

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

Кваліфікаційна наукова
праця на правах рукопису

ХИЖНЯК АНДРІЙ ВАСИЛЬОВИЧ

УДК 004.9:004.94:378.018.43(043.5)

ДИСЕРТАЦІЯ

МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРСОНАЛІЗОВАНОГО
НАВЧАННЯ З ІНЖЕНЕРНИХ СПЕЦІАЛЬНОСТЕЙ

122 – Комп'ютерні науки

12 – Інформаційні технології

галузь знань

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів,
текстів інших авторів супроводжується посиланнями на відповідне джерело.

_____ Хижняк Андрій Васильович
підпис

Науковий керівник

Казимир Володимир Вікторович
доктор технічних наук, професор

Чернігів – 2026

АНОТАЦІЯ

Хижняк Андрій Васильович. Моделі, методи та інформаційна технологія персоналізованого навчання з інженерних спеціальностей. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 – «Комп'ютерні науки» (12 – «Інформаційні технології»). – НУ «Чернігівська політехніка», МОН України, Чернігів, 2026.

Дисертаційне дослідження є завершеною науковою працею, в результаті якої виконано конкретне **наукове завдання** щодо комплексного забезпечення процесів автоматизованої генерації персоналізованих практичних інженерних завдань, автоматичного розгортання необхідних віртуальних навчальних середовищ та автоматичної перевірки результатів виконання цих завдань для підвищення рівня персоналізованого навчання та набуття студентами сталих практичних навичок.

Об'єктом дослідження є інформаційне забезпечення процесу персоналізованого навчання студентів інженерних спеціальностей. **Предметом** дослідження є моделі, методи та елементи інформаційної технології персоналізації практичного навчання студентів інженерних спеціальностей.

Метою дисертаційної роботи є підвищення рівня персоналізованого навчання студентів та набуття ними сталих практичних навичок за рахунок масштабованої автоматизації процесів створення, виконання та оцінювання результатів виконання персоналізованих практичних інженерних завдань з дотриманням норм академічної доброчесності.

У **вступі** обґрунтовано актуальність теми дослідження, сформульовані мета, задачі, методи дослідження та відображено зв'язок дослідження з науковими програмами кафедри, наведено наукову новизну і практичне значення результатів дисертаційної роботи.

У **першому розділі** виконаний аналіз персоналізованих практичних завдань в навчанні студентів інженерних спеціальностей, проведено аналіз методів

персоналізованого навчання, виявлено та систематизовано їх основні недоліки, визначено перспективні методи персоналізації, досліджено можливості AI для цілей персоналізації та проаналізовано підходи до формалізації завдань в інженерній освіті, обґрунтовано необхідність створення функціональної та формальних моделей, придатних для автоматизованої генерації, параметризації, автоматичного розгортання та перевірки. Сформульовано вимоги до віртуальних навчальних середовищ. Наведено актуальність дослідження, сформульовані наукові прогалини і поставлені задачі дослідження.

У другому розділі сформовано формальний базис дослідження, в межах якого реалізовано підхід централізованої формалізації ключових елементів персоналізованого навчання. Зокрема, вперше розроблено функціональну модель персоналізованого практичного інженерного завдання, яка формалізує послідовність його етапів (життєвий цикл) і забезпечує наскрізний зв'язок між ними, та формальну модель, яка визначає його структуру, основні компоненти та взаємозв'язки між ними. Вперше розроблено домен-специфічну мову опису навчальних завдань Learning Task Definition Language (LTDL), яка виступає уніфікованим засобом формального опису практичних завдань, їх параметрів, умов виконання, середовища, критеріїв оцінювання. Мова забезпечує однозначність, машиночитаність і формальну інтерпретованість завдань, що є необхідною умовою для автоматизованої генерації, розгортання та перевірки результатів виконання, а також подальшого масштабування. Формалізація створює підґрунтя для автоматизованої обробки практичних завдань в межах різних інженерних спеціальностей.

У третьому розділі вперше запропоновано архітектуру інтелектуального асистента і формалізовано процеси підтримки навчання, генерації контенту та адаптації до індивідуальних особливостей студента. Архітектура інтелектуального асистента базується на мультиагентній системі оркестрації AI-агентів і інтегрується з розробленою домен-специфічною мовою опису практичного завдання LTDL. Удосконалено 4 взаємопов'язані методи автоматизації персоналізованих практичних інженерних завдань: а саме: а) метод автоматизованої генерації на базі GenAI та

розробленої мови LTDL, б) метод автоматичного розгортання за рахунок використання мови LTDL та персонального AI-асистента, в) метод параметризації практичних завдань за рахунок поєднання стохастичної генерації параметрів із формальними обмеженнями мови LTDL, г) метод автоматичної перевірки виконаних завдань шляхом формалізації стану віртуального навчального середовища у вигляді впорядкованого вектора артефактів. Обґрунтовано використання даних методів для побудови комплексної інформаційної технології.

У четвертому розділі представлено результати, що стосуються розробки нової інформаційної технології персоналізації навчання студентів інженерних спеціальностей. Наведено архітектуру підсистеми та опис модулів. Також проведено оцінку ефективності запропонованих моделей, методів та інформаційної технології шляхом проведення експериментів за участю студентів із використанням реальних завдань та різних AI-моделей.

Основні результати дослідження та наукова новизна роботи полягають в розробці моделей і методів інформаційного забезпечення процесів персоналізованого навчання, теоретичних та методичних засад автоматизації та створенні на їх основі інформаційної технології комплексного керування життєвим циклом завдання, що дозволяє масштабувати персоналізоване навчання для набуття студентами сталих практичних навичок.

Вперше:

- розроблена функціональна модель персоналізованого практичного інженерного завдання, яка, на відміну від існуючих, визначає повну послідовність етапів його життєвого циклу від створення до оцінювання результатів з урахуванням контексту та необхідних ресурсів, що формує уніфікований підхід до програмної підтримки практичної підготовки з інженерних спеціальностей в процесі електронного навчання;
- розроблена домен-специфічна мова опису практичних завдань Learning Task Definition Language, граматику якої, на відміну від існуючих, охоплює повний життєвий цикл практичного завдання в одному формальному визначенні, що

забезпечує підтримку процесу персоналізованого навчання в автоматичному режимі;

- запропоновано архітектуру інтелектуального асистента, в якій, на відміну від існуючих, задіяна мультиагентна система, що реалізує BDI-парадигму в інтерпретації персоналізованого навчання з урахуванням формального визначення практичного завдання мовою LTDL, що забезпечує підвищення рівня персоналізації за рахунок ітераційної адаптації завдань під індивідуальну траєкторію навчання студента.

Удосконалено:

- методи автоматизації процесів генерації персоналізованих практичних завдань, їх масштабування, розгортання середовищ виконання та перевірки результатів, які, на відміну від відомих, ґрунтуються на інтеграції генеративних можливостей штучного інтелекту з формалізованим описом завдань мовою LTDL, що забезпечує дотримання академічної доброчесності та підвищення ефективності електронного навчання з одночасним скороченням часу набуття студентами сталих практичних навичок.

Практичне значення отриманих результатів. Отримані наукові результати у своїй сукупності утворюють нову інформаційну технологію персоналізації навчання студентів інженерних спеціальностей, яка комплексно забезпечує масштабовану автоматизовану генерацію персоналізованих практичних інженерних завдань, автоматичне розгортання середовища їх виконання та автоматичну перевірку результатів. Запропонована інформаційна технологія може бути використана для забезпечення набуття студентами інженерних спеціальностей сталих практичних навичок. Розроблені програмні засоби також можуть стати основою для створення більш потужних та ефективних систем навчання студентів інженерних спеціальностей.

Результати дисертаційного дослідження впроваджені:

- в рамках реалізації міжнародного наукового проєкту “Цифрова трансформація освітнього процесу ЗВО в Україні та Молдові для сталого співробітництва з

підприємствами” в рамках програми ERASMUS + “Розвиток потенціалу вищої освіти”

Ідентифікатор

проєкту:

01127683-DIGITRANS-ERASMUS-EDU-2023-CBHE (сертифікат про впровадження від 09 квітня 2026 р., Додаток Б).

- у навчальному процесі НУ «Чернігівська політехніка» при проведенні лекцій та лабораторних робіт з дисциплін “Операційні системи”, “Організація комп’ютерних мереж” та “Сучасні телекомунікаційні системи та IP-телефонія” в процесі навчання бакалаврів та магістрів спеціальності F7 (123) – комп’ютерна інженерія та з дисципліни «Методи та технології математичного та комп’ютерного моделювання складних систем» в процесі навчання аспірантів спеціальності F3 (122) – комп’ютерні науки (довідка про впровадження No202/08-590 від 02 квітня 2026 р., Додаток Б) відповідно до плану науково–дослідної роботи НУ «Чернігівська політехніка» (НДР “Цифрове навчальне середовище із віддаленим доступом”, державний реєстраційний номер 0125U000505).
- у Навчальному центрі PortaOne під час викладання публічних курсів з адміністрування ОС Linux та комп’ютерних мереж (довідка про впровадження від 01 квітня 2026 р., Додаток Б).
- в освітній діяльності компанії SendPulse Inc (сертифікат про впровадження від 02 квітня 2026 р., Додаток Б).

Ключові слова: персоналізоване навчання, практичні завдання, цифрова освітня екосистема, автоматична генерація завдань, автоматична перевірка завдань, формальні системи, формальні моделі, формалізація, штучний інтелект, генерація з доповненим пошуком, RAG, боти, AI агенти, віртуальне навчальне середовище, віртуальна машина, інформаційні технології, IT-системи.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці з основними науковими результатами дисертації

1. Khyzhniak, A. V., & Prila, O. A. (2025). Rozrobka systemy avtomatyzovanoi heneratsii ta perevirky parametryzovanykh praktychnykh zavdan [Designing a system for automated generation and automated assessment of parameterized practical assignments]. *Technical sciences and technologies*, (2(40)), 221–233. [https://doi.org/10.25140/2411-5363-2025-2\(40\)-221-233](https://doi.org/10.25140/2411-5363-2025-2(40)-221-233) (1.52 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології, UML-проектування компонентів підсистеми персоналізованого практичного навчання) (1.06 ум. друк. арк.)
2. Khyzhniak, A., & Kazymyr, V. (2025). Domenno-orientovana mova opysu personalizovanykh praktychnykh zavdan dlia inzhenernykh spetsialnostei [Domain-Specific Language for Describing Personalized Practical Tasks in Engineering Disciplines]. *Technical sciences and technologies*, (3 (41)), 261–271. [https://doi.org/10.25140/2411-5363-2025-3\(41\)-261-271](https://doi.org/10.25140/2411-5363-2025-3(41)-261-271) (1,28 ум. друк. арк.) (Особистий внесок здобувача: розробка формальної граматики домен-специфічної мови, тестування граматики, парсер та транслятор домен-специфічної мови для опису практичних інженерних завдань, проектування графічного редактора мови) (0,64 ум. друк. арк.)
3. Khyzhniak, A. V., & Kazymyr, V. V. (2025). Analysis of methods for supporting personalization in IT education. *Herald of Advanced Information Technology*, 8(3), 366–381. <https://doi.org/10.15276/hait.08.2025.24> (SCOPUS) (1,87 ум. друк. арк.) (Особистий внесок здобувача: системний аналіз існуючих методів персоналізації практичних завдань, виявити обмеження існуючих підходів) (0,93 ум. друк. арк.)
4. Khyzhniak, A.V., & Kazymyr, V.V. (2025). Integrated task generation, execution, and assessment methods for enhancing personalized learning. *Nauka i tekhnika*

syohodni, 13(54), 1650–1664.

[https://doi.org/10.52058/2786-6025-2025-13\(54\)-1650-1664](https://doi.org/10.52058/2786-6025-2025-13(54)-1650-1664) (1,75 ум. друк. арк.)

(Особистий внесок здобувача: удосконалення методів автоматичної генерації практичних завдань, автоматичного розгортання навчального середовища та автоматичної перевірки таких завдань) (0,88 ум. друк. арк.)

5. Khyzhniak, A. V., Kazymyr, V. V., & Mylytsia, A. Yu. (2025). The information technology for automated personalized practical tasks creation and assessment. *Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky*, (6), 174–185. <https://doi.org/10.32782/tnv-tech.2025.6.18> (Особистий внесок здобувача: розробка інформаційної технології персоналізації навчання, яка забезпечує інтеграцію моделей, методів та програмних засобів у єдиний комплекс автоматизованої генерації, виконання та перевірки практичних завдань) (0,47 ум. друк. арк.)
6. Khyzhniak, A. V., & Kazymyr, V. V. (2026). Modeli personalizatsii navchannia v tsyfrovomu osvithomu seredovyshchi [Models of personalized learning in a digital educational environment]. *Technical sciences and technologies*, (1(43)), 279–290. [https://doi.org/10.25140/2411-5363-2026-1\(43\)-279-290](https://doi.org/10.25140/2411-5363-2026-1(43)-279-290) (1,40 ум. друк. арк.) (Особистий внесок здобувача: функціональна модель життєвого циклу персоналізованого практичного завдання, формальна модель практичного завдання, архітектура персонального AI-асистента) (0.7 ум. друк. арк.)

Список публікацій за темою дисертації

1. Miziuk, V. A., Khyzhniak, A. V., & Khrenova, V. V. (2025). Vykorystannia adaptivnykh navchalnykh platform dlia personalizatsii dystantsiinoho navchannia [Use of adaptive learning platforms for personalization of distance learning]. *Pedahohichna akademiia: Naukovi zapysky*. <https://doi.org/10.5281/zenodo.14605125> (2,45 ум. друк. арк.) (Особистий внесок здобувача: аналіз навчальних платформ адаптивного навчання) (0,82 ум. друк. арк.)

