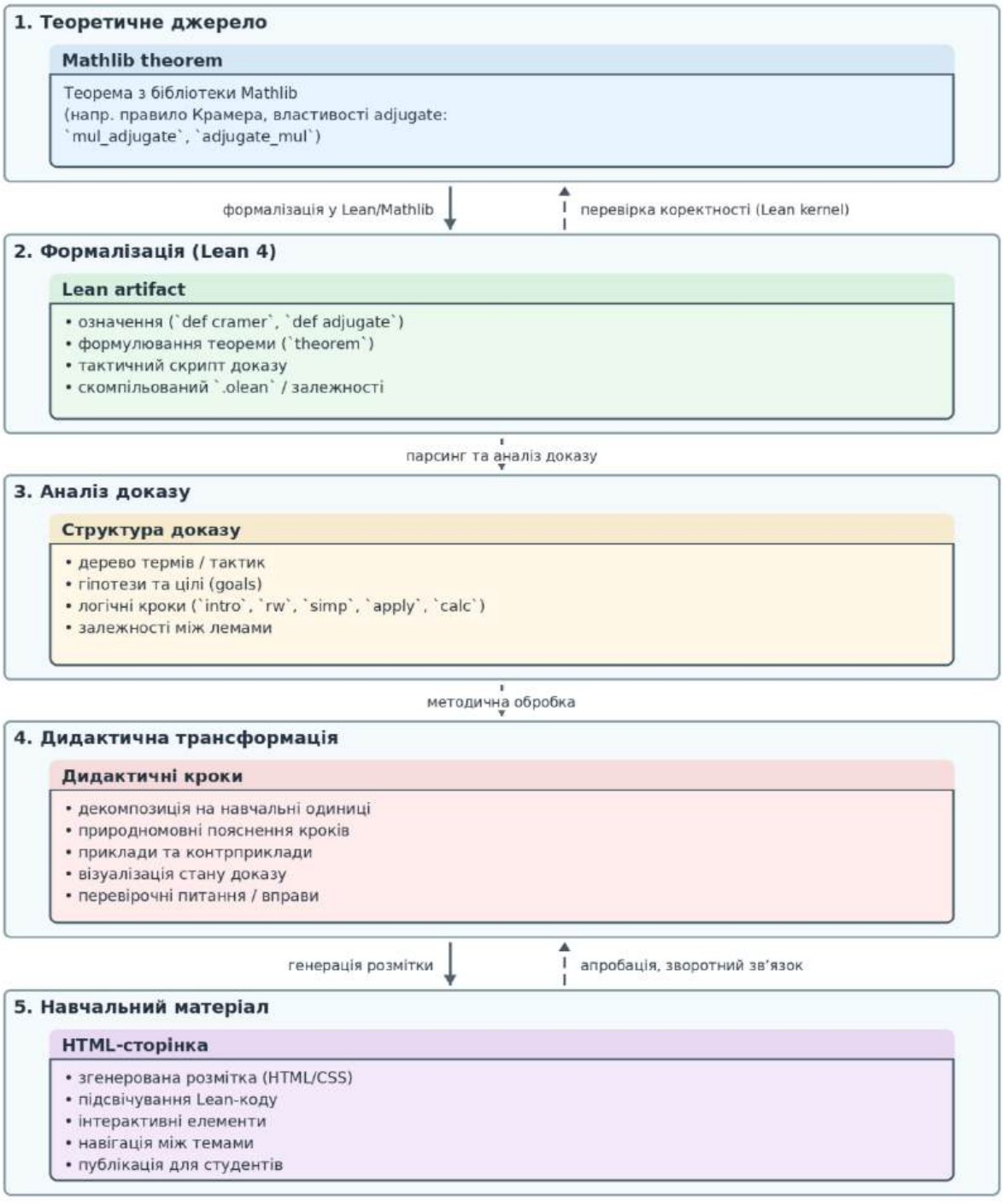


Рисунок 4.13 – Схема формування навчального HTML-матеріалу на основі Lean/Mathlib

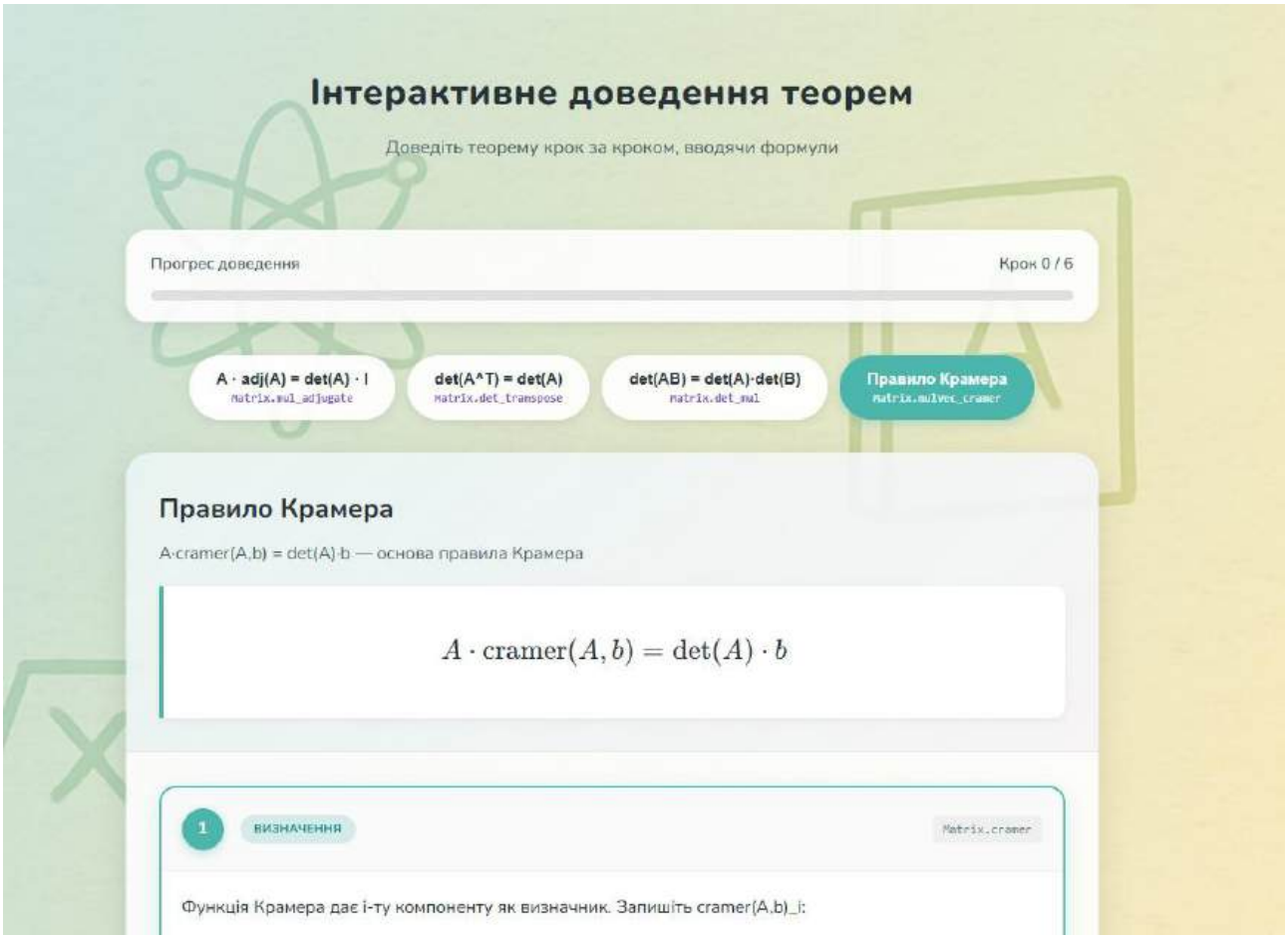


Рисунок 4.14 – Скріншот HTML-матеріалу з поетапним поясненням методу Крамера

4.4.5 Модуль мотиваційного регулювання та цифрових винагород

Модуль мотиваційного регулювання призначений для формування додаткового навчального стимулу на основі результатів взаємодії студента із системою. На відміну від простого нарахування балів за правильні відповіді, запропонований підхід передбачає зв’язування мотиваційної події зі станом знань, діагностичним результатом і навчальним прогресом. У такій моделі винагорода має не лише зовнішню, а й змістову функцію: вона фіксує навчальну цінність події.

У прототипі мотиваційний модуль може бути реалізований як журнал подій, у якому фіксуються успіхи, відновлення після помилки, проходження складної задачі, зменшення кількості підказок, послідовна робота над темою або

досягнення цільового рівня. У перспективі такі події можуть бути перетворені на цифрові винагороди або sense-коїни, а їх облік може здійснюватися через смартконтракти або permissioned blockchain-рівень. У дисертації цей компонент розглядається як модель подальшого розвитку системи, а не як обов'язкова умова базової працездатності прототипу [116–119].

Ключовим є те, що мотиваційний сигнал повинен спиратися на педагогічно значущу подію. Нарахування винагороди лише за кількість кліків або механічне проходження задач не відповідає меті системи. Натомість система повинна підтримувати події типу: «студент виправив типову помилку», «студент самостійно виконав задачу після попередньої невдачі», «студент засвоїв компонент знань, який раніше був проблемним». Саме такі події пов'язують мотивацію із внутрішнім прогресом.

Таблиця 4.6 – Приклади мотиваційних подій у системі

Подія	Умова виникнення	Мотиваційний зміст
CorrectFirstTry	Студент правильно виконав задачу з першої спроби.	Підтвердження стійкого засвоєння відповідного компонента.
RecoveryAfterError	Студент виправив помилку після діагностичного зворотного зв'язку.	Підтримка навчальної наполегливості й роботи з помилками.
HintReduction	Кількість підказок зменшилась порівняно з попередніми спробами.	Підтвердження зростання самостійності.
HardTaskSolved	Студент розв'язав задачу підвищеної складності.	Підкріплення переходу до складніших сценаріїв.
ProofStepCompleted	Студент правильно виконав крок у поетапному доведенні.	Підтримка формального мислення й логічної послідовності.

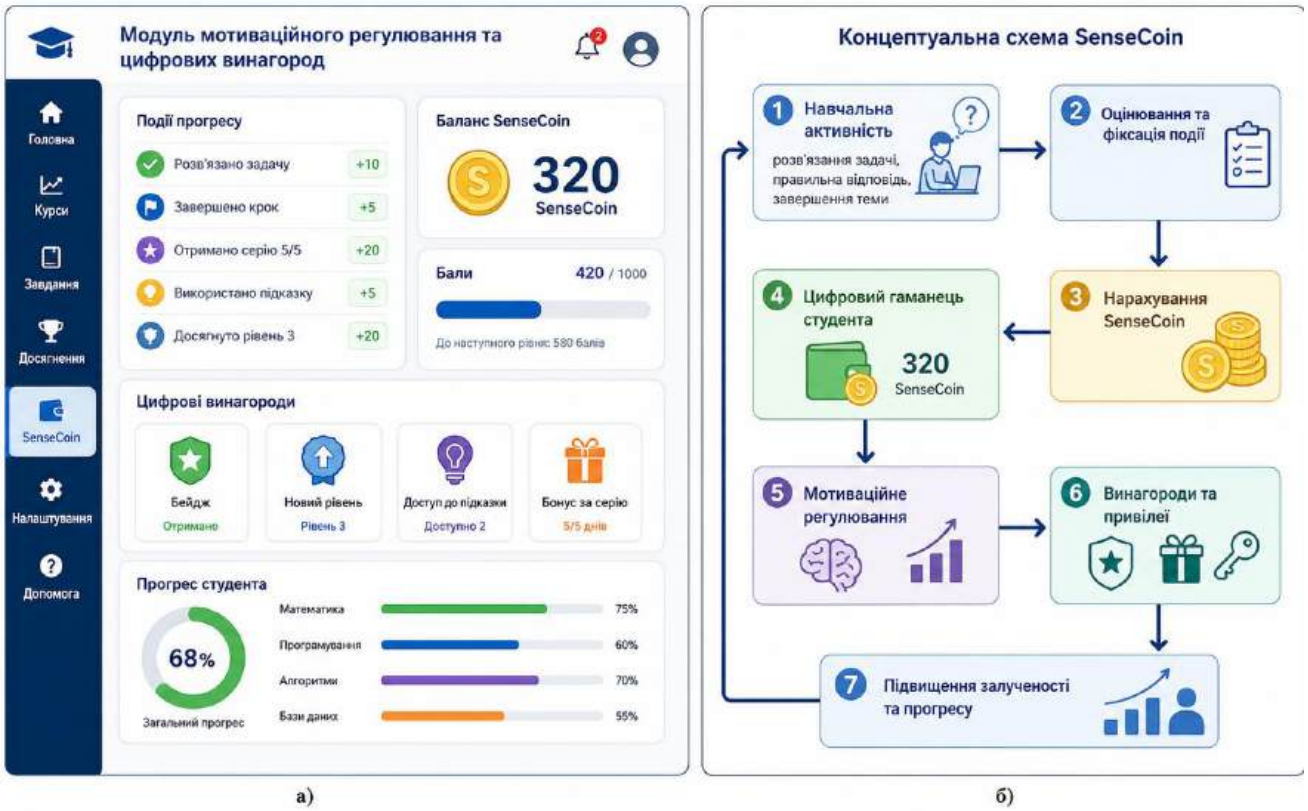


Рисунок 4.15 – Скріншот та схема модуля мотиваційного регулювання та цифрових винагород

4.5 Прототип використання формальної верифікації у навчанні

Окремим компонентом практичної частини є прототип використання формальної верифікації у навчанні. Він демонструє, як формально перевірені математичні твердження можуть бути використані для створення навчального матеріалу. На відміну від класичних електронних конспектів, такий матеріал спирається на формальний артефакт, який пройшов перевірку в системі доведення. Це дозволяє знизити ризик некоректних пояснень у складних теоретичних темах.