2. Khyzhniak A., Mylytsia A. Opportunities for using blockchain-technology and NFTs in building a digital learning environment. *Nauka i tekhnika syohodni*, 6(47), 2025, 864-874. [https://doi.org/10.52058/2786-6025-2025-6\(47\)-864-874](https://doi.org/10.52058/2786-6025-2025-6(47)-864-874) (Особистий внесок здобувача: побудова цифрового навчального середовища) (0,64 ум. друк. арк.)
3. Khyzhniak, A., & Kazymyr, V. (2025). Uzahalnena klasyfikatsiia rivniv personalizatsii praktychnykh zavdan v IT-osviti [Generalized classification of personalization levels in practical assignments in IT education]. *Nauka i tekhnika syohodni*, 7(48), 1932–1949. [https://doi.org/10.52058/2786-6025-2025-7\(48\)-1932-1949](https://doi.org/10.52058/2786-6025-2025-7(48)-1932-1949) (2,10 ум. друк. арк.) (Особистий внесок здобувача: система кількісних показників оцінки персоналізації для наукового аналізу, порівняння, та подальшого вдосконалення персоналізованого навчання) (1.05 ум. друк. арк.)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. A.Khyzhniak, O.Prila, P.Byvoino, S.Lytvyn. The necessity, preconditions and consequences of using gamification in the educational process. Новітні технології у науковій діяльності і навчальному процесі : збірник тез доповідей Всеукр. наук.-практ. конф. студентів, аспірантів та молодих учених (м. Чернігів, 19- 20 квітня 2023 р.) . – Чернігів : НУ «Чернігівська політехніка» 2023. с 56-58. Режим доступу: <http://ir.stu.cn.ua/handle/123456789/27778> (0,35 ум. друк. арк.) (Особистий внесок здобувача: використання гейміфікації в навчальному процесі для підвищення персоналізації) (0,09 ум. друк. арк.)
2. Хижняк А. В, Киселиця С. В. Переваги та ризики гейміфікації в соціально-освітньому контексті. Юність науки – 2023: соціально-економічні та гуманітарні аспекти розвитку суспільства : збірник тез доповідей XIII Міжнародної науково-практичної конференції студентів, аспірантів і молодих вчених (м. Чернігів, 26-27 квітня 2023 р.). – Чернігів : НУ «Чернігівська політехніка», 2023. – 770 с. Режим доступу:

- <http://ir.stu.cn.ua/handle/123456789/28308> (0,35 ум. друк. арк.) (Особистий внесок здобувача: вплив гейміфікації на навчання) (0,18 ум. друк. арк.)
3. Khyzhniak Andrii, Olga Prila, Svitlana Lytvyn An automated system for generating personalized practical tasks and their automatic assessment in distance learning. Юність науки – 2024 : збірник тез доповідей XIV Міжнар. наук.-практ. конф. студ., асп. і мол. вчен. (м. Чернігів, 24-26 квіт. 2024 р.). – Чернігів : НУ «Чернігівська політехніка», 2024. pp. 765-767. Режим доступу: <http://ir.stu.cn.ua/handle/123456789/30262> (0,35 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,12 ум. друк. арк.)
 4. Хижняк А.В., Пріла О.А. Актуальні проблеми дистанційного навчання ІТ-спеціалістів. Новітні технології сучасного суспільства (НТСС-2024) : V Міжнародна науково-практична конференція (м. Чернігів, 12 грудня 2024 р.) : тези доповідей – Чернігів : НУ «Чернігівська політехніка», 2025. – 346 с. Режим доступу: https://inel.stu.cn.ua/ntss/NTSS_2024_zbirnyk.pdf (0,35 ум. друк. арк.) (Особистий внесок здобувача: проблеми контролю академічної доброчесності при дистанційному навчанні) (0,18 ум. друк. арк.)
 5. Khyzhniak A., Prila O. Modeling and implementation of the system for automatized generation and evaluation of personalized practical assessments. Global Trends in Science, Technology, and Economy: Collection of Scientific Papers "International Scientific Unity" with Proceedings of the 1st International Scientific and Practical Conference. April 16-18, 2025. Graz, Austria. 328 p. DOI: 10.70286/ISU-16.04.2025. (0,35 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,18 ум. друк. арк.)
 6. Хижняк А.В., Пріла О.А. Концептуальна модель системи автоматизації створення та оцінювання персоналізованих практичних завдань. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 98): матеріали Міжнародної наукової інтернет-конференції, (м. Тернопіль,

- Україна, м. Ополе, Польща, 15-16 квітня 2025 р.) /редкол. : О. Патряк та ін. ГО “Наукова спільнота”, WSZIA w Opolu. Тернопіль : ФОП Шпак В.Б. 2025. 82 с. ISSN 2522-932X. Режим доступу: http://www.konferenciaonline.org.ua/data/downloads/file_1747665751.pdf (0,47 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,23 ум. друк. арк.)
7. Khyzhniak A., Kazymyr V., Zabasta A. Domain-Specific Language for personalized practical learning tasks description. Abstracts of XXVII International Scientific and Practical Conference. July 07-09, 2025. Munich, Germany. pp.127-130. Режим доступу: <https://eu-conf.com/wp-content/uploads/2025/05/CURRENT-TRENDS-IN-THE-DEVELOPMENT-OF-MODERN-EDUCATIONAL-TECHNOLOGIES.pdf> (0,47 ум. друк. арк.) (Особистий внесок здобувача: вимоги до DSL, яка покриває домен персоналізованих практичних інженерних завдань, формальна граматики домен-специфічної мови) (0,16 ум. друк. арк.)
8. Khyzhniak A. Levels and Index of Personalization of Practical Tasks in IT Education. 5th International on educational technology conference and online learning, ICETOL-2025. Abstract proceedings. 26-29 August 2025. Balikesir, Turkey. Режим доступу: https://www.icetol.com/wp-content/uploads/2025/10/icetol2025_abstract_proceedings.pdf (0,12 ум. друк. арк.) (Особистий внесок здобувача: система кількісних показників оцінки персоналізації) (0,12 ум. друк. арк.)
9. Khyzhniak A. Information Technology for Personalized Practical Task Creation and Assessment. 5th International on educational technology conference and online learning, ICETOL-2025. Abstract proceedings. 26-29 August 2025. Balikesir, Turkey. Режим доступу: https://www.icetol.com/wp-content/uploads/2025/10/icetol2025_abstract_proceedings.pdf (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,12 ум. друк. арк.)

10. Khyzhniak A. A Domain-Specific Language for Describing Personalized Practical Tasks in IT Education. 5th International on educational technology conference and online learning, ICETOL-2025. Abstract proceedings. 26-29 August 2025. Balikesir, Turkey. Режим доступу: https://www.icetol.com/wp-content/uploads/2025/10/icetol2025_abstract_proceedings.pdf (0,12 ум. друк. арк.) (Особистий внесок здобувача: вимоги до DSL, яка покриває домен персоналізованих практичних інженерних завдань, формальна граматики домен-специфічної мови) (0,12 ум. друк. арк.)
11. Khyzhniak A. Models and methods of personalization in engineering education. Modern Challenges in Economic and Technological Innovation: Collection of Scientific Papers with Proceedings of the 1st International Scientific and Practical Conference. International Scientific Unity. October 15-17, 2025. Bologna, Italy. pp. 158-162. DOI: <https://doi.org/10.70286/isu-15.10.2025> (0,58 ум. друк. арк.) (Особистий внесок здобувача: удосконалення методів автоматичної генерації практичних завдань, автоматичного розгортання навчального середовища та автоматичної перевірки таких завдань) (0,58 ум. друк. арк.)
12. Хижняк А.В., Казимир В.В. Вбудована модель персоналізованого практичного завдання в онлайн-освіті ІТ-фахівців. МОДС 2025: тези доповідей XX міжнародної конференції (10 – 12 листопада 2025 р., м. Чернігів) / М-во освіти і науки України; Нац. Акад. наук України; Академія технологічних наук України; Інженерна академія України та ін. – Чернігів : НУ «Чернігівська політехніка», 2025. – с.67-72. (0,70 ум. друк. арк.) (Особистий внесок здобувача: формальну модель практичного завдання, що визначає його структуру, параметри, середовище виконання та критерії оцінювання, придатну до автоматизованої обробки; порівняння можливостей різних LLM-моделей для автоматизованої генерації) (0,35 ум. друк. арк.)
13. Khyzhniak Andrii, Kazymyr Volodymyr. Hybrid methods for automated generation, deployment, and verification of personalized practical tasks. Новітні технології

- сучасного суспільства (НТСС-2025): VI Міжнародна науково-практична конференція (м. Чернігів, 11 грудня 2025 р.) : тези доповідей – Чернігів : НУ «Чернігівська політехніка», 2026.рр 164-165. (0,35 ум. друк. арк.) (Особистий внесок здобувача: удосконалення методів автоматичної генерації практичних завдань, автоматичного розгортання навчального середовища та автоматичної перевірки таких завдань) (0,18 ум. друк. арк.)
14. Khyzhniak A., Mylytsia A. On the lack of unified requirements for virtual learning environments in IT education. 5th International Scientific and Practical Conference «Modern Trends in the Development of Economy, Technology and Industry» Collection of Scientific Papers. January 7-9, 2026. Toronto, Canada.рр 515-517. (0,35 ум. друк. арк.) (Особистий внесок здобувача: визначення вимог до формалізації практичних завдань і навчальних середовищ) (0,18 ум. друк. арк.)
15. Хижняк А., Милиця А., Казнадій С., Горваль Д., Бобришев Є. Проблеми та перспективи сучасного практичного навчання інженерів. Львівський науковий форум. Матеріали XVII міжнародної науково-практичної конференції “Пріоритетні напрями досліджень в науковій та освітній діяльності”. 9-10 січня 2026 року. Львів, Україна. С. 49-53. (0,58 ум. друк. арк.) (Особистий внесок здобувача: концепція персоналізованого навчання інженерних спеціальностей) (0.12 ум. друк. арк.)

ABSTRACT

Khyzhniak Andrii. Models, Methods and Information Technology of personalized learning in engineering specialties. - Qualifying scientific work on the rights of manuscript.

PhD thesis under Specialty 122 – Computer Science (12 – Information technologies). – Chernihiv Polytechnic National University, Ministry of Education and Science of Ukraine, Chernihiv, 2026.

The dissertation research is a completed scientific work in which a specific scientific problem has been solved, namely the comprehensive support of processes for automated generation of personalized practical engineering tasks, automatic deployment of the required virtual learning environments, and automatic assessment of task execution results, aimed at enhancing the level of personalized learning and facilitating the acquisition of sustainable practical skills by students.

The **object** of the research is the information support of the personalized learning process for engineering students. The **subject** of this study is the models, methods and elements of information technology used to personalize learning for engineering specialties.

The **aim** of the dissertation is to enhance the level of personalized learning of students and to ensure the acquisition of sustainable practical skills through the scalable automation of the processes of creation, execution, and assessment of personalized practical engineering tasks, while ensuring compliance with the principles of academic integrity.

The **introduction** justifies the relevance of the research topic, outlines the purpose, objectives, and methods of the study, and highlights the connection between the research and the department's academic programs; it also describes the scientific novelty and practical significance of the dissertation's findings.

The first chapter analyzes personalized practical tasks in the education of engineering students, examines methods of personalized learning, identifies and outlines

their main limitations, and promising methods of personalization; the potential of AI for personalization is explored, and approaches to formalizing tasks in engineering education are analyzed; the necessity of creating functional and formal models suitable for automated generation, parameterization, automatic deployment, and verification is substantiated. Requirements for virtual learning environments have been formulated. The relevance of the research is presented, scientific gaps are identified, and research objectives are formulated.

The second chapter establishes the formal basis of the study, within which an approach to the centralized formalization of key elements of personalized learning is implemented. In particular, for the first time, a functional model of a personalized practical engineering task is developed, which formalizes the sequence of its stages (i.e., its life cycle) and ensures a consistent linkage between them, as well as a formal model that defines its structure, main components, and interrelationships. For the first time, a domain-specific language for describing learning tasks, Learning Task Definition Language, is developed, which serves as a unified means for the formal description of practical tasks, their parameters, execution conditions, environments, and assessment criteria. The language ensures unambiguity, machine-readability, and formal interpretability of tasks, which is a necessary prerequisite for the automated generation, automatic deployment, and verification of execution results, as well as for further scalability. Such formalization provides a foundation for the automated processing of practical tasks across different engineering specializations.

In the third chapter, an intelligent assistant architecture is proposed for the first time, and the processes for learning support, content generation, and adaptation to individual student characteristics are formalized. The intelligent assistant architecture is based on a multi-agent system for the orchestration of AI agents and is integrated with the developed domain-specific language for practical task description, LTDL. Four interrelated methods for the automation of personalized practical engineering tasks have been enhanced, namely: a) a method of automated generation based on GenAI and the developed domain-specific language LTDL, b) the method of automatic deployment based

on LTDL and a personal AI assistant, c) the method of parameterizing practical tasks by combining stochastic parameter generation with the formal constraints of the LTDL language, d) the method of automatic verification of completed tasks by formalizing the state of the virtual learning environment as an ordered vector of artifacts. The use of these methods for building a comprehensive information technology is justified.

The fourth chapter presents the results related to the development of a new information technology for personalized learning of engineering students. The architecture of the subsystem and the description of its modules are provided. The effectiveness of the proposed models, methods, and information technology is evaluated through experiments involving students, using real tasks and various AI models.

The main findings of the study and the scientific novelty of this work lie in the development of models and methods for personalized learning and the creation, based on them, of information technology for comprehensive task lifecycle management, which enables the scaling of personalized learning to help engineers acquire sustainable practical skills.

For the first time:

- a functional model of a personalized practical engineering task has been developed which, unlike existing approaches, defines the complete sequence of stages of its life cycle from creation to assessment of results, taking into account context and required resources, thereby establishing a unified approach to software support of practical training in engineering education within e-learning environments;
- a domain-specific language for describing practical tasks, Learning Task Definition Language (LTDL), has been developed, whose grammar, unlike existing ones, covers the entire life cycle of a practical task within a single formal definition, thus enabling support for the personalized learning process in an automatic mode;
- an architecture of an intelligent assistant has been proposed which, unlike existing approaches, employs a multi-agent system implementing the BDI paradigm in the context of personalized learning based on the formal definition of practical tasks using LTDL, thereby enhancing the level of personalization through iterative

adaptation of tasks to the individual learning trajectory of a student.

Have been improved:

- methods for automating the processes of generation of personalized practical tasks, their scalability, deployment of execution environments, and assessment of results, which, unlike existing methods, are based on the integration of generative AI capabilities with the formalized task description using LTDL, ensuring compliance with academic integrity and improving the effectiveness of e-learning while reducing the time required for students to acquire sustainable practical skills.

Practical value of the obtained results. The obtained scientific results collectively form a new information technology for personalized learning of engineering students, which provides comprehensive support for the scalable automated generation of personalized practical engineering tasks, automatic deployment of their execution environments, and automatic assessment of results. The proposed information technology can be used to support the acquisition of sustainable practical skills by engineering students. The developed software tools can also serve as a foundation for the creation of more advanced and efficient learning systems for engineering education.

The results of the dissertation research have been implemented:

- in the international research project “Digital Transformation of the Educational Process at Higher Education Institutions in Ukraine and Moldova for Sustainable Cooperation with Businesses” under the ERASMUS+ program “Capacity Building in Higher Education.” Project ID: 01127683-DIGITRANS-ERASMUS-EDU-2023-CBHE (Certificate of Implementation dated April 9, 2026).
- in the educational process of “Chernihiv Polytechnic” National University during lectures and laboratory works for the disciplines “Operating Systems,” “Computer Network Organization,” and “Modern Telecommunication Systems and IP-telephony” for bachelor's and master's students of specialty 123 – “Computer Engineering,” and for the discipline “Methods and Technologies of Mathematical and Computer Modeling of Complex Systems” for PhD students of specialty 122 –

“Computer Science” (Implementation Report No. 202/08-590 dated April 2, 2026) in accordance with the research plan of Chernihiv Polytechnic National University (Research Project “Digital Learning Environment with Remote Access,” state registration number 0125U000505).

- at the PortaOne Training Center when teaching public courses on Linux OS administration and computer networks (Certificate of Implementation dated April 1, 2026).
- in the educational activities of SendPulse Inc. (Certificate of Implementation dated April 2, 2026).

Keywords: personalized learning, practical tasks, digital learning ecosystem, automatic task generation, automatic task verification, formal systems, formal models, formalisation, artificial intelligence, Retrieval-Augmented Generation, RAG, bots, AI agents, virtual learning environment, virtual machine, information technology, IT systems.

ЗМІСТ

АНОТАЦІЯ	2
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ	7
ABSTRACT	14
ЗМІСТ	19
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	21
ВСТУП	22
РОЗДІЛ I. ПЕРСОНАЛІЗОВАНЕ НАВЧАННЯ В ІНЖЕНЕРНІЙ ОСВІТІ	33
1.1 Сутність та переваги персоналізованого навчання	33
1.2 Особливості навчання студентів інженерних спеціальностей	35
1.3 Аналіз методів персоналізації, їх можливостей та обмежень	44
1.4 Вимоги до VLE	53
1.5 Засоби формалізації в освітніх технологіях	57
1.6 Постановка задач дослідження	67
1.7 Висновки до розділу I	69
РОЗДІЛ II. МОДЕЛІ ПЕРСОНАЛІЗОВАНОВОГО НАВЧАННЯ З ІНЖЕНЕРНИХ СПЕЦІАЛЬНОСТЕЙ	71
2.1 Функціональна модель PPET	71
2.2 Формальна модель PPET	76
2.3 Формальна мова опису PPET	81
2.4 Висновки до розділу II	90
РОЗДІЛ III. МЕТОДИ ПЕРСОНАЛІЗОВАНОВОГО НАВЧАННЯ З ІНЖЕНЕРНИХ СПЕЦІАЛЬНОСТЕЙ	92
3.1 Архітектура інтелектуального асистента	93
3.2 Метод автоматизованої генерації PPET	102
3.3 Метод параметризації PPET	109
3.4 Метод автоматичного розгортання VLE	111
3.5 Метод автоматичної перевірки результатів виконання PPET	113
3.6 Варіанти використання методів	117
3.7 Висновки до розділу III	125
РОЗДІЛ IV. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРСОНАЛІЗОВАНОВОГО НАВЧАННЯ З ІНЖЕНЕРНИХ СПЕЦІАЛЬНОСТЕЙ	127
4.1 Складові інформаційної технології	127
4.2 Архітектура підсистеми автоматичного розгортання VLE	137
4.3 Програмний інтерфейс керування VLE	141
4.4 Експериментальне дослідження інформаційної технології	146

4.5 Висновки до розділу IV	167
ВИСНОВКИ	171
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	177
ДОДАТКИ	197
Додаток А. Список публікацій здобувача за темою дисертації	197
Додаток Б. Довідки про впровадження	204
Додаток В. Сертифікати участі в конференціях	207
Додаток Г. Засоби реалізації вимог до VLE	212
Додаток Д. Застосування процедури декомпозиції для типових завдань	213
Додаток Е. Опис компонентів формальної моделі PPEТ	214
Додаток Є. Тестування розробленої мови LTDL	226
Додаток Ж. Графічні зображення символів алфавіту мови LTDL	230
Додаток З. Системний промт для LLM	232
Додаток І. Оцінка якості генерації завдань	233

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI – Artificial Intelligence (штучний інтелект)

API – Application Programming Interface (програмний інтерфейс застосунків)

BRF – Belief Revision Function (функцію перегляду переконань)

DLE – Digital Learning Ecosystem (цифрова освітня екосистема)

EBNF – Extended Backus-Naur form (розширена нотація Бекуса-Наура)

DSL – Domain Specific Language (домен-специфічна мова)

GenAI – Generative Artificial Intelligence (генеративний штучний інтелект)

IT – Information Technology (інформаційна технологія)

LLM – Large Language Model (велика мовна модель)

LMS – Learning Management System (система управління навчанням)

LTDL – Learning Task Definition Language (мова опису навчальних завдань)

PAIA – Personal AI-Assistant (персональний AI-асистент)

PAL – Personalized Adaptive Learning (персоналізоване адаптивне навчання)

PPET – Personalized Practical Engineering Task (персоналізоване практичне інженерне завдання)

RAG – Retrieval Augmented Generation (доповнене пошуком генерування)

TSCI – Task Structural Complexity Index (індекс структурної комплексності завдання)

VLAN – Virtual Local Area Network (віртуальна локальна мережа)

VLE – Virtual Learning Environment (віртуальне навчальне середовище)

VM – Virtual Machine (віртуальна машина)

ВСТУП

Актуальність теми дослідження. У сучасних дослідженнях інженерної та комп'ютерної освіти підкреслюється ключова роль практичних (лабораторних) робіт як важливого компонента навчального процесу, адже розвиток сталих і затребуваних на ринку праці навичок програмування, мережевого адміністрування, аналізу даних тощо, а також формування експертності в цих галузях значною мірою пов'язане з виконанням практичних завдань. Теоретичні знання (які оцінюються тестами) та практичні навички (які формуються під час роботи в середовищі) є різними компетентностями. При цьому відзначаються обмеження використання традиційних теоретичних тестів для оцінювання рівня практичної підготовки, що посилює актуальність теми дослідження.

У сучасному дистанційному асинхронному навчанні спостерігається зростання ризиків академічної недоброчесності студентів, що зумовлено послабленням контролю за самотійним виконанням практичних завдань. Недостатня якість самотійного опрацювання практичних завдань призводить до зниження рівня підготовки студентів інженерних спеціальностей. Одним із перспективних напрямів підвищення академічної доброчесності та якості засвоєння знань розглядається впровадження персоналізованих підходів у навчанні, проте такі підходи зазвичай вимагають значних ресурсів з боку викладача. Застосування методів штучного інтелекту (Artificial Intelligence – AI) є одним із ключових чинників забезпечення персоналізації навчання та динамічної адаптації освітніх матеріалів, що є важливим для формування глибшого розуміння і якісного засвоєння матеріалу. З урахуванням розвитку технологій AI, машинного навчання та Big Data, персоналізоване навчання отримує додаткові можливості для автоматизації, де AI відіграє суттєву роль у розширенні можливостей освітніх систем.

Отже, потреба в інформаційному забезпеченні та автоматизації процесів персоналізованого навчання є значною. Автоматизація процесів, своєю чергою, є необхідною передумовою масштабованої персоналізації, що особливо важливо для

вищих навчальних закладів. У дослідженні це досягнуто шляхом удосконалення методів персоналізації (автоматизованої генерації, автоматичної перевірки та автоматичного розгортання навчальних середовищ) за рахунок використання можливостей AI та створення формального базису. Для автоматизації процесів генерації та перевірки завдань важливо, щоб завдання були структурованими, машиночитаними та параметризованими. У дослідженні зазначені властивості забезпечуються завдяки формальному визначенню їх внутрішньої структури з використанням формальних моделей і розробленої формальної домен-специфічної мови (Domain Specific Language – DSL), за допомогою яких спрощується процес розробки навчальних компонентів.

Ретельний аналіз наукових джерел свідчить про необхідність розроблення методів та інструментів, які б забезпечили автоматизовану генерацію та оцінювання персоналізованих практичних завдань, включаючи автоматичне розгортання і налаштування середовищ, в яких студенти виконують лабораторні практичні завдання, та підтверджує актуальність дослідження. Аналіз існуючих підходів показує відсутність єдиного узагальненого підходу до формалізації структури практичних завдань, що ускладнює масштабовану автоматизацію процесів їх генерації та перевірки. Таким чином, постає завдання розроблення методів і засобів, які б забезпечили масштабовану персоналізацію навчання при збереженні академічної доброчесності.

У цьому дослідженні обмежено предметну область практичними завданнями інженерного спрямування, що відповідає специфіці підготовки фахівців у галузі IT та інженерії. Інші типи навчальних завдань (теоретичні, тестові) не розглядаються. Основну увагу зосереджено на персоналізованих завданнях як на ефективному підході до формування сталих практичних навичок. З урахуванням цього, у роботі використовується поняття персоналізованого практичного інженерного завдання (Personalized Practical Engineering Task – PPET), як базової одиниці аналізу.

Освітні технології та методи персоналізації постійно вдосконалюються, особливо активно протягом останніх років з урахуванням нових можливостей AI. У

наукових дослідженнях розглядаються різні підходи до персоналізації навчання, зокрема у працях науковців Atikah Shemshack, Jonathan Michael Spector, Amir Hossein Nabizadeh, Benjamin Clément, Avi Segal, Leonard Tetzlaff, Sven Jacobs, Henning Peters, Steffen Jaschke, Natalie Kiesler, Umar Alkafaween, Ibrahim Albluwi, Paul Denny. Над вдосконаленням методів автоматизованої генерації та перевірки також працювали Jan Vykopal, Valdemar Švábenský, Pavel Seda, Pavel Čeleda, Isaac Agudo, Ruben Rios, Ana Nieto, Johnny Chan, Songyan Teng, Francisco de Assis Zampirolli, Paulo Henrique Pisani, João Marcelo Josko, Guiou Kobayashi, Francisco Fraga, Denise Goya. Детальний аналіз існуючих підходів та відповідних наукових джерел наведено у розділі I.

Проте роботи, присвячені автоматизації процесів генерації та перевірки практичних завдань, характеризуються низкою обмежень, що пов'язані із технічною складністю впровадження, обмеженими можливостями генерації завдань та автоматизації перевірки, потребою у додатковій верифікації завдань, згенерованих AI, неповною інтеграцією з системами управління навчанням (Learning Management System – LMS) та ризиками компрометації завдань при локальному виконанні. Існуючі DSL орієнтовані на окремі аспекти задачі та не забезпечують комплексного формального опису практичних завдань, що зумовлює концептуальні та технологічні обмеження щодо опису типів завдань, способів перевірки, рівнів складності, контекстів виконання тощо. Ці фактори обумовлюють доцільність удосконалення існуючих методів та розробки нових підходів, а також створення на їх базі гнучких, масштабованих і простих у використанні рішень для підтримки персоналізованого навчання в цифрових освітніх екосистемах (Digital Learning Ecosystem – DLE).

У роботі запропоновано комплексна інформаційна технологія персоналізації навчання студентів інженерних спеціальностей, функціональна та формальна моделі PPET, нова домен-специфічна мова та удосконалення методів персоналізації. Запропонований підхід спрямований на підвищення рівня персоналізованого навчання студентів, автоматизацію процесів виконання і оцінювання практичних завдань з метою набуття студентами сталих практичних навичок. Отже, розробка

формальних моделей та формальної мови, а також удосконалення на їх основі методів персоналізації та побудова інформаційної технології персоналізації навчання студентів інженерних спеціальностей, з урахуванням необхідності набуття студентами сталих практичних навичок є актуальним науковим завданням. Ця робота також підкреслює важливість створення автентичних контрольованих VLE середовищ для відпрацювання практичних навичок і підвищення ефективності підготовки фахівців інженерних спеціальностей.

Зв'язок роботи з науковими програмами, планами, темами. Представлена дисертаційна робота виконана відповідно до плану науково–дослідної роботи НУ «Чернігівська політехніка» (НДР “Цифрове навчальне середовище із віддаленим доступом”, державний реєстраційний номер 0125U000505) та частково в рамках реалізації міжнародного наукового проєкту “Цифрова трансформація освітнього процесу ЗВО в Україні та Молдові для сталого співробітництва з підприємствами” в рамках програми ERASMUS+ “Розвиток потенціалу вищої освіти” (ідентифікатор проєкту: 01127683-DIGITRANS-ERASMUS-EDU-2023-CBHE).