У межах роботи прототип побудовано на прикладі методу Крамера. Метод Крамера є зручним навчальним прикладом, оскільки поєднує обчислювальну процедуру, теоретичну умову застосовності, поняття визначника та структуру доведення. Він дозволяє показати різницю між простим застосуванням формули й розумінням того, чому ця формула є коректною.

4.5.1 Формалізація методу Крамера

Формалізація методу Крамера у навчальному контексті передбачає не лише подання формули, а й встановлення її умов. Для системи лінійних рівнянь із матрицею коефіцієнтів A і вектором правих частин b метод Крамера застосовується за умови, що визначник матриці A не дорівнює нулю. Кожна невідома обчислюється як відношення визначника матриці, отриманої заміною відповідного стовпця на b, до визначника A.

У Lean/Mathlib відповідні математичні твердження формалізуються через наявні структури лінійної алгебри, матриць, детермінантів і розв'язків систем. Використання Mathlib є принципово важливим: воно дозволяє спиратися на вже верифіковану формальну базу, а не створювати всю теорію заново. У дисертаційному дослідженні такий підхід трактується як використання формальної бібліотеки для побудови навчального пайплайна [111–115].

```
variable (A : Matrix n n α) (b : n → α)

/-- `cramerMap A b i` is the determinant of the matrix `A` with column `i` replaced with `b`,
  and thus `cramerMap A b` is the vector output by Cramer's rule on `A` and `b`.

  If `A * x = b` has a unique solution in `x`, `cramerMap A` sends the vector `b` to `A.det • x`.
  Otherwise, the outcome of `cramerMap` is well-defined but not necessarily useful.
-/
def cramerMap (i : n) : α :=
  (A.updateCol i b).det

theorem cramerMap_is_linear (i : n) : IsLinearMap α fun b => cramerMap A b i :=
  { map_add := det_updateCol_add _ _
    map_smul := det_updateCol_smul _ _ }

theorem cramer_is_linear : IsLinearMap α (cramerMap A) := by
  constructor <;> intros <;> ext i
  · apply (cramerMap_is_linear A i).1
  · apply (cramerMap_is_linear A i).2

/-- `cramer A b i` is the determinant of the matrix `A` with column `i` replaced with `b`,
  and thus `cramer A b` is the vector output by Cramer's rule on `A` and `b`.

  If `A * x = b` has a unique solution in `x`, `cramer A` sends the vector `b` to `A.det • x`.
```

Рисунок 4.16 – Фрагмент Lean/Mathlib-формалізації методу Крамера

Для навчальної системи важливо виділити не весь формальний код, а його дидактичну структуру. Студенту не обов’язково бачити всі технічні деталі формальної мови. Натомість система повинна подати ключові кроки: постановка задачі, перевірка умови невиродженості, заміна стовпця, обчислення визначника, отримання формули, інтерпретація результату. Саме такий перехід від формального артефакту до навчального сценарію є практичним результатом інтеграції proof-checking у навчання.

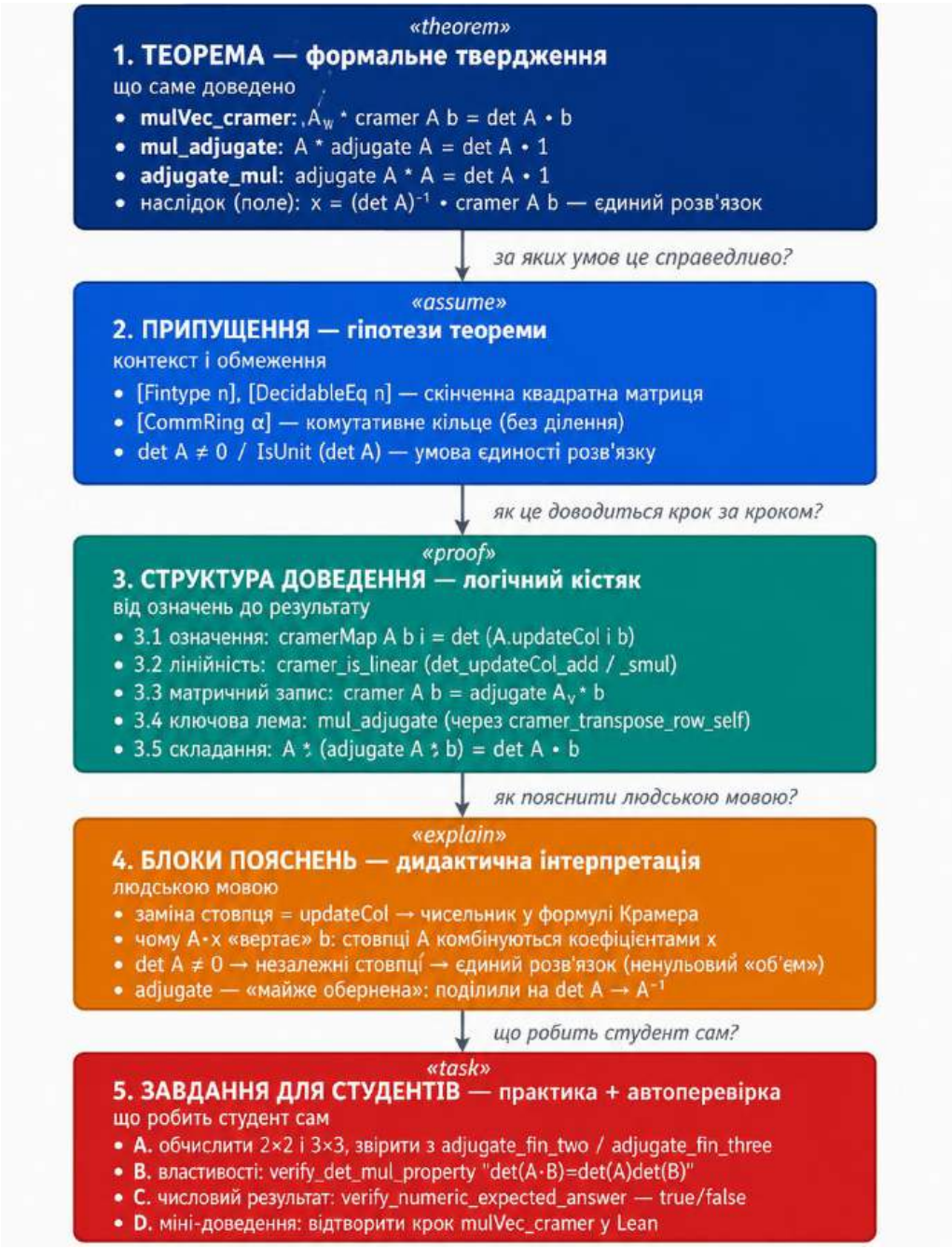


Рисунок 4.17 – Схема переходу від формального твердження до дидактичних кроків методу Крамера

4.5.2 Формування HTML-матеріалу на основі Lean

Формування HTML-матеріалу на основі Lean полягає у перетворенні формального або напівформального опису доказу на сторінку, придатну для навчального використання. Така сторінка повинна містити загальну постановку задачі, теоретичні передумови, послідовність кроків, пояснення математичного змісту, можливі помилки й контрольні запитання. У прототипі така сторінка може бути згенерована на основі формалізованої структури та додаткового дидактичного опису.

Для студента HTML-матеріал є не доказом у вузькому формальному сенсі, а інтерфейсом до формально верифікованого змісту. Це важливе розмежування. Формальна система забезпечує строгість, а навчальна сторінка забезпечує зрозумілість. Таким чином, прототип поєднує два рівні: рівень формальної коректності та рівень педагогічної подачі матеріалу.

У HTML-матеріалі доцільно виділяти такі блоки: “Ідея методу”, “Формальна умова застосовності”, “Покрокове доведення”, “Типові помилки”, “Зв’язок із задачами”, “Контрольні запитання”. Це дозволяє інтегрувати матеріал у навчальний сценарій і використовувати його не лише як довідку, а як активний елемент навчання.

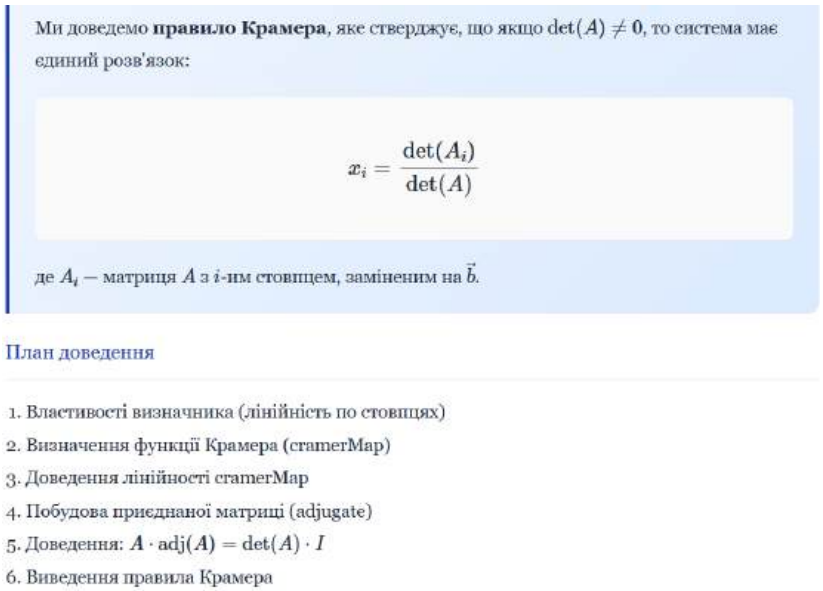


Рисунок 4.18 – Скріншот HTML-сторінки навчального матеріалу на основі Lean: загальний вигляд

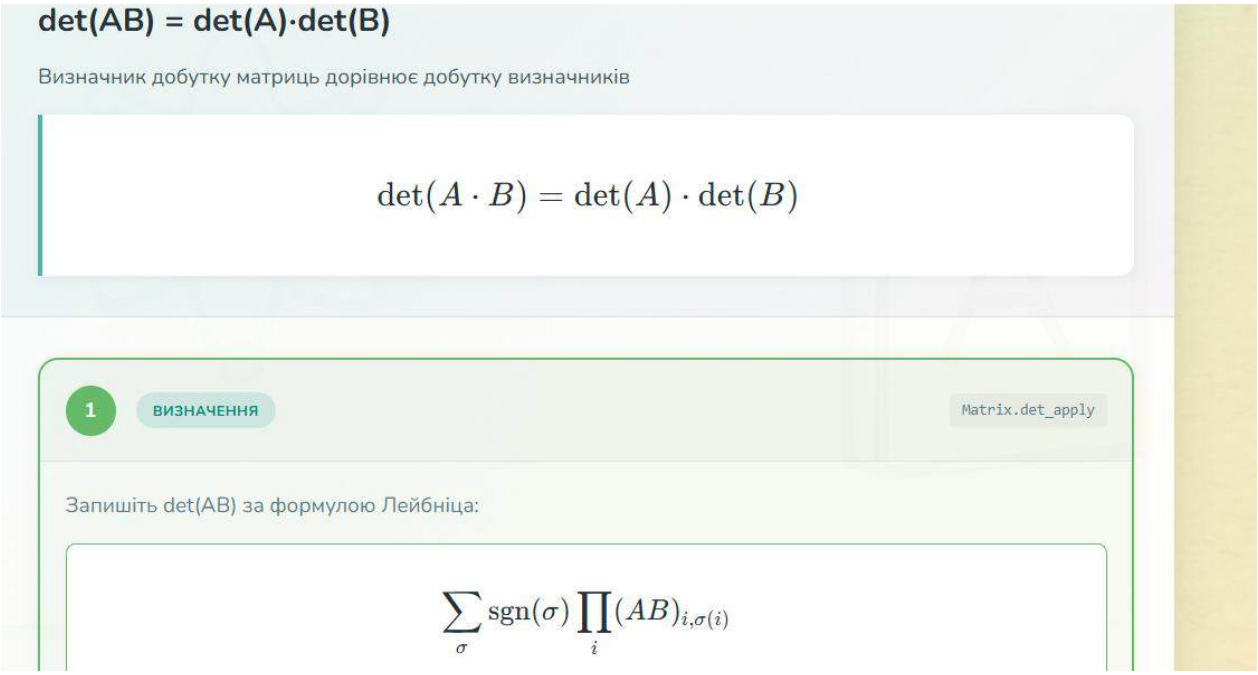


Рисунок 4.19 – Скріншот HTML-сторінки: блок поетапного доведення методу Крамера

Рисунок 4.20 – Скріншот HTML-сторінки: зв’язок формального артефакту з навчальним поясненням

4.6 Оцінка працездатності прототипу та перспективи розвитку

Оцінка працездатності прототипу у межах цієї роботи спрямована на перевірку того, чи реалізовані основні функціональні контури інформаційної технології: генерація задач, перевірка відповідей, збереження результатів,

підтримка адаптивної логіки, подання формально верифікованого матеріалу та можливість подальшого розширення системи. Така оцінка не підміняє повномасштабний педагогічний експеримент, але підтверджує інженерну здійсненність запропонованих моделей і методів.

Працездатність прототипу може бути перевірена через функціональне тестування основних сценаріїв. До таких сценаріїв належать: відкриття клієнтського інтерфейсу, завантаження теми, генерація задачі, введення правильної відповіді, введення неправильної відповіді, фіксація результату, повторна спроба, відображення підказки або пояснення, перехід до наступного завдання. Окремо перевіряється можливість відкриття HTML-матеріалу, сформованого на основі Lean/Mathlib-артефакту.

Таблиця 4.7 – Сценарії оцінки працездатності прототипу

Сценарій	Очікуваний результат	Критерій успішності
Відкриття клієнтської частини	Користувач бачить головний екран системи.	Сторінка завантажується без критичних помилок.
Вибір теми	Система відображає доступні задачі або навчальні елементи.	Навігація працює коректно.
Генерація задачі	Система створює індивідуальний варіант задачі.	Умова задачі має коректні параметри та еталонну відповідь.
Перевірка правильної відповіді	Система повідомляє про правильний результат.	Статус перевірки відповідає еталону.
Перевірка неправильної відповіді	Система повідомляє про помилку та формує можливий діагностичний сигнал.	Помилка фіксується в журналі взаємодії.

Перехід до наступної задачі	Система формує наступний навчальний крок.	Перехід відповідає логіці сценарію.
Відкриття Lean/HTML-матеріалу	Сторінка з поетапним поясненням доступна користувачу.	Матеріал відображається без помилок структури.
Розгортання через CI/CD	Після зміни коду виконується автоматичне складання і публікація.	GitHub Actions завершується успішно, клієнтська частина доступна.

Перевагою прототипу є те, що він поєднує реальну програмну реалізацію з можливістю подальшого розвитку наукових компонентів. Навіть якщо окремі модулі, наприклад глибока діагностика за деревом виразу або повна інтеграція Lean у LMS, реалізовані на рівні прототипу або демонстраційного кейсу, архітектура передбачає їх поступове нарощування. Це відповідає інженерній логіці створення складних інформаційних систем: спочатку реалізується робоче ядро, а потім розширюються спеціалізовані інтелектуальні модулі.

Перспективи розвитку системи включають п’ять напрямів. Перший – поглиблення діагностичного ядра через аналіз структури відповідей і типових помилок. Другий – повніша інтеграція формальної верифікації з навчальним інтерфейсом. Третій – розширення параметричного банку задач для різних тем точних наук. Четвертий – впровадження мотиваційного модуля з цифровими винагородами. П’ятий – проведення педагогічного експерименту для оцінювання впливу системи на якість навчання.

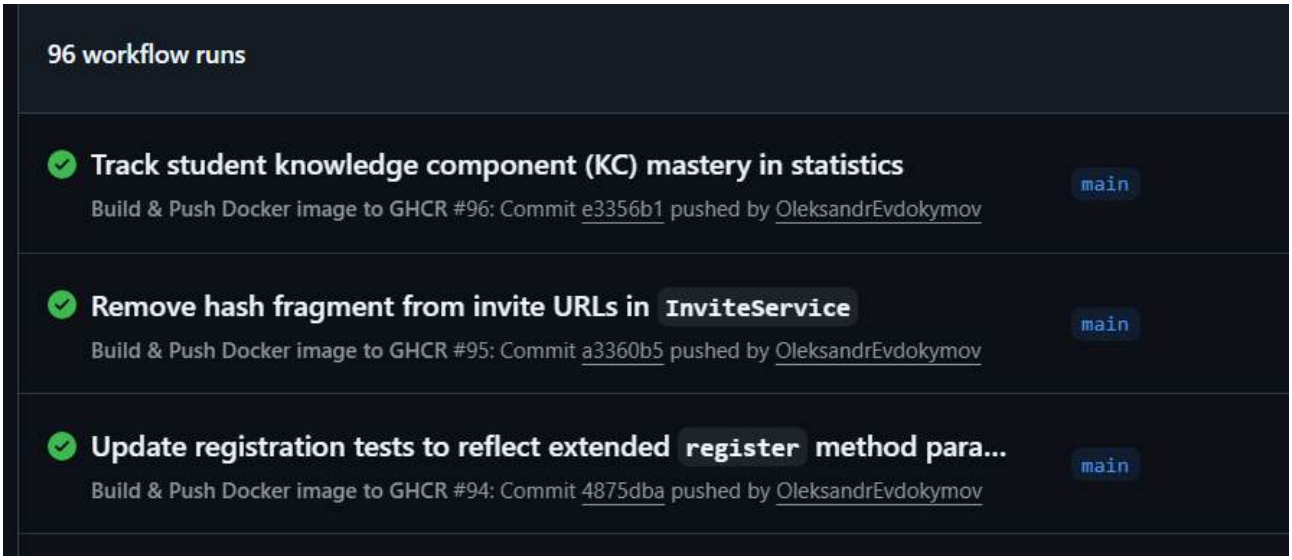


Рисунок 4.21 – Скріншот сторінки розгортання або GitHub Actions workflow для проекту «Мирна»

Загалом результати розроблення прототипу підтверджують можливість практичного використання запропонованої інформаційної технології як основи для побудови інтелектуальних навчальних систем нового покоління. Її особливість полягає у поєднанні інженерної реалізації, алгоритмічної генерації задач, діагностичної інтерпретації помилок, формально верифікованого навчального контенту та перспективного мотиваційного шару. Це забезпечує основу для подальшого розвитку системи як у навчальному, так і в науково-дослідному напрямі.

4.7 Експериментальна апробація та оцінювання працездатності прототипу

Для підтвердження працездатності реалізованих програмних механізмів було використано журнали роботи навчального прототипу з теми методу Крамера. У цьому підрозділі наведено саме отримані експериментальні дані: підсумкові рейтинги студентів, розподіл помилкових дій і покриття цих помилок діагностичними моделями. Персональні дані учасників не наводяться; у таблицях використано коди S1–S19.

Апробація не трактується як повний педагогічний експеримент із доведенням переваги над контрольною групою. Її призначення полягає у перевірці того, що прототип здатний забезпечувати параметричну генерацію задач, покрокову перевірку, фіксацію результатів, діагностику типових помилок і оновлення навчального стану в реальному сценарії роботи студента.

Таблиця 4.8 – Узагальнені дані експериментальної апробації прототипу

Показник	Значення
Предметна тема апробації	метод Крамера для систем лінійних алгебраїчних рівнянь
Кількість завершених протоколів	19
Початковий рейтинг	50 % для кожного учасника
Середній кінцевий рейтинг	97,89 %
Медіана / мода	99 % / 100 %
Мінімум / максимум	85 % / 100 %
Стандартне відхилення	3,40 %
Зафіксовані помилкові дії	56
Покриття відомими діагностичними моделями	87,5 % (49 із 56 помилкових дій)

За результатами 19 завершених протоколів усі учасники підвищили початковий рейтинг, який на старті становив 50 %. Середній кінцевий рейтинг дорівнював 97,89 %, медіана становила 99 %, а найчастіше значення дорівнювало 100 %. Це свідчить про те, що реалізований навчальний сценарій доводить користувача до завершення теми, а механізми повторних спроб, підказок і покрокової перевірки підтримують виправлення помилок у процесі виконання завдання.

Таблиця 4.9 – Кінцеві рейтинги учасників апробації

Учасник	Рейтинг, %	Учасник	Рейтинг, %
S1	97	S2	95
S3	100	S4	85
S5	99	S6	98
S7	100	S8	96
S9	100	S10	100
S11	100	S12	99
S13	99	S14	98
S15	97	S16	100
S17	100	S18	97
S19	100		

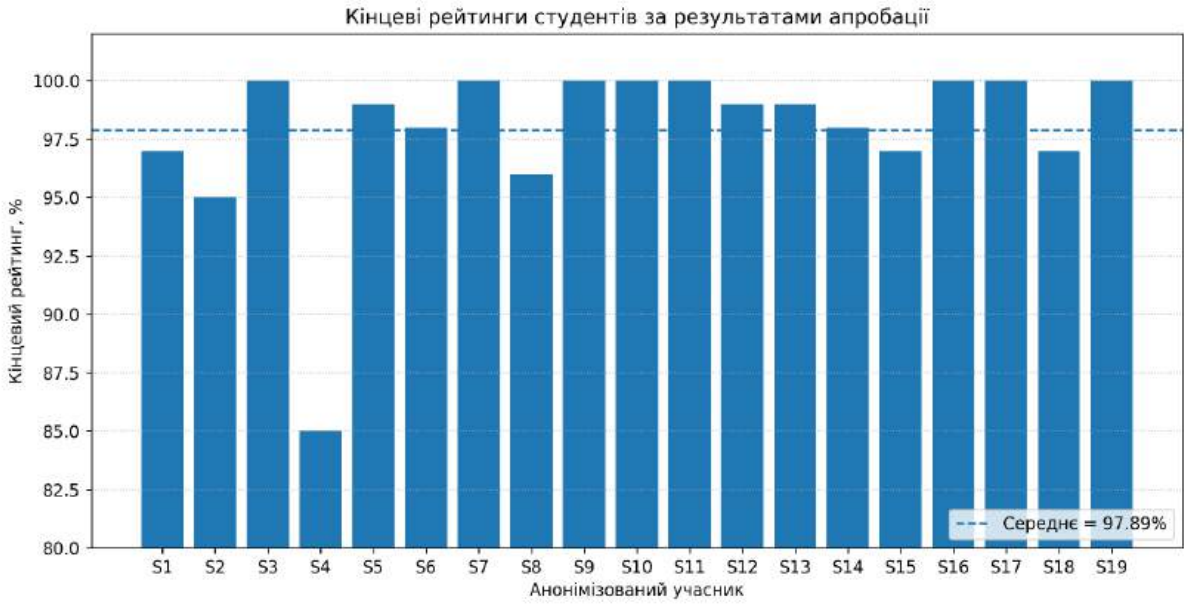


Рисунок 4.22 – Кінцеві рейтинги студентів за результатами апробації прототипу

Найнижче значення кінцевого рейтингу становило 85 %, найвище – 100 %. Значення 100 % отримали 8 із 19 учасників; ще 9 учасників завершили роботу в діапазоні від 96 % до 99 %. Така концентрація результатів у верхній частині шкали вказує не на автоматичне доведення переваги методики, а на

працездатність сценарію: система надає студенту можливість виправити помилку й завершити розв’язання з високим підсумковим результатом.