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення рівня персоналізованого навчання студентів та набуття ними сталих практичних навичок за рахунок масштабованої автоматизації процесів створення, виконання та оцінювання результатів виконання персоналізованих практичних інженерних завдань з дотриманням норм академічної доброчесності. У роботі використано непрямі індикатори сформованості практичних навичок, зокрема успішність і завершуваність виконання завдань, стабільність та відтворюваність результатів у контрольованому середовищі.

Завдання дослідження полягає в розробці нової інформаційної технології персоналізації навчання студентів інженерних спеціальностей по комплексному забезпеченню процесів автоматизованої генерації персоналізованих практичних інженерних завдань, автоматичного розгортання необхідних віртуальних навчальних середовищ (Virtual Learning Environment – VLE) та автоматичної перевірки результатів виконання цих завдань для підвищення рівня персоналізованого

навчання та набуття студентами сталих практичних навичок за рахунок використання формальних моделей, формальної мови та можливостей AI.

Для досягнення мети передбачається вирішення таких задач:

- виконати аналіз існуючих методів персоналізації практичних завдань у підготовці студентів інженерних спеціальностей;
- розробити функціональну та формальну моделі PPET;
- розробити формальну граматику для домен-специфічної мови опису PPET;
- розробити архітектуру інтелектуального асистента для підтримки процесу персоналізованого навчання в автоматичному режимі;
- удосконалити існуючі методи автоматизації процесів масштабованої генерації PPET, розгортання середовищ їх виконання та перевірки результатів виконання завдань з використанням розробленої DSL та існуючих моделей AI;
- розробити інформаційну технологію персоналізації навчання, яка забезпечує інтеграцію запропонованих моделей, методів та програмних засобів у єдиний автоматизований комплекс з одночасним вбудовуванням в середовище електронного навчання;
- оцінити ефективність розробленої інформаційної технології шляхом проведення експериментів за участю реальних студентів.

Об'єктом дослідження є інформаційне забезпечення процесу персоналізованого навчання студентів інженерних спеціальностей. **Предметом дослідження** є моделі, методи та елементи інформаційної технології персоналізації практичного навчання студентів інженерних спеціальностей.

Методи дослідження. В основу методології дослідження покладено 4 рівні аналізу: системний аналіз, концептуальний аналіз, асоціативний аналіз та емпіричний аналіз.

Під час розробки функціональної моделі PPET застосовувалось функціональне моделювання у нотації IDEF0. Запропонована формальна модель PPET була розроблена з використанням змішаного методологічного підходу, що поєднує емпіричний аналіз, індуктивне узагальнення, теоретичний синтез та методи

формальної специфікації. Методи експертних оцінок використовувались для кожної із ітерацій класифікації рівнів персоналізації практичних завдань, також системний і концептуальний аналіз, огляд літератури.

Для удосконалення методів автоматичної генерації та перевірки застосовано методи системного аналізу, генералізації, формалізації, алгоритмізації та методи AI, що забезпечили можливість побудови адаптивних і динамічних механізмів перевірки практичних завдань. Також використано узагальнення власного досвіду в галузі персоналізованої ІТ-освіти та експериментальна перевірка для оцінювання ефективності запропонованих рішень. Крім того, використані методи математичної статистики і методи виявлення залежностей.

Для проєктування компонентів інформаційної технології та розробки програмних засобів, які реалізують запропоновану інформаційну технологію, використовувався дизайн-ітерувальний підхід (design science methodology), UML-проєктування, об'єктно-орієнтований аналіз. Для деталізації функцій системи використовувався метод декомпозиції. Метод структурного аналізу використовувався для візуалізації процесів і моделювання руху даних.

Поєднання зазначених методів забезпечує гнучкість, рефлексивність і наукову обґрунтованість процесу. Достовірність одержаних результатів дослідження підтверджено їх апробацією на 13 науково-практичних і наукових конференціях, впровадженні елементів дослідження в навчальний процес НУ “Чернігівська Політехніка”, в міжнародний науковий проєкт ERASMUS+ “Розвиток потенціалу вищої освіти”, в Навчальному центрі PortaOne при викладанні публічних курсів та освітній діяльності компанії SendPulse Inc.

Основні результати дослідження та наукова новизна роботи полягають в розробці теоретичних та методичних засад автоматизації персоналізованого практичного навчання та створенні на їх основі інформаційної технології для комплексного забезпечення процесів автоматизованої генерації, автоматичної перевірки практичних завдань та розгортання VLE для підвищення рівня персоналізації навчання та набуття студентом сталих практичних навичок.

Вперше:

- розроблена функціональна модель персоналізованого практичного інженерного завдання, яка, на відміну від існуючих, визначає повну послідовність етапів його життєвого циклу від створення до оцінювання результатів з урахуванням контексту та необхідних ресурсів, що формує уніфікований підхід до програмної підтримки практичної підготовки з інженерних спеціальностей в процесі електронного навчання;
- розроблена домен-специфічна мова опису практичних завдань Learning Task Definition Language, граматики якої, на відміну від існуючих, охоплює повний життєвий цикл практичного завдання в одному формальному визначенні, що забезпечує підтримку процесу персоналізованого навчання в автоматичному режимі;
- запропоновано архітектуру інтелектуального асистента, в якій, на відміну від існуючих, задіяна мультиагентна система, що реалізує BDI-парадигму в інтерпретації персоналізованого навчання з урахуванням формального визначення практичного завдання мовою LTDL, що забезпечує підвищення рівня персоналізації за рахунок ітераційної адаптації завдань під індивідуальну траєкторію навчання студента.

Удосконалено:

- методи автоматизації процесів генерації персоналізованих практичних завдань, їх масштабування, розгортання середовищ виконання та перевірки результатів, які, на відміну від відомих, ґрунтуються на інтеграції генеративних можливостей штучного інтелекту з формалізованим описом завдань мовою LTDL, що забезпечує дотримання академічної доброчесності та підвищення ефективності електронного навчання з одночасним скороченням часу набуття студентами сталих практичних навичок.

Практичне значення отриманих результатів. Отримані в роботі наукові результати у своїй сукупності утворюють нову інформаційну технологію персоналізації навчання студентів інженерних спеціальностей при використанні

цифрових освітніх екосистем. Розроблена інформаційна технологія персоналізації навчання студентів інженерних спеціальностей має практичне втілення у вигляді програмного комплексу, до складу якого входять такі програмні засоби:

- підсистема автоматичного розгортання PPET та їх подальшої автоматичної перевірки, яка складається з програмного інтерфейсу застосунків (Application Programming Interface – API), та шлюза на основі FastAPI і мови Python, відповідних модулів генерації, валідації та перевірки PPET, модуля розгортання VLE на основі гіпервізора VirtualBox та контейнеризатора Docker;
- парсер і транслятор на основі розробленої формальної граматики мови LTDL з використанням ANTLR та мови Python для валідації LTDL-файлів опису завдання та створення скриптів розгортання VLE і перевірки прогресу виконання завдання студентами;
- веб-застосунок візуального редактора для графічного представлення мови LTDL, який дозволить викладачам створювати, переглядати та редагувати PPET в зручному форматі. Візуальний редактор створено з використанням мов HTML, CSS, JavaScript, TypeScript та бібліотеки React Flow на основі React;
- програмний засіб PAIA-moodle-plugin для розширення можливостей LMS Moodle, який дозволяє створювати персонального AI-асистента (Personal AI-Assistant, PAIA) та виступає інтерфейсом взаємодії з AI-асистентом для створення PPET та зберігання результатів їх виконання. Плагін створено з використанням мов HTML, CSS, JavaScript, PHP, SQL;
- програмний засіб LTDL-moodle-plugin для розширення можливостей LMS Moodle, який дозволяє створювати контрольні PPET та зберігати результатів їх виконання і створено з використанням мов HTML, CSS, JavaScript, PHP, SQL;
- мультиагентна система оркестрації AI-агентів, побудована на основі технологій Docker та Flowise.

Отриманий від дослідження ефект проявляється у наступному: збільшення кількості унікальних завдань, зменшення випадків академічної недоброочесності, підвищення рівня персоналізації навчання, зменшення навантаження на викладачів і

скорочення часових витрат на генерацію і перевірку завдань, підвищення якості контролю знань студентів, набуття студентами інженерних спеціальностей сталих практичних навичок.

Запропонована інформаційна технологія та (або) її компоненти можуть бути використані викладачами практичних інженерних дисциплін для автоматизованої генерації PPET, автоматичного розгортання VLE та автоматичної перевірки цих завдань та отримання студентами інженерних спеціальностей сталих практичних навичок, затребуваних в сучасному світі. Розроблені формальні моделі і методи є основою для впровадження в освітні процеси більш масштабних, адаптивних та функціональних DLE.

Результати дисертаційного дослідження впроваджені:

- в рамках реалізації міжнародного наукового проєкту “Цифрова трансформація освітнього процесу ЗВО в Україні та Молдові для сталого співробітництва з підприємствами” в рамках програми ERASMUS + “Розвиток потенціалу вищої освіти” Ідентифікатор: 01127683-DIGITRANS-ERASMUS-EDU-2023-CBHE (сертифікат про впровадження від 09 квітня 2026 р., Додаток Б).
- у навчальному процесі НУ «Чернігівська політехніка» при проведенні лекцій та лабораторних робіт з дисциплін “Операційні системи”, “Організація комп’ютерних мереж” та “Сучасні телекомунікаційні системи та IP-телефонія” в процесі навчання бакалаврів та магістрів спеціальності F7 (123) – комп’ютерна інженерія та з дисципліни «Методи та технології математичного та комп’ютерного моделювання складних систем» в процесі навчання аспірантів спеціальності F3 (122) – комп’ютерні науки (довідка про впровадження No202/08-590 від 02 квітня 2026 р., Додаток Б) відповідно до плану науково–дослідної роботи НУ «Чернігівська політехніка» (НДР “Цифрове навчальне середовище із віддаленим доступом”, державний реєстраційний номер 0125U000505).
- у Навчальному центрі PortaOne під час викладання публічних курсів з адміністрування ОС Linux та комп’ютерних мереж (довідка про впровадження

від 01 квітня 2026 р., Додаток Б).

- в освітній діяльності компанії SendPulse Inc (сертифікат про впровадження від 02 квітня 2026 р., Додаток Б).

Особистий внесок здобувача. Усі наукові результати, викладені в дисертаційній роботі і які виносяться до захисту, отримані автором особисто. В наукових роботах, опублікованих у співавторстві, в дисертації використані лише ті ідеї та положення, що є результатом особистої роботи автора.

Апробація результатів дисертації. Основні наукові та практичні результати дисертаційної роботи доповідались та обговорювались на 13 наукових та науково-практичних конференціях. Сертифікати участі наведено в додатку В.

Список наукових та науково-практичних конференцій:

1. Всеукраїнська науково-практична конференція студентів, аспірантів та молодих учених “Новітні технології у науковій діяльності і навчальному процесі” (м.Чернігів, Україна, 19-20 квітня 2023 р.)
2. XIII Міжнародна науково-практична конференція студентів, аспірантів і молодих вчених “ЮНІСТЬ НАУКИ – 2023: соціально-економічні та гуманітарні аспекти розвитку суспільства” (м.Чернігів, Україна, 26-27 квітня 2023 р.)
3. XIV Міжнародна науково-практична конференція студентів, аспірантів і молодих вчених “ЮНІСТЬ НАУКИ – 2024” (м.Чернігів, Україна, 24-26 квітня 2024 р.)
4. V Міжнародна науково-практична конференція “Новітні технології сучасного суспільства НТСС-2024” (м. Чернігів, Україна, 12 грудня 2024 р.)
5. I Міжнародна науково-практична конференція “Світові тенденції в науці, техніці та економіці” (м.Грац, Австрія, 16-18 квітня 2025 р.)
6. Міжнародна наукова інтернет-конференції (м. Тернопіль, Україна, м. Ополе, Польща, 15-16 квітня 2025 р.)
7. XXVII Міжнародна науково-практична конференція “Сучасні тенденції у розвитку сучасних освітніх технологій” (м.Мюнхен, Німеччина, 07-09 липня

2025 р.)

8. V Міжнародна конференція з освітніх технологій та онлайн-навчання, ICETOL-2025 (м.Баликесір, Туреччина, 26-29 серпня 2025 р.).
9. I Міжнародна науково-практична конференція “Сучасні виклики в галузі економічних та технологічних інновацій” (м.Болонья, Італія, 15-17 жовтня 2025 р.)
10. XX Міжнародна конференція “Математичне та імітаційне моделювання систем МОДС 2025” (м. Чернігів, Україна, 10-12 листопада 2025 р.)
11. VI Міжнародна науково-практична конференція “Новітні технології сучасного суспільства НТСС-2025” (м. Чернігів, Україна, 11 грудня 2025 р.)
12. V Міжнародна науково-практична конференція «Сучасні тенденції розвитку економіки, технологій та промисловості» (м.Торонто, Канада, 07-09 січня 2026 р).
13. XVII Міжнародна науково-практична конференція “Пріоритетні напрями досліджень в науковій та освітній діяльності” (м.Львів, Україна, 09-10 січня 2026 р.)

Публікації. За темою дисертаційного дослідження з викладенням основних результатів опубліковано 24 наукові праці. Серед них – 9 статей у наукових виданнях, включених до переліку наукових фахових видань України, 1 з них включена до міжнародної наукометричної бази Scopus [3]; 15 праць апробаційного характеру [10-24]. Результати роботи доповідалися на 13 міжнародних наукових конференціях.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, 4 розділів із висновками, висновків, переліку умовних скорочень, переліку посилань зі 176 джерел та 10 додатків. Загальний обсяг роботи становить 234 сторінки, з яких зміст на 2 сторінках, вступ на 11 сторінках, перелік умовних скорочень на 1 сторінці, основний текст на 176 сторінках, список використаних джерел із 176 найменувань на 20 сторінках, 10 додатків на 38 сторінках. Робота містить 47 рисунки та 28 таблиць.

РОЗДІЛ І. ПЕРСОНАЛІЗОВАНЕ НАВЧАННЯ В ІНЖЕНЕРНІЙ ОСВІТІ

1.1 Сутність та переваги персоналізованого навчання

Персоналізація (персоналізоване навчання) – це підхід, спрямований на розробку ефективної траєкторії засвоєння знань, яка відповідає сильним сторонам учня та враховує його слабкі сторони, щоб зрештою досягти бажаної мети. Підхід включає процес адаптації змісту, складності, формату і темпу виконання вправ під індивідуальні характеристики студента, такі як попередній рівень знань, стиль навчання, інтереси та навчальні цілі. На відміну від традиційних систем персоналізоване навчання ставить на перше місце результати навчання студента [1]. Завдання не просто змінюються для кожного, а формуються динамічно під час виконання відповідно до прогресу [1, 2]. Ключовим аспектом персоналізації практичних завдань є динамічна адаптація: навчальний матеріал та завдання не є статичними, а безперервно змінюються відповідно до розвитку знань, досвіду та успішності студента. [2, 3]. В українських освітніх документах загальноприйняте визначення поняття «персоналізоване навчання» з'явилося не так давно. В [4] вперше дається тлумачення персоналізації як «здатності створювати навчальне середовище, яке дає змогу студентам реалізувати власні цілі, темп та/або спосіб навчання».

Розвиток сучасних технологій зробив персоналізоване навчання все більш адаптивним, а адаптивне навчання все більш персоналізованим. Відповідно до цієї тенденції, виникає переглянтий метод навчання – **персоналізоване адаптивне навчання** (Personalized Adaptive Learning – PAL)[5]. PAL - це педагогічний підхід, який використовує технології для адаптивного коригування навчальних стратегій на основі моніторингу в реальному часі. Ключовими рисами персоналізованого адаптивного навчання є визнання унікальних рис кожного студента, моніторинг та адаптація на основі попередніх знань та на основі контролю прогресу студента в реальному часі, узгодження навчання з особистим довгостроковим баченням і цілями студентами, динамічна модифікація стратегій і контенту відповідно до

потреб студента. У контексті DLE такі завдання можуть змінювати порядок і темп завдань залежно від результатів попередніх кроків; коригувати складність через алгоритми машинного навчання і динамічного програмування, надавати персоналізовані підказки, зворотний зв'язок чи альтернативні задачі. PAL позитивно впливає на академічну успішність та загальний рівень навчання у вищій освіті, а також на мотивацію: персоналізовані навчальні плани ефективно підтримують залученість студентів [6]. Цей підхід є наступним кроком еволюції традиційних, універсальних методів навчання [7]. PAL вважається парадигмою дослідження освітніх технологій п'ятого покоління. Поштовхом до її розвитку став розвиток технологій великих даних, AI та машинного навчання, які відіграють вирішальну роль у посиленні персоналізованої освіти. Переваги персоналізованих завдань очевидні. Це як сприяння академічній доброчесності, бо унікальні варіанти унеможливають пряме копіювання відповідей між студентами, так і формування глибшого розуміння і більш якісного засвоєння матеріалу: варіативність вхідних параметрів змушує студента дійсно працювати над вирішенням задачі, застосовувати логіку і гнучкість у розв'язанні, адже списування стає неможливим. Згідно [8] понад 70% студентів погодилися, що персоналізовані завдання мотивують їх до самостійної роботи та зменшують рівень плагіату.

В рамках даного дисертаційного дослідження розглядаються також **параметризовані завдання** (в тому числі практичні) – як один із типів персоналізованих завдань, що мають однакову дидактичну мету, але є унікальним для кожного здобувача освіти. Унікальність досягається завдяки різному набору вхідних параметрів або змінним початковим умовам. Такі завдання передбачають унікальне параметризоване завдання для кожного студента, зберігаючи загальну методичну структуру та рівень складності.

Варто зазначити, що терміни "персоналізоване навчання", "адаптивне навчання", "індивідуалізоване навчання" та "кастомізоване навчання" часто використовуються як взаємозамінні (згідно [2]).

1.2 Особливості навчання студентів інженерних спеціальностей

Навчання інженерним спеціальностям фундаментально відрізняється від гуманітарних, суспільних та навіть чистих природничих наук. Якщо наука шукає відповідь на запитання “Чому це так працює?”, то інженерія відповідає на запитання “Як змусити це працювати для вирішення конкретної проблеми?”.

Ця орієнтація на створення працюючого рішення визначає специфіку інженерної освіти, ключові відмінності якої наведено в Таблиці 1.1.

Таблиця 1.1

Ключові відмінності інженерної освіти

Характеристика	Гуманітарні науки	Фундаментальні науки (Фізика, Математика)	Інженерія (ІТ, Мережі, Електроніка)
Кінцева мета	Інтерпретація, смисли, критика	Відкриття істини, законів природи	Створення працюючого артефакту
Тип мислення	Дивергентне (багато правильних відповідей)	Аналітичне, доказове	Синтетичне, проектне
Критерій успіху	Переконливість аргументації	Строгість доведення, відтворюваність	Функціональність в умовах обмежень
Ставлення до помилки	Помилка — це інша точка зору	Помилка спростовує гіпотезу	Помилка (баг) — це етап налагодження

Сформулюємо специфічні особливості інженерного навчання:

1. **Артефакт-орієнтованість.** В інженерії результат завжди матеріальний або логічно-відчутний. Студенту недостатньо просто знати теорію маршрутизації чи принципи роботи Ethernet. Він повинен написати код, налаштувати веб-сервер або підключити мікроконтролер ESP32 до датчика так, щоб система успішно передавала пакети даних. Оцінюється не лише знання, а й кінцевий продукт.

2. **Робота в умовах жорстких обмежень.** Інженерна задача ніколи не існує у вакуумі. Навчання вимагає від студента шукати баланс між суперечливими вимогами, куди можна віднести апаратні ліміти: (оптимізація коду під обмежену

пам'ять мікроконтролера); стандарти та протоколи (необхідність суворо дотримуватися специфікацій Modbus або MQTT); час та ресурси (рішення має бути не просто ідеальним, а таким, що можна реалізувати в задані терміни).

3. **Системне мислення.** Інженерія вимагає розуміння того, як дрібні компоненти взаємодіють у великій системі. Студент, який вивчає веб-програмування, стикається з тим, що фронтенд, бекенд, база даних та мережева інфраструктура тісно пов'язані. Зміна в одному модулі може непередбачувано «зламати» інший. Навчання фокусується на інтерфейсах – точках взаємодії між компонентами.

4. **Ітеративність та налагодження.** У гуманітарних науках есе можна написати з першого разу і просто відредагувати стиль. В інженерії жодна складна система не працює з першого запуску. Процес навчання на 70% складається з пошуку причин, чому система дала збій. Вміння читати логи помилок, використовувати симулятори (наприклад, Wokwi [9]) та інструменти профілювання є критично важливими навичками, які потрібно виховувати цілеспрямовано.

5. **Потреба у формуванні практичних навичок.** Особливо важливим компонентом підготовки інженерів є практичні завдання. У сучасних дослідженнях інженерної освіти **поняття сталих навичок** пов'язується з концепціями transferable skills та sustainability competencies, які передбачають здатність застосовувати знання в різних контекстах, адаптуватися до нових умов та інтегрувати набуті вміння у практичну діяльність. Зокрема, підкреслюється, що такі навички мають довгостроковий характер і є необхідними для ефективної професійної діяльності в умовах змін [10-13]. У даному дослідженні це поняття конкретизується та інтерпретується як сталі практичні навички - сформовані здатності до виконання професійних дій, які зберігаються в часі, відтворюються в різних контекстах та не залежать від зовнішньої допомоги чи конкретного сценарію виконання. На відміну від ситуативного виконання окремого завдання, сталість навички передбачає здатність студента коректно виконувати множину варіацій задачі за різних початкових умов та в різних середовищах виконання. У даному дослідженні поняття

сталих практичних навичок операціоналізується через здатність студента коректно виконувати множину параметризованих варіацій практичного завдання.

Розвиток таких сталих і затребуваних на ринку праці навичок програмування, мережевого адміністрування, аналізу даних тощо і загалом формування експертності в цих галузях ґрунтується на виконанні практичних завдань та набутті досвіду розв'язання реальних інженерних задач. У сучасній інженерній освіті це підтверджується підходом CDIO, який акцентує увагу на формуванні компетентностей через практичну діяльність [14], а також міжнародними рекомендаціями ACM/IEEE, що визначають практико-орієнтоване навчання як ключову складову підготовки ІТ-фахівців [15]. Міжнародні стандарти інженерної освіти, зокрема ABET, акцентують увагу на здатності студентів застосовувати знання для вирішення практичних інженерних задач у реальних умовах [16]. Ряд джерел в галузі комп'ютерної освіти підтверджують, що hands-on підхід значно підвищує ефективність засвоєння матеріалу та формування практичних навичок [17-19]. Мета вищої освіти, закладена у законодавстві України, продовжує цю концепцію. Заклади вищої освіти мають надати ті компетентності, що потрібні для результативної подальшої професійної діяльності в певній спеціальності або галузі знань [20]. Але, на жаль, за даними Міністерства цифрової трансформації України (2020), майже 50% випускників ІТ-спеціальностей опиняються за межами своєї галузі через невідповідність змісту навчальних програм потребам ринку [21]. Саме тому практичні кейси мають бути закладені у програми, і тоді сприятимуть розвитку компетентності студентів та їх працевлаштуванню (згідно дослідження інтеграції професійних стандартів і реальних задач у програми підготовки ІТ-спеціалістів [22]). А отже, в таких галузях як комп'ютерні та інженерні науки, практичні завдання мають бути основними видами навчального процесу.

6. Потреба у формалізації завдань. Через те, що інженерні завдання передбачувані у своїй структурі (вхідні дані -> процес -> результат), їхня постановка вимагає високого рівня формалізації. Потрібна чітка модель практичного завдання: архітектура середовища, початкові стани, критерії валідності та очікувані метрики

продуктивності. Саме потреба у чіткій формалізації та багатокритеріальному оцінюванні робить процес створення якісних інженерних практичних завдань таким складним для викладача.

Крім того, у сучасній інженерній освіті практичне завдання зазвичай має подвійну природу. По-перше, воно вимагає зміни стану, наприклад, модифікації конфігурацій системи, розгортання контейнерів або зміни середовищ виконання. По-друге, воно передбачає створення артефактів, включаючи вихідний код, скрипти, файли конфігурації або дескриптори розгортання. Традиційні описи завдань та моделі оцінювання рідко розрізняють ці виміри, хоча вони є педагогічно та технічно різними.

7. Залежність від інструментів. Сучасне вирішення інженерних задач за своєю суттю є опосередкованим інструментами. Учні покладаються на конкретні інструменти та середовища, такі як компілятори, платформи контейнеризації, хмарні сервіси та часто – інструменти AI для виконання практичних завдань. Існуючі формальні описи завдань рідко визначають або обмежують використовувані інструменти, розглядаючи їх як зовнішні та нерелевантні для оцінювання, незважаючи на їх істотний вплив як на процес навчання, так і на його результати. Існуючі підходи не передбачають механізмів для формального визначення того, як має виконуватися завдання, які інструменти дозволені або необхідні, та як можна перевірити дотримання цих обмежень у складних, недетермінованих середовищах.

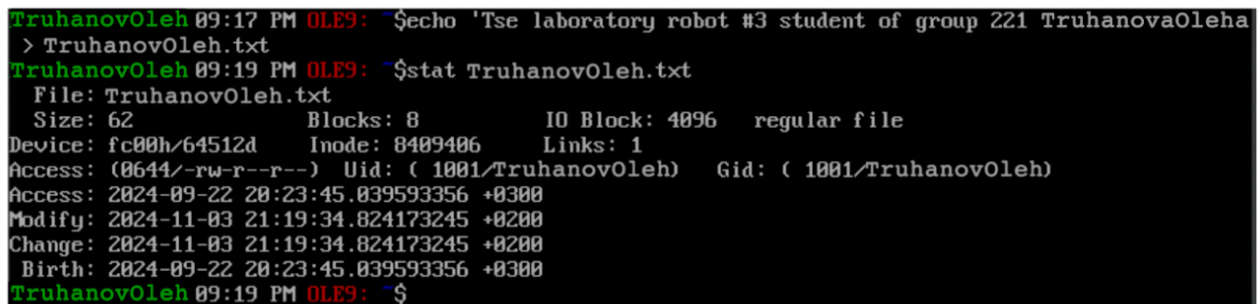
Поточні виклики практичного навчання. В дистанційному навчанні відслідкувати виконання практичного завдання конкретним студентом видається майже неможливим. Ця проблема існувала і в офлайн навчанні, але не була настільки масовою: навіть отримуючи завдання однакове для всієї групи (не персоналізоване), більшість студентів виконували практичні роботи в аудиторії під наглядом викладача, а в разі самотійного виконання практичного завдання вдома студенти презентували і захищали роботу очно в класі. Наразі, коли навчання відбувається дистанційно, а тим паче – асинхронно, додатковий виклик, з яким стикаються викладачі, це академічна недоброчесність, адже дистанційний формат

послаблює контроль за самостійним виконанням завдань. І під час онлайн-оцінювання питання шахрайства постає більш гостро: прослідкувати і впевнитись в тому, що студент самостійно виконав практичне завдання (а отже і здобув практичний навичок) важко.