Окремо було проаналізовано 56 помилкових дій, зафіксованих у протоколах. Найбільшу частку становили помилки знаку та методичні помилки, пов’язані з правилом Саррюса. Саме ці класи помилок є типовими для методу Крамера, оскільки обчислення визначників вимагає коректного врахування знаків, порядку множників і структури допоміжних визначників.

Таблиця 4.10 – Розподіл помилкових дій за діагностичними класами

Клас помилки	Кількість випадків	Частка, %
Помилка знаку	22	39.3
Методична помилка (Саррюс)	14	25.0
Структурна помилка (плутанина)	8	14.3
Зворотне ділення	5	8.9
Невідомий клас помилки	7	12.5
Всього	56	100,0

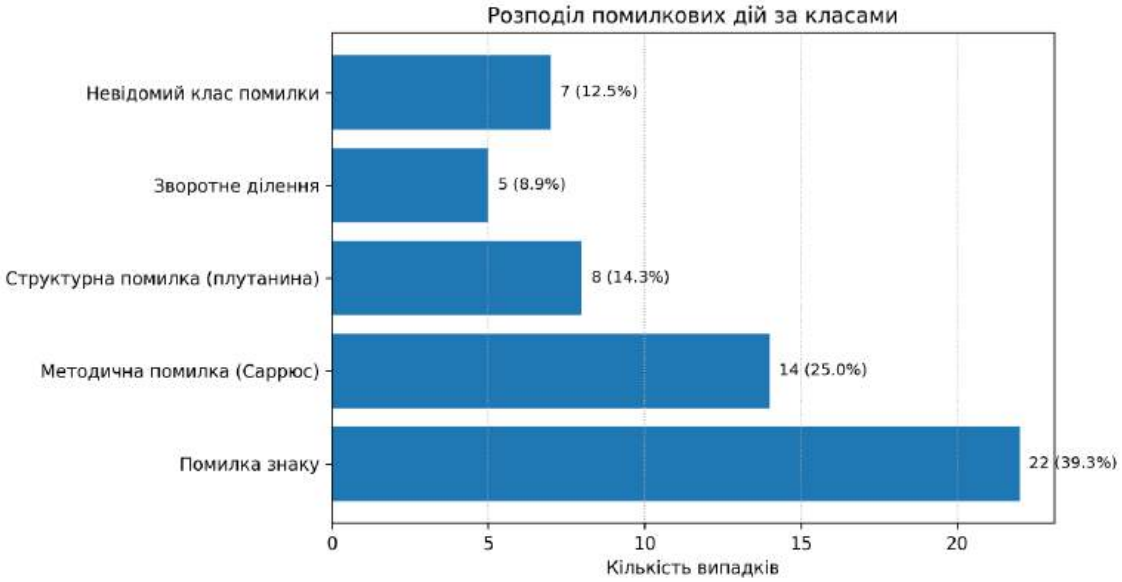


Рисунок 4.23 – Розподіл помилкових дій за класами у протоколах апробації

Відомі діагностичні моделі покрили 49 із 56 помилкових дій, тобто 87,5 %. До непокритих належали 7 випадків, віднесених до невідомого класу. Цей результат є важливим з інженерної точки зору: він показує, що діагностичне ядро не лише фіксує факт неправильної відповіді, а й у більшості випадків встановлює ймовірну причину помилки, яку можна використати для формування підказки або корекції подальшого навчального впливу.

Таким чином, експериментальна апробація підтвердила працездатність інформаційної технології на рівні прототипу: задачі генеруються, відповіді перевіряються, результати журналюються, типові помилки класифікуються, а отримані дані можуть бути використані для уточнення моделі студента. Обмеженням наведених результатів є відсутність контрольної групи, тому вони підтверджують працездатність і діагностичну придатність прототипу, але не використовуються як самостійне статистичне доведення підвищення якості навчання порівняно з традиційною методикою.

Висновки до розділу 4

1. Сформульовано вимоги до інформаційної технології підтримки навчання точним наукам, які охоплюють функціональні, діагностичні, педагогічні, архітектурні, експлуатаційні та безпекові аспекти.

2. Побудовано машинну модель системи, що відображає перехід від системних моделей до програмно реалізовних сутностей, алгоритмів і структур даних.

3. Описано архітектуру інтелектуальної навчальної програми «Мирна», що базується на Angular, Java 17, Spring/Spring Boot, PostgreSQL, GitHub, GitHub Actions і GitHub Pages.

4. Охарактеризовано основні програмні модулі прототипу: генератор задач, модуль перевірки відповідей, діагностичне ядро, модуль формально верифікованого навчального контенту та модуль мотиваційного регулювання.

5. Показано, що алгоритмічна генерація задач і автоматична перевірка відповідей утворюють базове операційне ядро системи, а діагностика, формальна верифікація та мотивація є інтелектуальними розширеннями цього ядра.

6. Описано прототип використання Lean/Mathlib для формування поетапного навчального HTML-матеріалу на прикладі методу Крамера.

7. Визначено сценарії оцінки працездатності прототипу та напрями подальшого розвитку, зокрема поглиблення діагностики, розширення банку задач, інтеграцію формальної верифікації та проведення педагогічного експерименту.

8. Проведено експериментальну апробацію працездатності прототипу на основі 19 завершених протоколів роботи студентів із навчальним модулем методу Крамера; середній кінцевий рейтинг становив 97,89 %, медіана – 99 %, а стандартне відхилення – 3,40 %.

9. Проаналізовано 56 помилкових дій: найбільшу частку становили помилки знаку (39,3 %) та методичні помилки, пов’язані з правилом Саррюса (25,0 %); відомі діагностичні моделі покрили 87,5 % помилок, що підтверджує діагностичну придатність прототипу.

10. Отримані результати апробації свідчать, що запропонована інформаційна технологія забезпечує не лише автоматизовану перевірку кінцевої відповіді, а й накопичення діагностично значущих даних для подальшого уточнення моделі студента, адаптивного добору задач і розширення бази діагностичних правил. При цьому результати апробації слід інтерпретувати як підтвердження працездатності та навчальної придатності прототипу, а не як повний доказ статистично значущого підвищення успішності порівняно з контрольною групою.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-прикладну задачу розроблення моделей, методів та інформаційної технології підтримки навчання точним наукам, що забезпечують системне поєднання діагностики причин помилок, адаптивного вибору навчальних задач, параметричної генерації коректних варіантів, формальної верифікації математичних доведень і алгоритмів, а також мотиваційного регулювання навчального процесу. Запропоновані рішення спрямовані на підвищення змістовної глибини комп'ютерної підтримки навчання точних дисциплін і подолання обмежень систем, орієнтованих лише на перевірку кінцевої відповіді.

Наступні висновки сформульовано за результатами вирішення поставлених завдань.

1. У результаті аналізу сучасного стану проблеми підтримки навчання точним наукам встановлено, що точні науки виконують системоутворювальну роль у підготовці фахівців STEM-сфери, а зниження рівня математичної підготовки, зафіксоване у міжнародних і національних освітніх вимірюваннях, створює потребу в нових інтелектуальних засобах підтримки навчання. Показано, що проблема має не лише кількісний характер, пов'язаний зі зниженням результатів, а й якісний характер, який проявляється у недостатньому розвитку здатності студентів до доведення, виведення, побудови алгоритмів і перенесення знань на нетипові задачі.

2. Систематизований огляд інтелектуальних навчальних систем, цифрових платформ, методів knowledge tracing, генеративного штучного інтелекту та систем формальної верифікації показав, що наявні рішення здебільшого підтримують окремі функції навчального процесу: автоматичну перевірку відповіді, генерацію задач, адаптивний вибір матеріалу або мовне пояснення. Водночас вони не забезпечують цілісного поєднання діагностики першопричин помилок, формальної коректності міркувань, параметричної генерації задач і мотиваційного регулювання в межах єдиної інформаційної технології.

3. Виконано системне моделювання процесу підтримки навчання точним наукам, у межах якого побудовано взаємопов’язаний комплекс моделей: вербальну, структурну, функціональну, математичну та машинну. Така ієрархія моделей дозволила послідовно перейти від змістовного опису предметної області та навчального процесу до формалізованих співвідношень, алгоритмічних процедур і програмної реалізації прототипу інформаційної технології.

4. Розроблено діагностичні моделі встановлення причин помилкових відповідей, які, на відміну від моделей простого оцінювання стану знань, розглядають помилку як спостережуваний наслідок прихованої причини. Запропоновано підхід до встановлення першопричини помилки за непрямими ознаками: структурою відповіді, способом запису, першим розходженням з еталоном, проміжними діями, поведінковими характеристиками та історією взаємодії. Це дозволяє локалізувати помилку на рівні конкретних компонентів знань і уникнути некоректного узагальнення про незнання всієї теми.

5. Запропоновано багаторівневу модель оновлення стану знань студента, яка поєднує баєсівське оновлення ймовірностей засвоєння компонентів знань, структурну діагностику помилок і поведінковий слід навчальної взаємодії. Такий підхід дозволяє оновлювати не всю модель знань загалом, а лише ті компоненти, що пов’язані з установленою діагностичною гіпотезою, і створює основу для адаптивного вибору наступних навчальних задач.

6. Розроблено метод інтеграції формальної верифікації математичних доведень у навчальний процес, що використовує системи доказів, зокрема Lean/Mathlib, як джерело формально коректних математичних артефактів. Запропонований метод дозволяє представляти доведення як послідовність дидактично інтерпретованих кроків, пов’язаних з означеннями, лемами, теоремами та контрольними питаннями, що забезпечує перехід від формальної строгості до навчального сценарію, придатного для студента.

7. Розроблено метод інтеграції формальної верифікації алгоритмів у навчання побудові алгоритмічних рішень. Метод ґрунтується на формалізації специфікацій, передумов, післяумов, інваріантів і доказових зобов’язань, що

дозволяє розглядати коректність алгоритму як навчальний об’єкт. Це особливо важливо для підготовки фахівців із відмовостійких, вбудованих і критичних до безпеки систем, де емпіричне тестування не може повністю замінити формальне обґрунтування коректності алгоритмічних рішень.

8. Розроблено метод адаптивного вибору навчальних задач, який враховує стан знань студента, діагностичні гіпотези, складність задачі, історію навчальної взаємодії, мотиваційний стан і необхідність підтримання різноманітної навчальної траєкторії. Запропонований підхід дозволяє розглядати вибір задачі не як випадкову або лінійну послідовність, а як керований навчальний вплив, спрямований на усунення виявлених прогалин і розвиток цільових компонентів знань.

9. Розроблено алгоритмічну модель параметричної генерації навчальних задач, яка включає шаблон задачі, домен параметрів, предикат коректності, генератор, еталонну відповідь і процедуру перевірки результату. Модель забезпечує створення варіативних, математично коректних екземплярів задач, дозволяє уникати вироджених або педагогічно непридатних варіантів і формує основу для автоматичної перевірки відповідей та подальшої діагностики помилок.

10. Дістала подальшого розвитку модель мотиваційного регулювання навчального процесу на основі системи цифрових винагород, інтегрованої зі станом знань студента. Запропоновано пов’язувати мотиваційні стимули не лише з фактом правильної відповіді, а й зі змістовними навчальними подіями: подоланням помилки, відновленням знань, послідовністю зусиль, творчим внеском, соціальною взаємодією та прогресом у складних задачах. Така модель створює передумови для побудови сенсо-економічних механізмів у навчальних системах, у тому числі з використанням смартконтрактів і цифрових токенів.

11. Теоретично обґрунтовано можливість використання квантових алгоритмів для прискорення окремих процедур у просторі параметричних навчальних задач, зокрема генерації кандидатів, оцінювання щільності допустимих параметрів, пошуку релевантних задач і ранжування діагностичних

гіпотез. Показано, що квантове підсилення має розглядатися не як заміна класичної інформаційної технології, а як перспективний допоміжний рівень, тоді як критичні перевірки коректності, логіка навчального сценарію та формальна верифікація залишаються у класичному довіреному ядрі.

12. Дістали подальший розвиток інформаційна технологія та прототип інтелектуальної навчальної програми «Мирна», що реалізує основні програмні компоненти підтримки навчання точним наукам: генерацію задач, перевірку відповідей, елементи діагностики знань, адаптивну логіку переходів, модулі формально верифікованого навчального контенту та мотиваційного регулювання. Архітектура прототипу побудована з використанням Angular, Java 17, Spring/Spring Boot, PostgreSQL, GitHub, GitHub Actions і GitHub Pages, що забезпечує можливість подальшого розвитку, супроводу та розгортання системи в освітньому середовищі.

13. Створено прототип використання формальної верифікації у навчанні на прикладі формування поетапного HTML-матеріалу за формально верифікованими артефактами Lean/Mathlib, зокрема для подання методу Крамера. Такий прототип підтверджує можливість перетворення формально доведених математичних тверджень у дидактично структурований навчальний матеріал, що може бути інтегрований у загальний адаптивний навчальний цикл.

14. Оцінка працездатності розробленого прототипу показала можливість практичної реалізації запропонованих моделей і методів у вигляді програмних модулів та навчальних сценаріїв. При цьому педагогічна ефективність запропонованої інформаційної технології потребує подальшого експериментального підтвердження на вибірках студентів, що визначено як один із напрямів подальших досліджень.

Практичне значення отриманих результатів полягає у можливості їх використання під час розроблення інтелектуальних навчальних систем, цифрових освітніх платформ, систем дистанційного та змішаного навчання, а також програмних засобів підтримки курсів математики, алгоритмів і суміжних точних дисциплін. Основні положення дисертаційної роботи доведено до рівня

алгоритмічних рішень, програмних модулів і прототипу інтелектуальної навчальної програми «Мирна». Практичне використання результатів підтверджується двома актами впровадження.

Таким чином, мету дисертаційної роботи досягнуто, поставлені завдання вирішено, а отримані результати є науково обґрунтованими, логічно завершеними та такими, що мають теоретичне і практичне значення для розвитку інформаційних технологій підтримки навчання точним наукам.

Подальші дослідження доцільно зосередити на:

- розширенні банку параметричних задач для різних розділів вищої математики, дискретної математики, алгоритмів і суміжних точних дисциплін;
- поглибленні діагностичних моделей шляхом включення багатокрокових розв’язань, дерев виразів, проміжних дій, часових і поведінкових ознак;
- проведенні педагогічного експерименту для кількісного оцінювання впливу інформаційної технології на результати навчання, мотивацію та стійкість засвоєння матеріалу;
- інтеграції формальної верифікації не лише для окремих теорем і прикладів, а й для ширших навчальних модулів з лінійної алгебри, математичного аналізу, дискретної математики та теорії алгоритмів;
- розвитку математичного інтерфейсу для роботи студентів із формальними системами доказів без необхідності безпосереднього опанування складного синтаксису Lean, Rocq/Coq або Isabelle/HOL;
- удосконаленні мотиваційної моделі, включно з експериментальною перевіркою сенсо-економічного підходу, правил нарахування цифрових винагород і механізмів прозорого обліку навчальних досягнень;
- дослідженні квантово-орієнтованих методів прискорення пошуку та оптимізації в просторі навчальних задач за умови збереження класичного довіреного ядра перевірки коректності;

- створенні масштабованої версії інформаційної технології з підтримкою багатокористувацького режиму, розширеної аналітики навчання, API для інтеграції з LMS та можливістю використання у закладах вищої освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кулік, А. С., Чухрай, А. Г., Гавриленко, О. В., Гайдачук, Д. О., Кулік, І. А. Інтелектуальна комп'ютерна підтримка навчання складанню алгоритмів та SQL-запитів: монографія. Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «ХАІ», 2020. 192 с.

2. Ланда, Л. Н. Алгоритмізація в навчанні. Київ: Просвітництво, 1966. 523 с.

3. Український центр оцінювання якості освіти. Загальна інформація про дослідження PISA // Український центр оцінювання якості освіти : [Електронний ресурс]. - Режим доступу: https://testportal.gov.ua/pisa/.

4. Український центр оцінювання якості освіти. НМТ-2024: офіційний звіт за результатами проведення // Український центр оцінювання якості освіти : [Електронний ресурс]. - 2024. - Режим доступу: https://testportal.gov.ua/nmt-2024-ofitsijnyj-zvit-za-rezultatamy-provedennya/.

5. Український центр оцінювання якості освіти. НМТ-2025: офіційний звіт за результатами проведення // Український центр оцінювання якості освіти : [Електронний ресурс]. - 2025. - Режим доступу: https://testportal.gov.ua/nmt-2025-ofitsijnyj-zvit-za-rezultatamy-provedennya/.

6. Український центр оцінювання якості освіти. Офіційний звіт про проведення НМТ у 2023 році. Том 1 / Український центр оцінювання якості освіти. - Київ : УЦОЯО, 2023. - Режим доступу: https://testportal.gov.ua/ofzvit/.

7. Чухрай, А. Г. Методологія навчання алгоритмам: монографія. Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «ХАІ», 2017. 336 с.

8. Aleven V. A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors / V. Aleven, B. M. McLaren, J. Sewall, K. R. Koedinger // International Journal of Artificial Intelligence in Education. - 2009. - Vol. 19. - P. 105-154.

9. Almarashdi H. S. Unveiling the potential: A systematic review of ChatGPT in transforming mathematics teaching and learning / H. S. Almarashdi // Eurasia

Journal of Mathematics, Science and Technology Education. - 2024. - Vol. 20, No. 12. - em2555. - doi: 10.29333/ejmste/15739.

10. Andris, M., Willems, K., Fijyé, B., De Wolf, P. Neurocognitive Mechanisms Underlying Math Avoidance in Individuals with Math Anxiety. 2023.

11. Angular Documentation // Angular.dev: [Електронний ресурс]. - Режим доступу: https://angular.dev/.

12. Angular Material Documentation // Angular Material: [Електронний ресурс]. - Режим доступу: https://material.angular.dev/.

13. Apt K. R. Verification of Sequential and Concurrent Programs / K. R. Apt, F. S. de Boer, E.-R. Olderog. - London : Springer, 2009. - 502 p.

14. Avigad, J., Massot, P. Mathematics in Lean. Lean Community Tutorial, 2020–2025. Available: https://leanprover-community.github.io/mathematics_in_lean/

15. Awaji, B., Solaiman, E. Design, Implementation, and Evaluation of Blockchain-Based Trusted Achievement Record System for Students in Higher Education. arXiv:2204.12547, 2022.

16. Bass L. Software Architecture in Practice / L. Bass, P. Clements, R. Kazman. - 4th ed. - Boston: Addison-Wesley, 2021.

17. Bertot Y. Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions / Y. Bertot, P. Castéran. - Berlin : Springer, 2004. - 469 p.

18. Brassard, G., Høyer, P., Mosca, M., Tapp, A. Quantum Amplitude Amplification and Estimation. Contemporary Mathematics, 2002, vol. 305, pp. 53–74. doi: 10.48550/arXiv.quant-ph/0005055.

19. Brusilovsky P. Adaptive Hypermedia / P. Brusilovsky // User Modeling and User-Adapted Interaction. - 2001. - Vol. 11. - P. 87-110.

20. Buzzard K. The Xena Project and formalising mathematics in Lean / K. Buzzard // The Xena Project: [Електронний ресурс]. - Режим доступу: https://xenaproject.wordpress.com/.

21. Carneiro M. The Type Theory of Lean / M. Carneiro. - Pittsburgh : Carnegie Mellon University, 2019. - 107 p.

22. Chi M. T. H. Eliciting self-explanations improves understanding / M. T. H. Chi, N. de Leeuw, M.-H. Chiu, C. LaVancher // Cognitive Science. - 1994. - Vol. 18, No. 3. - P. 439-477.

23. Chukhray, A., Dvinskykh, D., Narozhnyy, V., Stoliarenko, T. Using an Expression Tree for Adaptive Learning. In: 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT), 2023, pp. 1–5. doi: 10.1109/DESSERT61349.2023.10416497.

24. Chukhray A. Possibilities of using Intelligent Tutoring Systems (ITS) in Higher Mathematics Courses / A. Chukhray, T. Stoliarenko, O. Yevdokymov, V. Demyanenko // Open Information and Computer Integrated Technologies. - 2025. - No. 102. - P. 92-119.

25. Chukhray, A., Yevdokymov, O., Mandrikova, L., Dehtiarova, T., Havrylenko, O. Diagnostic Models and a Crypto-Based Sense-Economic Approach to Enhancing Motivation in Intelligent Mathematics Learning Systems. ICTERI-2025.

26. Coquand, T., Huet, G. The Calculus of Constructions. Information and Computation, 1988, vol. 76, no. 2–3, pp. 95–120. doi: 10.1016/0890-5401(88)90005-3.

27. Corbett, A. T., Anderson, J. R. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. User Modeling and User-Adapted Interaction, 1994, vol. 4, no. 4, pp. 253–278. doi: 10.1007/BF01099821.

28. de Moura, L., Kong, S., Avigad, J., van Doorn, F., von Raumer, J. The Lean Theorem Prover (System Description). In: Automated Deduction – CADE-25, 2015, vol. 9195, pp. 378–388. doi: 10.1007/978-3-319-21401-6_26.

29. de Moura, L., Ullrich, S. The Lean 4 Theorem Prover and Programming Language. In: Automated Deduction – CADE 28, 2021, vol. 12699, pp. 625–635. doi: 10.1007/978-3-030-79876-5_37.

30. Desmarais M. C. A review of recent advances in learner and skill modeling in intelligent learning environments / M. C. Desmarais, R. S. J. d. Baker // User Modeling and User-Adapted Interaction. - 2012. - Vol. 22. - P. 9-38.

31. Dijkstra E. W. A Discipline of Programming / E. W. Dijkstra. - Englewood Cliffs : Prentice-Hall, 1976. - 217 p.

32. Docker Documentation // Docker Docs: [Електронний ресурс]. - Режим доступу: https://docs.docker.com/.

33. du Plooy, E., Casteleijn, D., Franzsen, D. Personalized Adaptive Learning in Higher Education: A Scoping Review of Key Characteristics and Impact on Academic Performance and Engagement. Heliyon, 2024, vol. 10, e39630. doi: 10.1016/j.heliyon.2024.e39630.

34. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software / E. Evans. - Boston: Addison-Wesley, 2003. - 560 p.

35. Farhi, E., Goldstone, J., Gutmann, S. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028, 2014.

36. Fernandes F. A. Towards a Blockchain-based Software Engineering Education / F. A. Fernandes, C. M. L. Werner // arXiv:2304.04549. - 2023.

37. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: PhD dissertation / R. T. Fielding. - University of California, Irvine, 2000.

38. Filliâtre J.-C. Why3 - Where Programs Meet Provers / J.-C. Filliâtre, A. Paskevich // ESOP 2013. - 2013. - P. 125-128.

39. Floyd R. W. Assigning meanings to programs / R. W. Floyd // Proceedings of Symposia in Applied Mathematics. - 1967. - Vol. 19. - P. 19-32.

40. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. - Boston: Addison-Wesley, 2002. - 560 p.

41. Gage M. WeBWorK: Generating, delivering, and checking math homework via the Internet / M. Gage, A. Pizer // ICTCM Proceedings. - 1999.

42. Gamma E. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. - Boston: Addison-Wesley, 1994. - 395 p.

43. Giovannetti, V., Lloyd, S., Maccone, L. Quantum Random Access Memory. Physical Review Letters, 2008, vol. 100, 160501. doi: 10.1103/PhysRevLett.100.160501.

44. GitHub Actions Documentation // GitHub Docs: [Електронний ресурс]. - Режим доступу: https://docs.github.com/en/actions.

45. GitHub. GitHub Enterprise Cloud Documentation // GitHub Docs: [Електронний ресурс]. - Режим доступу: https://docs.github.com/en/enterprise-cloud@latest.

46. GitHub Pages Documentation // GitHub Docs: [Електронний ресурс]. - Режим доступу: https://docs.github.com/en/pages.

47. Gries D. The Science of Programming / D. Gries. - New York : Springer, 1981. - 366 p.

48. Gulati, R., Jethi, A., Ghosh, A., Shimpi, A., Naik, A. Implementation of an Educational Model Leveraging the Power of Incentive Reward Theory with Blockchain. ITM Web of Conferences, 2020, vol. 32, 03027. doi: 10.1051/itmconf/20203203027.

49. Hew, K. F., Huang, W., Lo, C. Gamification Enhances Student Intrinsic Motivation, Perceptions of Autonomy and Relatedness but Not Competence: A Meta-Analysis and Systematic Review. Educational Technology Research and Development, 2023, vol. 72, no. 1, pp. 1–26. doi: 10.1007/s11423-023-10337-7.

50. Hibernate ORM Documentation // Hibernate: [Електронний ресурс]. - Режим доступу: https://hibernate.org/orm/documentation/.