Просте чи пряме перенесення не персоналізованих завдань в дистанційну форму навчання дає змогу студентам подавати звіти без виконання самих завдань, а підтвердження персонального виконання завдання студентом потребує додаткових зусиль як студента, так і викладача. Наприклад, запис студентом процесу виконання на відео є додатковим і не профільним навантаженням, потребує більше часу на виконання, підвищує рівень стресу і потребує додаткового програмного забезпечення. З боку викладача – збільшується часові витрати на перевірку завдань, а також з'являється критичне навантаження в певні проміжки часу перед сесією.

Практика персоналізації завдань за допомогою видачі унікального варіанту в рази краще за не персоналізовані завдання. В ідеальному випадку являє собою набір варіантів, де кількість варіантів дорівнює максимальній кількості студентів в групі (потоці), коли кожен студент отримує схоже, але персоналізоване завдання, і має представити окрему роботу, і просте копіювання готових відповідей у колеги по навчанню вже не може бути для студента рішенням. І частково цей підхід виправдовував себе. Маємо конкретні приклади та кейс-стаді з реального життя, де використання персоналізованих завдань принесло позитивні результати. В 2022/2023 навчальному році проведено дослідження персоналізації завдань по дисципліні “Операційні системи Unix” (кафедра ІКС НУ “Чернігівська Політехніка, викладач - Хижняк А.В.), а саме: для підвищення контролю за самостійним виконанням завдань в завдання лабораторних робіт додано персоналізацію на основі ПІБ студента. Студент мав завантажити в систему LMS скріншоти виконаних практичних робіт, де мало бути чітко відображений користувач системи. Це – гарна заміна статичним варіантам. Просте запозичення скріншотів у колег по навчанню в такому випадку вже не спрацьовує. Відмічались лише поодинокі випадки виконання роботи в цілому іншим студентом і кілька спроб підробки скріншотів за допомогою Photoshop.

Приклад такої підробки - на рисунку 1.1.



```
TruhanovOleh 09:17 PM OLE9: ~$echo 'Tse laboratory robot #3 student of group 221 TruhanovaOleha
> TruhanovOleh.txt
TruhanovOleh 09:19 PM OLE9: ~$stat TruhanovOleh.txt
  File: TruhanovOleh.txt
  Size: 62          Blocks: 8          IO Block: 4096   regular file
Device: fc00h/64512d Inode: 8409406    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/TruhanovOleh)   Gid: ( 1001/TruhanovOleh)
Access: 2024-09-22 20:23:45.039593356 +0300
Modify: 2024-11-03 21:19:34.824173245 +0200
Change: 2024-11-03 21:19:34.824173245 +0200
Birth: 2024-09-22 20:23:45.039593356 +0300
TruhanovOleh 09:19 PM OLE9: ~$
```

Рис.1.1 Підробка скріншоту виконаного завдання за допомогою Photoshop.

Це лише виявлені випадки, тоді як реальні масштаби проблеми, ймовірно, є значно більшими. 60 % студентів зізнаються у регулярному списуванні під час онлайн-іспитів[23]. Серед основних його форм виокремлюють звернення за допомогою до третіх осіб під час онлайн-іспитів, передачу облікових даних (логін і пароль для авторизації в системі) для проходження оцінювання іншими особами, а також використання AI для генерації відповідей [24, 25]. І таке академічне шахрайство в умовах онлайн-освіти є складнішим для виявлення [26]. Емпіричні дані автора підтверджують зазначені тенденції. Кількість індивідуальних завдань і їх “індивідуальність” не важлива, бо за допомогою AI студенти генерують відповіді, схеми, діаграми, і навіть - скріншоти виконаних лабораторних робіт. В наступному навчальному році встановлено, що 6,7 % студентів використовували AI для фальсифікації відповідей, а 16,7 % – графічні редактори. Розподіл робіт за якістю виконання завдань по дисципліні “Операційні системи Unix” у 2023/2024 навчальному році (кафедра ІКС НУ “Чернігівська Політехніка, викладач - Хижняк А.В.), наведений на рисунку 1.2.

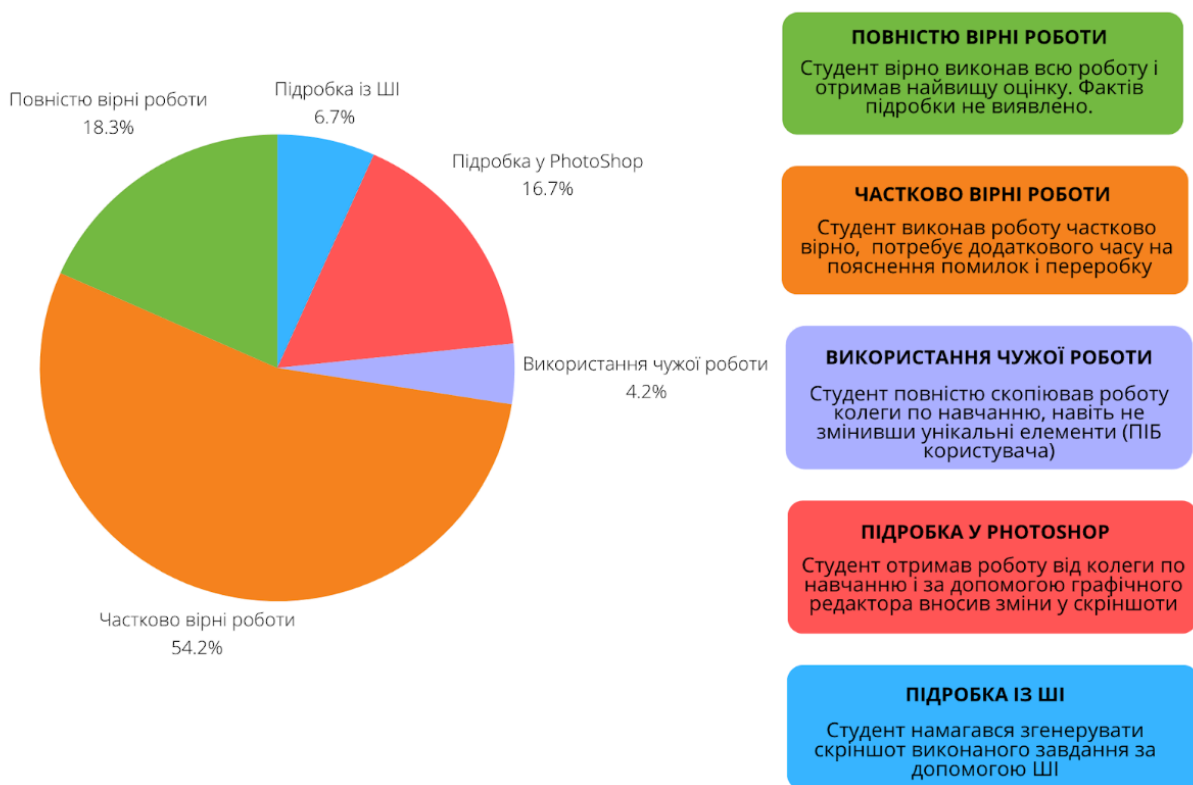


Рис. 1.2. Розподіл робіт за якістю виконання практичних завдань

Шлях до зниження рівня академічного шахрайства та набуття студентами сталих практичних навичок полягає у впровадженні різних автоматизованих форматів персоналізованого практичного навчання. Дослідження [27] підтверджує, що автоматизовані інструменти підвищують мотивацію студентів, покращують якість їхніх робіт та розвивають практичні навички програмування. В [28] автори прямо пов'язують автоматизовану оцінку з академічною доброчесністю. Це джерело чудово підтверджує запропонований в даній дисертації підхід, оскільки воно поєднує автоматизацію, масштабованість та механізми пераметризації для боротьби з академічною недоброчесністю.

Неавтоматизовані процеси вимагають значних часових ресурсів з боку викладача. Для розрахунку середньостатистичних часових витрат викладача на створення унікальних завдань в рамках курсу “Операційні системи Unix” були використані дані по кількості студентів спеціальності 123 за останні 5 років

(дисципліна викладається студентам на 3-му курсі). Дані в Таблиці 1.2 отримані від деканату ННІ ЕІТ НУ “Чернігівська Політехніка”.

Таблиця 1.2

Кількість студентів за 2021-2025рр.

Рік	Кількість студентів на 3 курсі
2024-2025	95
2023-2024	105
2022-2023	131
2021-2022	96

В середньому в потоці маємо 105 студентів. В курсі “Операційні системи Unix” 6 лабораторних робіт. Для підготовки 105 унікальних варіантів завдань для кожної з них, включаючи заповнення завдання в LMS та його подальшу ручну перевірку, потрібно не менше 15 хвилин. Отже, на підготовку та перевірку виконаних практичних завдань для однієї дисципліни викладач має витратити $105 \times 15 \times 6 = 9540$ хв (157 годин), що в 6 разів перевищує обсяг годин, виділених на цей вид робіт і займає 75% всіх виділених на дисципліну годин (за навантаженням викладача на дисципліну виділяється 207 годин, з них лише 25,6 годин - безпосередньо на перевірку лабораторних робіт).

З метою проведення аналізу поточного стану створення і видачі персональних (варіативних або параметризованих) завдань в НУ “Чернігівська Політехніка” проведено опитування викладачів інженерних кафедр. Абсолютна більшість викладачів вважають доцільним використання індивідуальних (варіативних або параметризованих) практичних завдань і намагаються використовувати їх в своїй викладацькій роботі, хоч і в обмеженому вигляді. Викладачі, які не використовують індивідуальні завдання, відмічають, що хотіли б застосовувати їх в освітній діяльності. Серед причин НЕ використання таких завдань в першу чергу називають великі часові витрати, які необхідні для підготовки та перевірки таких завдань, а також важкість процесу ручної генерації завдань. Отже, великий мінус впровадження персоналізованих завдань – це часові та ресурсні витрати, і

масштабування процесів вимагає впровадження автоматизованих інструментів, що є актуальним завданням сьогодення.

Співвідношення практичних, інженерних та персоналізованих завдань. У даному дослідженні розглядається обмежений клас навчальних завдань, а саме практичні завдання інженерного спрямування, що обумовлено специфікою підготовки фахівців у галузі ІТ та інженерії, де ключову роль відіграє формування практичних навичок. У зв'язку з цим інші типи навчальних завдань (теоретичні, тестові тощо) не розглядаються в межах даної роботи. Також акцент зроблений саме на персоналізованих завданнях, як найбільш перспективних для підвищення ефективності навчального процесу та формування сталих практичних навичок. Співвідношення розглянутих класів завдань наведено на рисунку 1.3.



Рис.1.3. Співвідношення практичних, інженерних та персоналізованих завдань

З урахуванням зазначеного, базовою одиницею аналізу є персоналізоване практичне інженерне завдання (Personalized Practical Engineering Task – PPET), яке охоплює як формальний опис структури завдання, так і його конкретні параметризовані реалізації, що застосовуються у навчальному процесі. На відміну від підходів, що розрізняють узагальнену модель та її конкретні реалізації, у даній роботі модель та її інстанції об'єднані в межах єдиного поняття PPET.

При цьому розмежування між ними здійснюється на рівні інтерпретації:

модель РРЕТ – формальний опис структури завдання, що містить множину параметрів, обмеження, середовище виконання та критерії оцінювання;

екземпляр РРЕТ – конкретна реалізація завдання, отримана шляхом параметризації моделі для певного студента з урахуванням його індивідуальних характеристик.

1.3 Аналіз методів персоналізації, їх можливостей та обмежень

У сфері персоналізованого навчання застосовуються різні методи і методики. Важливо розрізняти методи персоналізованого навчання (педагогічні/дидактичні підходи, які описують як має бути організований процес навчання, щоб він був індивідуально орієнтованим і пристосованим до потреб студента [29]) та технологічні та аналітичні методи, які застосовуються для цілей персоналізації навчання і дозволяють реалізувати або підтримати її у конкретному середовищі. Останні орієнтовані на обробку, аналіз або генерацію даних, є інструментами в межах персоналізованих стратегій та потребують інтеграції в освітній сценарій. В поєднанні з теоріями персоналізації (Mastery Learning [30, 31], SelfDetermination Theory [32], Metacognitive Theory [33], ModelCentered Instruction [34], Activity Theory / CulturalHistorical Activity Theory [35, 36], Challengepoint framework [37]) ці методи дають змогу побудувати ефективні персоналізовані середовища для виконання і перевірки РРЕТ.

Підходи до персоналізації практичних завдань у цифровій освіті дуже різняться, від базових форм індивідуального доступу до комплексних адаптивних стратегій. Відповідно до класифікації персоналізації практичних завдань [38] проведено аналіз освітніх платформ, порівняння їхніх можливостей і виділення обмежень. Їх опис і приклади застосування наведені в Таблиці 1.3

Таблиця 1.3

Опис і приклади персоналізації практичних завдань

Ном ер рівн я	Назва	Опис	Приклади застосування
L0	Без персоналізації	Студенти отримують однакове завдання без жодних варіацій. Відсутнє врахування індивідуальних особливостей, контексту або навіть індивідуального кабінету.	Електронні або надруковані однакові завдання, доставлені студенту без можливостей LMS.
L1	Персоналізація доступу	Формування індивідуального навчального простору (персонального кабінету), де зберігається особиста інформація студента, оцінки, завдання, прогрес, результати.	Профіль у LMS. Стандартні завдання в рамках LMS [39-41].
L2	Варіативна персоналізація (variant-based personalization)	Кожен студент отримує унікальний варіант завдання з фіксованого переліку заготовок. Зміни не суттєво впливають на методику розв'язання, проте частково знижують списування.	Рандомізована видача варіанту завдання можливостями LMS [39,40,42,43].
L3	Параметризована персоналізація (parameterized personalization)	Завдання однакові по структурі і за рівнем складності, але генеруються автоматично на основі змінних параметрів. Підходить для масового індивідуального оцінювання, орієнтованого на зниження списування, зменшення часових затрат викладача.	Заміна числових значень, імен, портів у мережах, структури масивів, використання індивідуальних VLE для виконання завдань [44-48].
L4	Персоналізація за рівнем знань	Завдання підбираються автоматично або викладачем відповідно до рівня знань студента (на основі тестування або попередніх результатів) та з урахуванням прогресу. Складність завдання змінюється динамічно, що підтримує зону найближчого розвитку.	Легка задача → успішне виконання → складніша наступна [42(частково), 49-52].
L5	Інтерактивна персоналізація	Виконання завдань в інтерактивному середовищі з миттєвим зворотним зв'язком і динамічною корекцією. Завдання адаптуються на основі реакції в реальному часі.	Веб-платформи, інтерактивні IDE, системи миттєвого фідбеку, AI-асистент [41, 51, 53-58].