51. Hoare C. A. R. An axiomatic basis for computer programming / C. A. R. Hoare // Communications of the ACM. - 1969. - Vol. 12, No. 10. - P. 576-580.

52. Hunt T. An open-source system for computer-aided assessment in mathematics / T. Hunt // International Journal of Mathematical Education in Science and Technology. - 2012.

53. Iannone, P., Thoma, A. Interactive Theorem Provers for University Mathematics: an Exploratory Study of Students' Perceptions. International Journal of

Mathematical Education in Science and Technology, 2024. doi: 10.1080/0020739X.2023.2178981.

54. IEEE Computer Society. Guide to the Software Engineering Body of Knowledge (SWEBOK). - Version 4.0. - IEEE Computer Society, 2024.

55. ISO/IEC 25010:2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. - Geneva: ISO, 2011.

56. ISO/IEC/IEEE 42010:2022. Software, systems and enterprise - Architecture description. - Geneva: ISO, 2022.

57. Jest Documentation // Jest: [Електронний ресурс]. - Режим доступу: https://jestjs.io/docs/getting-started.

58. Jirgensons, M., Kapenieks, J. Blockchain and the Future of Digital Learning Credential Assessment and Management. Journal of Teacher Education for Sustainability, 2018, vol. 20, no. 1, pp. 145–156. doi: 10.2478/jtes-2018-0009.

59. JUnit 5 User Guide // JUnit: [Електронний ресурс]. - Режим доступу: https://junit.org/junit5/docs/current/user-guide/.

60. Khan Academy. Khanmigo: AI-powered personal tutor and teaching assistant // Khanmigo : [Електронний ресурс]. - Режим доступу: https://www.khanmigo.ai/.

61. Koedinger K. R. Intelligent tutoring goes to school in the big city / K. R. Koedinger, J. R. Anderson, W. H. Hadley, M. A. Mark // International Journal of Artificial Intelligence in Education. - 1997. - Vol. 8. - P. 30-43.

62. Kristensen, M., Larsen, D., Sejdelin, L., Svoboda, K. The Role of Mathematics in STEM Activities: Syntheses and a Framework from a Literature Review. International Journal of Education in Mathematics, Science and Technology, 2023, vol. 12, pp. 418–431.

63. Kulik A. Diagnostic models of intelligent tutor system for teaching skills to solve algebraic equations / A. Kulik, A. Chukhray, M. Chukhray // International Journal of Emerging Technologies in Learning. - 2007. - Vol. 2, No. 1.

64. Kulik, A., Zeleniak, O., Chukhray, A., Prokhorov, O., Yashyna, O., Havrylenko, O., Yevdokymov, O., Torzhkov, A., Zayarnyi, O. The Concept of

Intelligent Training System for Ukrainian School Final STEM Exam Preparation. System Research and Information Technologies, 2025, no. 2, pp. 91–101. doi: 10.20535/SRIT.2308-8893.2025.2.09.

65. Leino K. R. M. Dafny: An Automatic Program Verifier for Functional Correctness / K. R. M. Leino // Logic for Programming, Artificial Intelligence, and Reasoning. - 2010. - P. 348-370.

66. Leroy X. Formal verification of a realistic compiler / X. Leroy // Communications of the ACM. - 2009. - Vol. 52, No. 7. - P. 107-115.

67. Liu, J., Li, Y., Chen, X., et al. Formalizing Quantum Hoare Logic in Isabelle/HOL. Journal of Automated Reasoning, 2021, vol. 65, pp. 529–561. doi: 10.1007/s10817-021-09604-x.

68. Mao, Y., Lin, C., Chi, M. Deep Learning vs. Bayesian Knowledge Tracing: Student Models for Interventions. Journal of Educational Data Mining, 2018, vol. 10, no. 2, pp. 28–54.

69. Martin, B., Mitrovic, A. Automatic Problem Generation in Constraint-Based Tutors. In: Intelligent Tutoring Systems. Springer, 2002, pp. 388–398.

70. Martin-Löf, P. Intuitionistic Type Theory. Napoli: Bibliopolis, 1984.

71. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design / R. C. Martin. - Boston: Prentice Hall, 2017. - 432 p.

72. Massot, P. Teaching Mathematics Using Lean and Controlled Natural Language. Leibniz International Proceedings in Informatics (LIPIcs), 2024, vol. 309, art. 27. doi: 10.4230/LIPIcs.ITP.2024.27.

73. The mathlib Community. The Lean Mathematical Library. In: Proceedings of CPP 2020, 2020, pp. 367–381. doi: 10.1145/3372885.3373824.

74. Mayer R. E. Multimedia Learning / R. E. Mayer. - 2nd ed. - Cambridge : Cambridge University Press, 2009. - 304 p.

75. Monras, A., Beige, A., Wiesner, K. Hidden Quantum Markov Models and Non-Adaptive Read-Out of Many-Body States. New Journal of Physics, 2012, vol. 14, 013041. doi: 10.1088/1367-2630/14/1/013041.

76. Montanaro, A. Quantum Speedup of Monte Carlo Methods. Proceedings of the Royal Society A, 2015, vol. 471, 20150301. doi: 10.1098/rspa.2015.0301.

77. Murray, T. Authoring Intelligent Computer Tutoring Systems: An Analysis of the State of the Art. International Journal of Artificial Intelligence in Education, 1999, pp. 98–129.

78. National Center for Education Statistics. Mathematics Achievement of U.S. Students. Washington, D.C.: U.S. Department of Education, 2023.

79. Nipkow T. Isabelle/HOL: A Proof Assistant for Higher-Order Logic / T. Nipkow, L. C. Paulson, M. Wenzel. - Berlin : Springer, 2002. - 215 p.

80. O'Gorman, B., et al. Bayesian Network Structure Learning Using Quantum Annealing. European Physical Journal Special Topics, 2015, vol. 224, pp. 163–188. doi: 10.1140/epjst/e2015-02349-3.

81. OECD. Mathematics for Life and Work: A Comparative Perspective on Mathematics to Inform Upper Secondary School Reform in England. Paris: OECD Publishing, 2024.

82. OECD. PISA 2022 Results. State of Learning and Equity in Education. Vol. I. Paris: OECD Publishing, 2024.

83. OECD. PISA 2022 Results (Volume I and II): Country Notes. Ukrainian regions (18 of 27) / OECD. - Paris : OECD Publishing, 2023. - Режим доступу: https://www.oecd.org/en/publications/pisa-2022-results-volume-i-and-ii-country-notes_ed6fbcc5-en/ukrainian-regions-18-of-27_78043794-en.html.

84. OpenAPI Initiative. OpenAPI Specification // OpenAPI Initiative: [Електронний ресурс]. - Режим доступу: https://spec.openapis.org/oas/latest.html.

85. Oracle. Java Platform, Standard Edition 17 Documentation // Oracle Docs: [Електронний ресурс]. - Режим доступу: https://docs.oracle.com/en/java/javase/17/.

86. OWASP Foundation. Application Security Verification Standard // OWASP: [Електронний ресурс]. - Режим доступу: https://owasp.org/www-project-application-security-verification-standard/.

87. Özhan, S. Ç., Kocadere, S. A. The Effects of Flow, Emotional Engagement, and Motivation on Success in a Gamified Online Learning Environment. Journal of

Educational Computing Research, 2020, vol. 57, no. 8, pp. 2006–2031. doi: 10.1177/0735633118823159.

88. Paas F. Cognitive Load Theory: Instructional Implications of the Interaction between Information Structures and Cognitive Architecture / F. Paas, A. Renkl, J. Sweller // Instructional Science. - 2003. - Vol. 32. - P. 1-8.

89. Pedan S. Implementation of Computer Tutoring Program Adaptation Methods / S. Pedan, A. Chukhray, S. Hall, A. Kulik // Journal of Higher Education Theory and Practice. - 2012. - Vol. 12, No. 1. - P. 39-55.

90. Pedan S. Pedagogical development for computer tutoring: inner loop tasks / S. Pedan, A. Kulik, A. Chukhray, S. Hall // Journal of Business and Accounting. - 2011. - Vol. 4, No. 1. - P. 76-90.

91. Pelánek, R. Bayesian Knowledge Tracing, Logistic Models, and Beyond: An Overview of Learner Modeling Techniques. User Modeling and User-Adapted Interaction, 2017, vol. 27, no. 3–5, pp. 313–350. doi: 10.1007/s11257-017-9193-2.

92. Piech, C., Bassen, J., Huang, J., et al. Deep Knowledge Tracing. In: Advances in Neural Information Processing Systems 28, 2015, pp. 505–513. doi: 10.48550/arXiv.1506.05908.

93. Pierce B. C. Software Foundations / B. C. Pierce, C. Casinghino, M. Greenberg, V. Sjöberg, B. Yorgey. - [Electronic resource]. - Access mode: https://softwarefoundations.cis.upenn.edu/.

94. PostgreSQL Documentation // PostgreSQL Global Development Group: [Електронний ресурс]. - Режим доступу: https://www.postgresql.org/docs/.

95. PostgreSQL JDBC Driver Documentation // pgJDBC: [Електронний ресурс]. - Режим доступу: https://jdbc.postgresql.org/documentation/.

96. Rocq Prover. The Rocq Prover official website // Rocq Prover : [Електронний ресурс]. - Режим доступу: https://rocq-prover.org/.

97. Rojo Robas, V., Madariaga, J. M., Villarroel, J. D. Secondary Education Students' Beliefs about Mathematics and Their Repercussions on Motivation. Mathematics, 2020, vol. 8, no. 3, art. 368. doi: 10.3390/math8030368.

98. Sangwin C. J. Computer Aided Assessment of Mathematics / C. J. Sangwin. - Oxford : Oxford University Press, 2013. - 176 p.

99. Shailendra, S., Kadel, R., Sharma, A. Framework for Adoption of Generative Artificial Intelligence in Education. SITE, Melbourne Institute of Technology, 2024.

100. Shi, L., Cristea, A. I. Motivational Gamification Strategies Rooted in Self-Determination Theory. In: Intelligent Tutoring Systems. Springer, 2016, pp. 294–299. doi: 10.1007/978-3-319-39583-8_32.

101. Shih, S. C., Chang, C. C., Kuo, B. C., et al. Mathematics Intelligent Tutoring System for Learning Multiplication and Division of Fractions Based on Diagnostic Teaching. Education and Information Technologies, 2023, vol. 28, pp. 9189–9210.

102. Sommerville I. Software Engineering / I. Sommerville. - 10th ed. - Boston: Pearson, 2016. - 816 p.

103. Spring Boot Reference Documentation // Spring.io: [Електронний ресурс]. - Режим доступу: https://docs.spring.io/spring-boot/.

104. Spring Framework Documentation // Spring.io: [Електронний ресурс]. - Режим доступу: https://docs.spring.io/spring-framework/reference/.

105. Spring Security Reference // Spring.io: [Електронний ресурс]. - Режим доступу: https://docs.spring.io/spring-security/reference/.

106. Sweller J. Cognitive load during problem solving: Effects on learning / J. Sweller // Cognitive Science. - 1988. - Vol. 12, No. 2. - P. 257-285.

107. Tan K. H. Token Economy for Sustainable Education in the Future: A Scoping Review / K. H. Tan, M. Kasiveloo, I. H. Abdullah // Sustainability. - 2022. - Vol. 14, No. 2. - Article 716.

108. Thoma, A., Iannone, P. Learning about Proof with the Theorem Prover LEAN: the Abundant Numbers Task. International Journal of Research in Undergraduate Mathematics Education, 2022, vol. 8, pp. 64–93. doi: 10.1007/s40753-021-00140-1.

109. Trinh T. H. Solving olympiad geometry without human demonstrations / T. H. Trinh, Y. Wu, Q. V. Le, H. He, T. Luong // Nature. - 2024. - Vol. 625. - P. 476-482. - doi: 10.1038/s41586-023-06747-5.

110. UNESCO. Guidance for generative AI in education and research / UNESCO. - Paris : UNESCO, 2023. - 44 p. - Режим доступу: https://unesdoc.unesco.org/ark:/48223/pf0000386693.

111. Van Vaerenbergh, S., Pérez-Suay, A. A Classification of Artificial Intelligence Systems for Mathematics Education. In: Mathematics Education in the Age of Artificial Intelligence. Springer, Cham, 2022.

112. VanLehn K. Intelligent tutoring systems for continuous embedded assessment / K. VanLehn // The Future of Assessment: Shaping Teaching and Learning. - 2008. - P. 113-138.