L6	Персоналізація моделей учня	Використання індивідуальних характеристик для глибшої персоналізації. Це стиль навчання, швидкість виконання, типові помилки, сильні і слабкі сторони, зібрані через поведінкові або аналітичні моделі. Такий індивідуальний підхід підвищує конвертацію контенту із завдання в знання студента.	ML-моделі, нейроадаптивні системи, що підбирають формат/формулювання (графічний, код, текст), кількість повторів завдань, тощо [51, 54, 55,59-62].
L7	Мотиваційно-гейміфікована персоналізація	Використання ігрових механік для підвищення мотивації і персоналізації процесу навчання.	Використання балів, рейтингів, рівней, бонусів [56, 63, 64].
L8	Контекстна персоналізація	Завдання адаптуються до особистих інтересів, спеціалізації або професійного контексту студента.	GPT-генерація, knowledge-graph диспетчери, AI-агенти, профільні симулятори [52 65, 66(частково), 67, 68].
L9	Особистісно-когнітивна	Адаптація завдань на основі стилю навчання, темпераменту, емоційного стану, когнітивних характеристик або фізіологічних даних.	Врахування когнітивних і психологічних особливостей студента, використання сенсорних даних (пульс, час реакції) для зміни типу завдань [69].

Пояснення щодо розмежування видів персоналізації, обґрунтування критеріїв та верифікація запропонованого порядку наведено в [38]. Детальний аналіз існуючих методів персоналізації представлений автором в науковій публікації [70], зупинимось детальніше лише на тих, які знайшли безпосереднє удосконалення в рамках дослідження. Методи автоматизованої генерації та перевірки завдань ефективніше працюють в комплексі і є сферою наукових інтересів автора дослідження з акцентом на практичні завдання для інженерних дисциплін. Велика кількість наукових праць присвячена розробці саме цих методів та впровадженню інструментів для автоматизованої генерації та перевірки завдань, що підтверджує актуальність теми дослідження.

Методи автоматичної генерації завдань [53, 68, 71-73] передбачають створення навчальних завдань або тестових питань за алгоритмічно заданими правилами або з використанням AI без ручного втручання викладача, враховувати рівень складності, темп навчання та прогалини у знаннях студента, підвищувати

варіативність і унікальність завдань (важливо в масовому онлайн-навчанні та при тестуванні). Вони є ключовим механізмом персоналізації навчання у DLE, дозволяючи адаптувати зміст, складність і тип завдань до індивідуальних потреб студента. Методи автоматичної генерації завдань можуть реалізовуватись на основі шаблонів і правил (статичні шаблони з параметризацією), Natural Language Processing (генерація питань на основі тексту з використанням трансформерів або seq2seq моделей), DSL (використання предметно-орієнтованих мов для формалізованого опису типових завдань, які автоматично компілюються в індивідуальні варіанти), динамічного формування завдань залежно від результатів студента (рівень успішності, допущені помилки).

Окремо і ширше розглянемо **можливості штучного інтелекту для генерації завдань**. Використання генеративного штучного інтелекту (Generative Artificial Intelligence, GenAI) в освітньому процесі забезпечує комплексні переваги за двома основними напрямками. Для студентів AI-інструменти дозволяють реалізувати персоналізацію через адаптацію змісту, складності та темпу навчання до індивідуальних потреб, що сприяє глибшому засвоєнню знань та розвитку критичного мислення. Використання AI дозволяє створювати диференційовані навчальні сценарії та адаптувати освітній процес до потреб кожного студента, що є ключовою перевагою сучасних освітніх систем [74]. Для викладачів GenAI виступає засобом автоматизації рутинних процесів, зокрема створення, адаптації та оцінювання практичних завдань [75]. GenAI відкриває значний потенціал саме в рамках автоматизації процесів персоналізації, зокрема у сфері створення, адаптації та оцінювання практичних завдань. Одним із перспективних напрямків є автоматична генерація тестових наборів за допомогою LLM, що показує себе не гірше за традиційні підходи, а іноді й виявляє неоднозначності у завданнях [71, 76-78]. В [68,72] використовується GenAI для автоматичного створення програмувальних завдань, включаючи опис задачі, каркас коду, юніт-тести та модельні рішення. Оцінка якості завдань показала високу функціональність та розв'язуваність, а також високу задоволеність студентів персоналізацією завдань.

GenAI також може адаптувати рівень складності завдання у реальному часі, змінюючи параметри на основі аналізу відповіді студента. Наприклад, якщо студент невдало виконав певний блок, система може дати спрощену задачу або пояснення, а потім повернутися до більш складного рівня. Можлива також диференційована генерація [71], зміна кількості підпитань, обсягу коду, контексту, що є важливим в аспекті персоналізації – система не дає занадто легких або занадто складних варіантів.

Іншим напрямом персоналізації навчання за допомогою AI є генерація пояснень, підказок та адаптивних коментарів у відповідь на студентські рішення. Це підвищує залученість студентів, однак може створювати додаткове когнітивне навантаження або вводити в оману через можливу неточність рекомендацій. Взаємодія студентів зі зворотним зв'язком від AI має активний характер: вони аналізують підказки, співвідносять їх із власними знаннями та приймають рішення щодо їх використання. Загалом процес взаємодії відбувається у циклі «отримання → оцінка → застосування або уточнення», що підкреслює активну роль студента у використанні такого зворотного зв'язку.

В [68] досліджується використання GenAI (зокрема GPT-4) для створення персоналізованих програмувальних завдань у вступному курсі програмування. У роботах [73, 79] автори також досліджують можливості використання Chat GPT для автоматичного створення програмувальних завдань. Проте і тут ефективність створених завдань безпосередньо залежить від якості GPT-4, яка може варіюватися, а автоматично згенеровані завдання можуть містити помилки або бути занадто складними, що вимагає додаткового контролю з боку викладачів.

В [80] автори пропонують новий тип програмувальних завдань (Prompt Problems), що вимагають від студентів формулювання чітких, точних та ефективних запитів до LLM. Запропонований підхід дозволяє не лише покращити їхні технічні навички, але й розвинути критичне мислення та здатність до самостійного навчання, проте має кілька обмежень: студенти можуть демонструвати суттєво різні результати, бо успіх значною мірою залежить від досвіду роботи з LLM (здатності

правильно формулювати запити), що створює ризик нерівного навчального досвіду.

Проте основною проблемою є контроль якості таких генерацій – часто LLM створює тести, які вимагають валідації. В [76] автори демонструють, що LLM можуть формулювати питання або тести, які покривають аспекти, що випадають із “класичного” набору тестів, включаючи нестандартні випадки. Використання GenAI у вільному режимі призводить до генерації некоректних, надто складних завдань або генерується тест, який не відповідає рамкам специфікації (edge case), який може бути хибно інтерпретований системою – і такий тест необхідно фільтрувати. І, незважаючи на очевидні переваги LLM для автоматичної генерації завдань, рішення мають низку недоліків. По-перше, моделі обмежені у роботі з нестандартними або складними алгоритмічними завданнями і можуть створювати неповні або хибні тестові випадки. По-друге, тести, згенеровані моделлю, можуть бути надмірно простими або, навпаки, надто складними для початкового рівня, що вимагає додаткового контролю з боку викладача. Крім того, LLM не здатні повністю замінити експертну перевірку, оскільки якість і повнота тестів потребують верифікації людиною, що зменшує частину автоматизації. Нарешті, будь-які помилки у описі завдання або еталонному рішенні, можуть бути відтворені моделлю у тестах, що підвищує ризик неправильної оцінки студентських робіт.

Тому для використання подібних генерацій в масштабі LMS важливо контролювати генерацію тестів через валідаційні модулі – наприклад, перевірку проти референсного рішення або формальних правил та динамічно оцінювати продуктивність (затримка, обсяг генерації, навантаження на серверну інфраструктуру навчального закладу). GenAI дає перспективу суттєвої оптимізації процесів персоналізації, але потребує обережного інтегрування з механізмами контролю, валідації й адаптивності. Scoping Review 2024 року [81] підтверджує вказані твердження: "обіцянку персоналізованого навчання та інтелектуальних тьюторських систем", але і прямо вказує на "занепокоєння щодо інструментів, таких як ChatGPT, що продукують нісенітницю (nonsense)". Це обґрунтовує необхідність використання формальної мови як механізму валідації та контролю для

"галюцинацій" AI.

Методи автоматизованої перевірки завдань [44-47, 82] (*automated assessment methods*) – це сукупність методів, спрямованих на автоматизовану оцінку навчальних завдань (без прямої участі викладача) з метою забезпечення швидкого та об'єктивного фідбеку, оновлення моделі знань студента, реалізації адаптивного навчального середовища, є необхідною умовою масштабованої персоналізації без ручного втручання викладача і основою для динамічної адаптації змісту, складності та типу завдань відповідно до індивідуального профілю учня. Методи автоматизованої перевірки включають [83] методи порівняння із заданим еталоном, Unit-тестування, Code-Based Evaluation, NLP-based Grading для оцінки відкритих тестів, перевірку за допомогою моделей навчання тощо. Цікавими є і гібрид-підходи, які поєднують декілька методів. Наприклад, в дослідженні [82] представлено гібридний підхід до автоматичного оцінювання, що поєднує NLP-можливості та алгоритм Random Forest Regression, спрямовані на безперервну оцінку якості тексту. Наведемо кілька існуючих методів.

Фреймворк **SERA** [44] призначений для автоматизованого створення та оцінювання лабораторних вправ у курсах інформаційної безпеки, використовує шаблони вправ (*activity templates*), на основі яких генеруються індивідуальні варіанти завдань для студентів. Опис структури вправи здійснюється у вигляді параметризованих JSON-шаблонів, що дозволяє автоматизувати генерацію варіантів завдань та їх перевірку. Разом з тим даний підхід має ряд обмежень: інтеграція з LMS системою є обмеженою і використовується переважно для отримання списку студентів; результати виконання завдань представляються у вигляді файлів, структура яких перевіряється окремими модулями, що обмежує можливість перевірки станів реального навчального середовища; система не забезпечує повної інтеграції процесів генерації, виконання та автоматичної перевірки практичних завдань у єдиному середовищі.

APG automatic problem generation [45]. Метод APG реалізується через Python-скрипти, які параметризують конфігураційні файли і дозволяють кожному

студенту запустити свою унікальну віртуальну машину (Virtual Machine - VM) для виконання лабораторної роботи в рамках своєї локальної ОС. Stand alone server очікує на відповідь студента, яку останній отримує, виконавши завдання на своїй локальній VM. Кожна відповідь буде унікальною за рахунок параметризації навчального середовища. Автори відслідковують підробки через розбиття завдання на етапи виконання і відслідковування послідовності завантажень відповідей, і також через завантаження вірних відповідей інших студентів (які не є вірними для тих, хто завантажив), і через IP адресу. Серед недоліків слід відзначити локальність розгортання VLE, що дозволяє студентам шляхом реверс інжинірингу отримати відповіді, не виконуючи безпосередньо завдання; не вистачає можливості автоматичної перевірки виконання завдання (перевірка напівавтоматична, треба вносити відповіді) та інтеграції з LMS для отримання завдання та збору результатів.

PAGE [46] генерує індивідуальні варіанти тестів для кожного студента шляхом варіювання параметрів питань і відповідей, інтегрується з LMS Canva через API, що дозволяє використовувати її без розгортання додаткових компонентів інфраструктури. Практичне застосування системи показало ефективність автоматичної генерації персоналізованих завдань, особливо у масових онлайн-курсах, де складно забезпечити індивідуальний контроль навчального процесу. Водночас підхід базується переважно на випадковому виборі елементів із великого підготовленого дата-сету, а не на повноцінній генерації завдань, що обмежує гнучкість персоналізації; використання PAGE потребує від викладачів знань програмування, зокрема мови JavaScript для генерації питань і відповідей; відмічається значне навантаження на LMS через необхідність зберігання великої кількості унікальних тестів.

MCTest [47] реалізує метод параметризованого автоматизованого оцінювання, використовує параметризовані шаблони завдань та автоматичну перевірку рішень у курсах програмування (*programming exercise*). Метод передбачає генерацію шаблонів завдань із використанням параметрів за допомогою LaTeX, Python та C++, інтеграцію з Moodle і плагіном VPL для виконання і перевірки коду безпосередньо у

браузері. Перевагою підходу є швидкий зворотний зв'язок без участі викладача, що, за результатами дослідження, позитивно впливає на середні результати навчання. Разом з тим система має обмеження, зокрема необхідність вказувати версію завдання після завантаження рішення.

Попри наявні успішні приклади впровадження подібних методів в DLE, залишаються наступні виклики: технічна складність впровадження; обмежена автоматизація перевірки; неповна інтеграція з LMS; ризики компрометації завдань при локальному виконанні; взаємодія тільки із артефактами, у вигляді файлів, яка не враховує стан самого VLE. Ці аспекти підкреслюють потребу в удосконаленні існуючих методів, а також створенні на їх базі гнучких, масштабованих і простих у використанні рішень для підтримки персоналізованого навчання в DLE. З метою обґрунтування необхідності удосконалення існуючих методів визначені і сформульовані їх основні недоліки та перспективні напрямки досліджень, які наведені в Таблиці 1.4.