113. VanLehn, K. The Behavior of Tutoring Systems. International Journal of Artificial Intelligence in Education, 2006, vol. 16, no. 3, pp. 227–265.

114. VanLehn K. The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems / K. VanLehn // Educational Psychologist. - 2011. - Vol. 46, No. 4. - P. 197-221.

115. W3C. Web Content Accessibility Guidelines (WCAG) 2.2 // W3C Recommendation: [Електронний ресурс]. - Режим доступу: https://www.w3.org/TR/WCAG22/.

116. Wadler, P. Propositions as Types. Communications of the ACM, 2015, vol. 58, no. 12, pp. 75–84. doi: 10.1145/2699407.

117. Waterproof. Educational software for proving mathematical statements // Waterproof : [Електронний ресурс]. - Режим доступу: https://impermeable.github.io/.

118. The Waterproof plugin for the Coq proof assistant // GitHub : [Електронний ресурс]. - Режим доступу: https://github.com/impermeable/coq-waterproof.

119. Wemmenhove, J., et al. Waterproof: Educational Software for Learning How to Write Mathematical Proofs. arXiv:2211.13513, 2022. doi: 10.48550/arXiv.2211.13513.

120. Yan, X. K., Hanna, G. Using the Lean Interactive Theorem Prover in Undergraduate Mathematics. International Journal of Mathematical Education in Science and Technology, 2023. doi: 10.1080/0020739X.2023.2227191.

121. Yevdokymov, O., Chukhray, A., Stoliarenko, T. Operationalizing the Formalizability of Mathematics Problems for Intelligent Tutoring Systems: Taxonomy, Measurement Protocol, and Educational Impact. CEUR Workshop Proceedings, 2026, vol. 4164, paper 25.

122. Zhou, H., Zhang, X., Chen, L. A Survey of Blockchain-Based Educational Systems: Principles, Applications, and Challenges. IEEE Access, 2021, vol. 9, pp. 112354–112372. doi: 10.1109/ACCESS.2021.3103271.

ДОДАТОК А. СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Kulik, A.; Trofymova, I.; Chukhray, A.; Stoliarenko, T.; Yevdokymov, O. Relevant Objectives of Developing SQL Adaptive Learning Technology. Proceedings of the 2022 IEEE 12th International Conference on Dependable Systems, Services and Technologies, DESSERT 2022. Conference paper. DOI: 10.1109/DESSERT58054.2022.10018757. EID: 2-s2.0-85147846036. Part of ISBN: 9798350333046. Greece. (Scopus). (*Особистий внесок: участь у формулюванні актуальних задач розроблення адаптивної технології навчання SQL; аналіз вимог до інтелектуальної підтримки навчання SQL; підготовка матеріалів, пов'язаних із навчальними сценаріями та апробацією підходу*)

2. Євдокимов О. О. Актуальні задачі розробки адаптивної технології навчання SQL. XVI Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті». Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Україна. *(Особистий внесок: постановка проблеми, систематизація актуальних задач розроблення адаптивної технології навчання SQL, підготовка тез і формулювання висновків)*

3. Chukhray, A.; Stoliarenko, T.; Yevdokymov, O. Development of a web-based SQL intelligent tutoring system SQLTOR. Poster. Digital Theme UK-Ukraine Research Twinning Conference, 27 March - 30 March 2023. *(Особистий внесок: участь у формуванні концепції веборієнтованої інтелектуальної навчальної системи SQLTOR, аналіз функціональних вимог і підготовка матеріалів постерної презентації)*

4. Чухрай А. Г., Євдокимов О. О. та ін. Розроблення і впровадження інтелектуальної комп'ютерної системи підготовки школярів України до НМТ з точних наук: звіт з НДР. Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харківський авіаційний інститут», 2024. *(Особистий внесок: участь у розробленні концепції інтелектуальної комп'ютерної системи підготовки до*

НМТ з точних наук, аналіз предметної області, формування вимог до адаптивної підтримки навчання та підготовка відповідних розділів звіту)

5. Чухрай А. Г., Столяренко Т. Л., Євдокимов О. О., Дем'яненко В. А. Можливості використання інтелектуальних навчальних систем (ITS) в ЗВО для курсів вищої математики. Відкриті інформаційні та комп'ютерні інтегровані технології. № 102. Національний аерокосмічний університет «Харківський авіаційний інститут», 10.02.2025. DOI: 10.32620/oikit.2024.102.07. *(Особистий внесок: аналіз можливостей застосування ITS у курсах вищої математики; участь у класифікації типів математичних задач, визначенні обмежень наявних цифрових платформ і підготовці тексту статті)*

6. Кулік А.; Зеленяк О.; Чухрай А.; Прохоров О.; Яшина О.; Гавриленко О.; Євдокимов О.; Торжков А.; Заярний О. The concept of intelligent training system for Ukrainian school final STEM exam preparation. System Research and Information Technologies. Educational and Scientific Complex «Institute for Applied System Analysis» of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», 28.06.2025. DOI: 10.20535/SRIT.2308-8893.2025.2.09. (Scopus). *(Особистий внесок: участь у формуванні концепції інтелектуальної системи підготовки до фінального STEM-іспиту; аналіз вимог до діагностики знань, адаптивного добору задач і архітектури навчальної системи; підготовка фрагментів рукопису)*

7. Євдокимов О. О., Чухрай А. Г., Столяренко Т. Л. Діагностичні моделі та сенсо-економічний підхід на основі блокчейну в інтелектуальних навчальних системах. Матеріали І Науково-практичної конференції «Сучасні технологічні рішення для освіти, науки, бізнесу та міського господарства»: Секція 1. Рішення із використанням штучного інтелекту в галузі освіти та науки. Київ - Харків : ТОВ «Центр Дуальної освіти», 25 листопада 2025. 54 с. *(Особистий внесок: розроблення й опис ідеї поєднання діагностичних моделей із сенсо-економічним мотиваційним механізмом; формулювання блокчейн-орієнтованого підходу до цифрових винагород; підготовка тез)*

8. Євдокимов О. О., Мандрікова Л. В. Інтелектуальні навчальні програми та інтегровані асистенти доведень у навчанні квантових обчислень і програмування. Інформаційні технології і автоматизація - 2025: матеріали XVIII міжнародної науково-практичної конференції. Одеса, 30-31 жовтня 2025 р. Одеса : Видавництво ОНТУ, 2025. 1316 с. *(Особистий внесок: формулювання підходу до інтеграції інтелектуальних навчальних програм і асистентів доведень; аналіз можливостей застосування формальної верифікації у навчанні квантових обчислень і програмування; підготовка тексту тез)*

9. Chukhray, A.; Yevdokymov, O.; Mandrikova, L.; Dehtiarova, T.; Havrylenko, O. Diagnostic Models and a Crypto-Based Sense-Economic Approach to Enhancing Motivation in Intelligent Mathematics Learning Systems. 2026. DOI: 10.1007/978-3-032-10477-9_11. (Scopus). *(Особистий внесок: участь у розробленні діагностичних моделей для інтелектуальних систем навчання математики, формуванні crypto-based sense-economic мотиваційного підходу, аналізі результатів та підготовці рукопису)*

10. Yevdokymov, O.; Chukhray, A.; Stoliarenko, T. Operationalizing the Formalizability of Mathematics Problems for Intelligent Tutoring Systems: Taxonomy, Measurement Protocol, and Educational Impact. Proceedings of the 5th International Workshop of IT-professionals on Artificial Intelligence, ProfIT AI 2025. Liverpool, United Kingdom, October 15-17, 2025. *(Особистий внесок: розроблення таксономії формалізованості математичних задач, формулювання протоколу вимірювання, обґрунтування освітнього впливу та підготовка основного тексту публікації)*

11. Yevdokymov, O.; Luchsheva, O. Development of a prototype intelligent web-based system for learning mathematics through an adaptive scenario. Aerospace Technic and Technology. No. 2, 2026. DOI: 10.32620/aktt.2026.2.06. *(Особистий внесок: розроблення архітектури та прототипу веборієнтованої інтелектуальної системи навчання математики, реалізація адаптивного сценарію, аналіз працездатності прототипу та підготовка рукопису.)*

12. Yevdokymov, O.; Luchsheva, O. A Mathematical Model of Automatic Verification of Formalized Proofs and a Conservative Presentation Interface over Lean. Bulletin of V. N. Karazin Kharkiv National University, series «Mathematical modeling. Information technology. Automated control systems». 2026. Vol. 69. P. 33-40. DOI: 10.26565/2304-6201-2026-69-03. *(Особистий внесок: розроблення математичної моделі автоматичної перевірки формалізованих доведень, формування підходу до консервативного інтерфейсу подання над Lean, аналіз прикладів застосування та підготовка тексту статті.)*

ДОДАТОК Б. ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ

1. 12th International Conference on Dependable Systems, Services and Technologies (DESSERT 2022, Scopus) (м. Афіни, Греція, 2022) – дистанційна участь із доповіддю

2. XVI Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (м. Харків, Україна, 2022) – дистанційна участь

3. International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (ICTERI 2025, Scopus) (Франція, 2025) – дистанційна участь із доповіддю

4. І Науково-практична конференція «Сучасні технологічні рішення для освіти, науки, бізнесу та міського господарства» (м. Київ – Харків, Україна, 2025) – дистанційна участь

5. XVIII Міжнародна науково-практична конференція «Інформаційні технології і автоматизація – 2025» (м. Одеса, Україна, 2025) – дистанційна участь

6. 5th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2025, Scopus) (м. Ліверпуль, Велика Британія, 2025) – дистанційна участь із доповіддю

7. 5th International Conference on Machine Learning & Big Data Analytics (ICMLBDA 2025, Scopus) (м. Марампалли, штат Керала, Індія, 2025) – дистанційна участь із доповіддю, нагорода Best Paper Award у напрямах “Next Gen Analytics” та “Data-Driven Models and Analytics for Circular Economy and Sustainable Innovation”

ДОДАТОК В. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

«ЗАТВЕРДЖУЮ»
Проректор з НПР
Національного аерокосмічного університету
«Харківський авіаційний інститут»
к.т.н., доцент
_____ Андрій ГУМЕННИЙ
«___» _____ 2026 р.

АКТ
впровадження результатів дисертаційної роботи аспіранта Олександра Євдокимова в освітній процес кафедри інженерії програмного забезпечення Національного аерокосмічного університету «Харківський авіаційний інститут»

Комісія у складі: голова комісії – завідувач кафедри інженерії програмного забезпечення (№603), д.т.н., професор Ігор ТУРКІН, членів комісії – к.т.н., доцент, доцент кафедри 603 Ілона ШЕВЧЕНКО, к.т.н., доцент, доцент кафедри 603 Євгенія СОКОЛОВА засвідчує, що інформаційна технологія (ІТ) підтримки навчання формалізованим наукам (ПНФН), розроблена в рамках дисертаційного дослідження аспіранта Олександра Євдокимова, впроваджена в освітній процес кафедри інженерії програмного забезпечення при викладанні навчальної дисципліни «Основи інженерії систем» у 631пст і 632пст групах.

Було розроблено і впроваджено цикл практичних робіт, які стосуються побудови студентами моделей згаданої вище ІТ ПНФН, а саме: *перша* практична робота присвячена побудові вербальної моделі ІТ ПНФН і варіантів використання (use cases); у *другій* роботі розглядається питання декомпозиції на підсистеми, побудови об'єктно-орієнтованих моделей підсистем з використанням патернів проєктування; *третя* робота присвячена побудові моделі даних ІТ ПНФН; *четверта* робота – передбачає опис математичних моделей і методів, які використовуються у ІТ ПНФН для генерації завдань, оцінювання результатів та діагностики рівня підготовки; *п'ята* робота спрямована на обгрунтований вибір архітектурних і програмних засобів реалізації системи.

Комісія підтверджує, що наведений цикл практичних робіт відповідає сучасному стану питання розробки складних програмних систем і дає змогу студентам набути знання, вміння та навички проєктування актуальних ІТ в галузі програмної інженерії.

Голова комісії:
д.т.н., професор, завідувач кафедри 603 Ігор ТУРКІН

Члени комісії:
к.т.н., доцент, доцент кафедри 603 Ілона ШЕВЧЕНКО

к.т.н., доцент, доцент кафедри 603 Євгенія СОКОЛОВА

Приватний заклад (організація, установа) Приватна Гімназія "Еммануїл"

м. Мукачево, вул. Набережна Незалежності, 3, тел.: 0633179798, e-mail: nvk.emmanuil@gmail.com, код ЄДРПОУ 39848526

від «02» 04 2026 р. № 08/04-12

ЗАТВЕРДЖУЮ
Директор Приватного закладу (організації, установи) Приватної Гімназії "Еммануїл"
_____ Коропець Інна Ігорівна
«02» 04 2026 р.

АКТ

про впровадження результатів дисертаційної роботи Євдокимова Олександра Олеговича на тему «МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПІДТРИМКИ НАВЧАННЯ ТОЧНИМ НАУКАМ», представленої на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки у навчальний процес Приватного закладу (організації, установи) Приватної Гімназії "Еммануїл"

м. Мукачево «02» 04 2026 р.

Комісія у складі: голова комісії — КОРОПЕЦЬ Інна Ігорівна, директор; члени комісії — ТРІФАНОВА Діана Андріївна, вчитель початкових класів; ДАНКУЛИНЕЦЬ Оксана Віталіївна, вчитель початкових класів склала цей акт про те, що у період з «16» 03 2026 р. по «20» 03 2026 р. у Приватному закладі (організації, установі) Приватній Гімназії "Еммануїл" впроваджено результати дисертаційної роботи Євдокимова Олександра Олеговича, представленої на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки. Дисертаційна робота виконана Євдокимовим О.О. на кафедрі математичного моделювання та штучного інтелекту Національного Аерокосмічного Університету «Харківський Авіаційний Інститут».

Форма впровадження: адаптивний веб-застосунок ІКНП «Мирна» (інтелектуальна комп'ютерна навчальна програма), призначений для підтримки навчання точним наукам, організації розв'язування навчальних задач, автоматизованої генерації вправ, автоматизованої фіксації результатів виконання та накопичення навчальної аналітики.

Під час впровадження виконано такі роботи:

1. Надано доступ учителю ТРІФАНОВІЙ Діані Андріївні та учням 1 класу до веб-застосунку ІКНП «Мирна» для використання у навчальному процесі з математики.
2. Організовано виконання учнями навчальних задач у середовищі системи та проведено експериментальне відстеження часу, витраченого кожним учнем на розв'язання кожної задачі.
3. Проінструктовано вчителя ТРІФАНОВУ Діану Андріївну щодо можливостей системи, порядку організації роботи учнів, перегляду результатів і використання даних про час виконання задач.
4. Опрацьовано можливості застосування отриманих даних для індивідуалізації навчальної підтримки, добору задач і підготовки методичних рекомендацій.

До впроваджених результатів належать: моделі подання навчальних задач і параметрів навчальної взаємодії; методи збирання та аналізу даних про виконання задач; інформаційна технологія підтримки навчання точним наукам; програмні засоби веб-застосунку ІКНП «Мирна».

Педагогічний ефект впровадження полягає у підвищенні об'єктивності спостереження за ходом виконання навчальних задач, накопичені даних для аналізу індивідуального темпу роботи учнів, оперативному виявленні задач, що викликають труднощі, та розширенні можливостей учителя щодо індивідуалізації навчальної підтримки.

Основні результати впровадження розглянуто та схвалено на методичному засіданні Приватного закладу (організації, установи) Приватної Гімназії "Еммануїл" (протокол № 7 від «23» 03 2026 р.).

Висновок комісії: наукові та практичні результати дисертаційної роботи Євдокимова О.О. на тему «Моделі, методи та інформаційна технологія підтримки навчання точним наукам» впроваджено в навчальний процес Приватного закладу (організації, установи) Приватної Гімназії "Еммануїл" у вигляді веб-застосунку ІКНП «Мирна» та рекомендовано до використання під час навчання точним наукам і аналізу навчальної діяльності учнів.

Голова комісії _____ КОРОПЕЦЬ І. І.
Члени комісії _____ ТРІФАНОВА Д. А.
_____ ДАНКУЛИНЕЦЬ О. В.
Директор закладу _____ КОРОПЕЦЬ І. І.

Аерокосмічний ліцей на базі Національного аерокосмічного університету "ХАІ"
м. Харків, вул. Вадима Манька, 17, тел.: (057)7884037, e-mail: liceykhai@ukr.net, код ЄДРПОУ 30193232

від «20» травня 2026р. № 16

ЗАТВЕРДЖУЮ
Директор Аерокосмічного ліцею на базі
Національного аерокосмічного університету "ХАІ"
Волкова Неоніла Дмитрівна
«20» травня 2026 р.

АКТ

про впровадження результатів дисертаційної роботи
Євдокимова Олександра Олеговича
на тему «МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПІДТРИМКИ НАВЧАННЯ ТОЧНИМ НАУКАМ», представленої на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки

у навчальний процес Аерокосмічного ліцею на базі Національного аерокосмічного університету

м. Харків «20» травня 2026 р.

Комісія у складі: голова комісії — ВОЛКОВА Неоніла Дмитрівна, директор; члени комісії — ЛЮБОЖЕНКО Алла Георгіївна, вчитель математики; ЛИСАК Олександр Павлович, вчитель хімії та біології склала цей акт про те, що у період з «11» травня 2026 р. по «15» травня 2026 р. у Аерокосмічному ліцеї на базі Національного аерокосмічного університету "ХАІ" впроваджено результати дисертаційної роботи Євдокимова Олександра Олеговича, представленої на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки. Дисертаційна робота виконана Євдокимовим О.О. на кафедрі математичного моделювання та штучного інтелекту Національного Аерокосмічного Університету «Харківський Авіаційний Інститут».

Форма впровадження: адаптивний веб-застосунок ІКНП «Мирна» (інтелектуальна комп'ютерна навчальна програма), призначений для підтримки навчання точним наукам, організації розв'язування навчальних задач, автоматизованої генерації вправ, автоматизованої фіксації результатів виконання та накопичення навчальної аналітики.

Під час впровадження виконано такі роботи:

1. Надано доступ учителю ЛЮБОЖЕНКО Аллі Георгіївні та учням 11-А класу до веб-застосунку ІКНП «Мирна» для використання у навчальному процесі з математики.
2. Організовано виконання учнями навчальних задач у середовищі системи та проведено експериментальне відстеження часу, витраченого кожним учнем на розв'язання кожної задачі.
3. Проінструктовано вчителя ЛЮБОЖЕНКО Аллу Георгіївну щодо можливостей системи, порядку організації роботи учнів, перегляду результатів і використання даних про час виконання задач.
4. Опрацьовано можливості застосування отриманих даних для індивідуалізації навчальної підтримки, добору задач і підготовки методичних рекомендацій.

До впроваджених результатів належать: моделі подання навчальних задач і параметрів навчальної взаємодії; методи збирання та аналізу даних про виконання задач; інформаційна технологія підтримки навчання точним наукам; програмні засоби веб-застосунку ІКНП «Мирна».

Педагогічний ефект впровадження полягає у підвищенні об'єктивності спостереження за ходом виконання навчальних задач, накопиченні даних для аналізу індивідуального темпу роботи учнів, оперативному виявленні задач, що викликають труднощі, та розширенні можливостей учителя щодо індивідуалізації навчальної підтримки.

Основні результати впровадження розглянуто та схвалено на методичному засіданні Аерокосмічного ліцею на базі Національного аерокосмічного університету "ХАІ" (протокол № 4 від «19» травня 2026 р.).

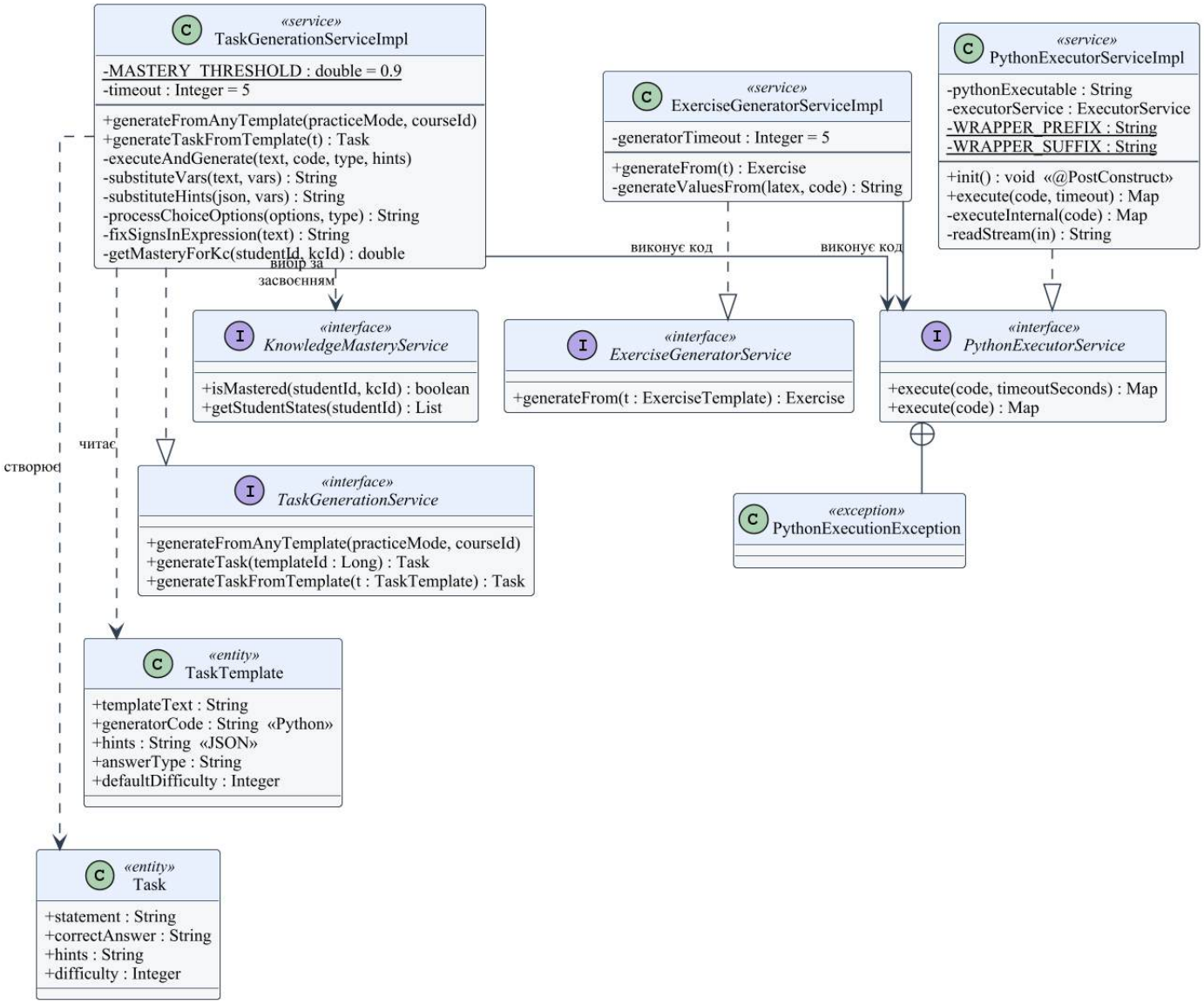
Висновок комісії: наукові та практичні результати дисертаційної роботи Євдокимова О.О. на тему «Моделі, методи та інформаційна технологія підтримки навчання точним наукам» впроваджено в навчальний процес Аерокосмічного ліцею на базі Національного аерокосмічного університету "ХАІ" у вигляді веб-застосунку ІКНП «Мирна» та рекомендовано до використання під час навчання точним наукам і аналізу навчальної діяльності учнів.

Голова комісії ВОЛКОВА Н. Д.
Члени комісії ЛЮБОЖЕНКО А. Г.
ЛИСАК О.П.

Директор закладу ВОЛКОВА Н. Д.

ДОДАТОК Г. ДІАГРАМИ КЛЮЧОВИХ ПРОГРАМНИХ КЛАСІВ, ЩО МІСТЯТЬ НАУКОВУ НОВИЗНУ

Діаграма класів підсистеми
динамічної генерації задач прототипу «Мирна»



Адаптивний вибір (TaskGenerationServiceImpl)
1. Відсів КЗ із P(засвоєння) ≥ 0.9.
2. Сортування шаблонів за зростанням P.
3. Випадково — один із 3 найменш засвоєних.
4. Підстановка #var в умову, відповідь, підказки.

Ізольоване виконання (PythonExecutorServiceImpl)
Окремий процес CPython (ProcessBuilder) з обмеженням часу; результат — словник *result* у форматі JSON.

Діаграма класів бассівської мережі моделювання знань студента (BKT) прототипу «Мирна»