Таблиця 1.4

Обмеження існуючих методів

Обмеження	Опис
Обмежений функціонал автоматичної генерації	Відбувається випадковий вибір конкретного завдання з існуючою множини замість генерації
Обмежена предметна область	Не всі рішення підходять для параметризації і перевірки саме практичних завдань, а тільки, наприклад, для текстових.
Високий поріг входу	Викладач повинен володіти не лише предметною областю, а й технічними навичками роботи з шаблонізаторами, скриптами генерації, програмування скриптованими мовами.
Інтеграційні труднощі	Важка інтеграція з локальними LMS, брак інструментів, які можуть ефективно інтегрувати персоналізоване оцінювання та додаткові матеріали, такі як великі набори даних.
Локальне розгортання навчального середовища	Реверс інжиніринг дозволяє отримати відповіді, не виконуючи безпосередньо завдання.
Часто відсутність повної автоматичної перевірки завдань	Викладачі та студенти, наприклад, вносять вірні відповіді в систему для перевірки.
Валідація стану середовища	Системи працюють тільки із артефактами, наприклад, у вигляді файлів, і не враховують стан самого середовища.

Візуальне порівняння можливостей вказаних методів та систем, що реалізують ці методи, представлено в Таблиці 1.5.

Таблиця 1.5

Порівняння існуючих методів персоналізації завдань.

Рішення	Практичні завдання	Індив. навч.середовище	Автоматич.розгортання	Автоматизована перевірка	Мова опису	Інтеграція з LMS	Гейміфікація	Масштабованість	Прості шаблони
SERA [44]	-	-	-	+	+	+-	-	+	-
APG[45]	+	+-	+-	-	-	-	+	+-	+
PAGE[46]	-	-	-	+	-	+	-	-	-
MCTest [47]	+-	-	-	+	-	+	-	-	+-

1.4 Вимоги до VLE

Теоретичне дослідження проблеми та власні дослідження автора дали змогу сформулювати вимоги до VLE. Засоби реалізації для кожної із вимог наведені в додатку Г. Вимоги не є довільними, а є синтезом ключових проблем, які обговорюються в науковій літературі, присвяченій практичним віртуальним лабораторіям для ІТ-освіти. Серед головних вимог варто відмітити наступні:

Віддалене розгортання (окремий сервер, хмарна платформа). Допоможе уникнути реверс інжинірингу з боку студентів з метою отримання відповідей без виконання завдання, а також уникнути локальних проблем на машинах студентів і проблем залежностей у ПЗ (*dependency hell*). Якщо студент розгортає VLE на своєму локальному комп'ютері, він отримує повний контроль над ним (може дослідити файли образу, знайти приховані відповіді, скрипти перевірки) або просто скопіювати всю налаштовану VM і передати її іншому студенту. Локальне розгортання створює і величезні проблеми підтримки (наприклад, "у мене не працює VirtualBox"). Централізоване, серверне розгортання вирішує цю проблему [84-86]. Система

CvLabs [87] розроблена для "масштабування для підтримки великої кількості учнів" і використовує "хмарну інфраструктуру". Розробники кібер-полігону KYPO [88] прямо стверджують, що реалістичні вправи з кібербезпеки неможливо безпечно проводити на локальних машинах. Використання клієнт-серверної моделі, де система виступає як сервер (backend), а студент – як клієнт з тонким клієнтом (браузер, термінал) означає, що студент має лише обмежений доступ (наприклад, через SSH, веб-інтерфейс) до екземпляра середовища, але не до його "шаблону". Така практика є основою для концепції Lab-as-a-Service (LaaS).

Інструменти реалізації – це розгортання VLE у хмарних провайдерах (AWS, Azure, GCP) або на локальному сервері університету з використанням гіпервізорів (VMware) чи систем оркестрації (Kubernetes, Vagrant). Студенту видаються лише облікові дані для доступу (IP, логін, пароль).

Середовища мають бути реалістичними. Саме реалістичні VLEs відіграють значну роль у заповненні прогалин у навичках фахівців з програмування, мережевого та системного адміністрування, кібербезпеки. Середовище не повинно бути спрощеною "пісочницею", в якому тестується лише одна команда. Воно має імітувати реальні робочі умови та інженерні задачі, структурні та соціальні складності реального світу. Наприклад, завдання з комп'ютерних мереж має включати кілька віртуальних вузлів (клієнт, сервер, маршрутизатор), які з'єднані між собою, а не просто один комп'ютер.

Досягається за допомогою засобів оркестрації. Docker Compose та Vagrant дозволяють зв'язати кілька VM або імітувати архітектуру мікросервісів у єдиному ізольованому середовищі.

Середовища мають бути автентичними. Середовище має відображати ті неідеальні умови, з якими стикаються інженери. Воно не повинно бути "стерильним". Наприклад, в рамках завдання "створити директорію у /var/www/", студент може зіткнутися з тим, що у нього немає прав на запис у /var/www/, і частиною завдання є діагностика цієї проблеми та її вирішення. V.Potkonjak [89] у своєму огляді підкреслює, що VLE використовуються для підтримки

"конструктивістської... освіти... у складних інженерних темах". Це прямо підтверджує ідею, що проста задача "створи папку" має бути ускладнена реалістичними проблемами (наприклад, правами доступу), щоб відповідати "складній" темі. VLE, розроблені на основі конструктивістських теорій (де студент "відкриває" знання сам, а не просто слухає), показують вищу ефективність [90]. Ідея про те, що проблема з правами доступу є частиною завдання, а не помилкою, прямо підтримується в літературі з проблемно-орієнтованого навчання (*problem-based learning*). В [91] підтверджується, що "стерильні" лабораторні роботи, де все працює з першої спроби, не розвивають навичок діагностики та вирішення проблем (troubleshooting), які є важливими для ІТ-фахівців.

Саме тому частиною модулю розгортання є скрипти налаштування (provisioning instructions), які використовуються не лише для створення потрібних файлів, але й для встановлення реалістичних обмежень. Наприклад, скрипт може: встановити некоректні права доступу, запустити процес, який "займає" потрібний порт, встановити неповні або застарілі пакети ПЗ, налаштувати правила брандмауера, які блокують потрібний трафік, доки студент їх не виправить.

Середовища мають бути обфусковані. В наданих студенту середовищах повинні бути відсутні "цифрові сліди" розгортання (артефакти у вигляді файлів, скрипти ініціалізації в тимчасових папках, команди в історії, тощо). Обфускація (obfuscation) – це процес приховування слідів автоматичного налаштування VLE. Студент повинен аналізувати "докази" (логи або файли), а не сліди скрипта. Дана вимога тісно пов'язана з концепцією відтворюваності (*Reproducibility*) – кожен студент має починати з ідентичного базового образу (*Golden Image*), який не містить підказок в історії команд.

Середовище повинно бути ізольованим. VLE одного студента не повинне мати жодного доступу (мережевого чи на рівні файлової системи) до VLE іншого студента. Це критично для безпеки та академічної доброчесності. Досягається через контейнеризацію (Docker, LXC). Це їхня вбудована функція. Ядро Linux ізолює файлові системи (через OverlayFS), процеси (через Namespaces) та мережу (через

virtual bridges) для кожного контейнера, через використання VMs, які забезпечують повну (апаратну) ізоляцію, оскільки кожна VM має власне віртуальне "залізо" і ядро ОС та через мережеві політики. Використання віртуальних локальних мереж (Virtual Local Area Network, VLAN) на рівні комутаторів або мережевих політик у Kubernetes, щоб трафік одного VLE не міг досягти іншого.

Середовище повинно бути безпечним. Ця вимога має два аспекти: захист студента (VLE не повинно містити шкідливого ПЗ) та захист інфраструктури навчального закладу (студент не повинен мати змоги "втекти" з VLE і отримати доступ до хост-сервера, на якому воно запущене).

Досягається завдяки запуску без прав суперкористувача (запуск процесів усередині контейнерів від імені користувача з низькими привілеями); механізмами безпеки ядра Linux (використання профілів Seccomp та AppArmor/SELinux для жорсткого обмеження системних викликів, які може робити контейнер); використанню VMs (апаратна віртуалізація робить "втечу" з VM надзвичайно складною задачею).

Середовище повинно бути контрольованим. Студент не повинен мати змоги використовувати надані йому (безкоштовні) ресурси для зловмисної діяльності: DDoS-атак, розсилки спаму, брутфорсу або майнінгу криптовалют. Досягається завдяки обмеженню ресурсів та обмеженню мережі. Контейнери (Docker, K8s) та VM дозволяють жорстко обмежити для кожного VLE відсоток CPU та обсяг RAM. Мережеві політики хост-сервера необхідно налаштувати так, щоб заборонити VLE будь-який вихідний трафік в Інтернет, окрім чітко дозволеного списку (наприклад, доступ до репозиторію, до сайту із завданням).

Вимога контрольованості (запобігання майнінгу, DDoS) є ключовою для платформ, що працюють у хмарі, де зловживання ресурсами призводить до прямих фінансових втрат. В дослідженні [92] описують мережеві політики та квоти (*resource quotas*) як обов'язкові компоненти.

Середовище повинно бути інтероперабельним. Це здатність VLE інтегруватися з іншими університетськими системами (наприклад, LMS). Ця вимога

є однією з найважливіших для того, щоб VLE стала реальною частиною навчальної екосистеми закладу, а не просто "окремим сайтом", куди треба заново реєструвати студентів та вручну переносити оцінки.

Досягти цієї інтеграції (здатності "спілкуватися" з іншими системами) можна за допомогою різних засобів. Наприклад, стандартизований освітній протокол (зокрема LTI) дозволяє "вбудовувати" один освітній інструмент всередину іншого (VLE в LMS), підтримує безшовну аутентифікацію та автоматичну передачу оцінок: коли студент виконує практичне завдання у VLE, система автоматично надсилає оцінку назад у Moodle. Також використовуються стандартизовані протоколи автентифікації (Single Sign-On) та власні API, вебхуки та пакетна синхронізація.

Середовище повинно бути кросплатформним. VLE не має бути жорстко прив'язане до певного провайдера віртуалізації (наприклад, тільки до VirtualBox) і підтримувати використання різних бекендів: VM, LXC контейнерів, хмарних сервісів, JupyterLab, Docker, Kubernetes тощо. Це дозволяє уникнути "прив'язки до постачальника" (*vendor lock-in*) та адаптувати лабораторію до наявних в університеті ресурсів. Потрібна гнучкість, аби за необхідності додати підтримку нового бекенду.

В даній роботі це досягається завдяки абстракції та модульній архітектурі. Для кожного бекенду (Docker, VirtualBox, K8s) створюється окремий драйвер, який знає, як транслювати абстрактний опис завдання у конкретні команди API цього бекенду.

Середовище повинно бути продуктивним і надійним. Система має бути доступною 24/7, швидко завантажуватися та витримувати пікові навантаження (наприклад, під час екзаменів), що особливо актуально в умовах сучасного стану енергосистеми України. Головний принцип – уникнення єдиної точки відмови (Single Point of Failure, SPOF).

Засоби реалізації: резервування, кластеризація, Load Balancing, моніторинг навантаження, реплікація баз даних та оптимізація запитів до неї, оркестрація та контейнеризація, Image Baking та Golden Image.

1.5 Засоби формалізації в освітніх технологіях

Конструктивна узгодженість Біггса [93] та підхід CDIO (Conceive, Design,

Implement, Operate – «Сприймати, Проектувати, Впроваджувати, Експлуатувати»), підкреслюють необхідність узгодження результатів навчання, навчальних активностей і методів оцінювання, а також орієнтацію практичних завдань на реальні інженерні сценарії [94, 95]. Водночас ці підходи не пропонують формального машинно-читаного представлення практичних завдань, придатного для автоматизованого виконання та перевірки у DLE. Паралельно розвиваються віртуальні лабораторії та навчальні середовища, що забезпечують студентам доступ до інструментально насичених середовищ моделювання та експериментування. Такі системи сприяють розвитку практичних навичок [96, 97] і дозволяють автоматизувати оцінювання результатів навчання, однак їх використання у масових курсах потребує ефективних механізмів формального опису та перевірки завдань [83]. В існуючих стандартах завдання формалізуються за допомогою представлень, орієнтованих на питання-відповіді або вхід-вихід, таких як специфікації IMS QTI та завдання з програмування на основі тестових випадків [98-100]. Ці представлення є ефективними для вікторин та алгоритмічних задач, але є недостатніми для сучасних інженерних завдань, що включають конфігурацію системи, управління інфраструктурою та взаємодію зі складними ланцюгами інструментів. Системи автоматизованого оцінювання програмування на кшталт Kattis [101], DOMJudge [102], CodeRunner [103] також здебільшого використовують модель перевірки «вхід–вихід», яка є ефективною для алгоритмічних задач, але недостатньою для завдань системного адміністрування або інфраструктури [99]. Як результат, існуючі системи зосереджуються на формалізації того, що виробляється, а не на тому, як це виробляється, залишаючи процес виконання, використання інструментів та трансформації стану системи поза межами формального представлення та перевірки завдань (результати \neq процес \neq інструменти \neq стани).

Формальні представлення навчальних сценаріїв дозволяють створювати доменні моделі, які можуть автоматично оброблятися для генерації поведінки системи та її конфігурацій, що підкреслює важливість формального опису завдань поза межами простих моделей «вхід–вихід» [104]. Формальні методи традиційно

використовують математичні моделі для специфікації та верифікації систем, і ці принципи можуть бути застосовані для формалізації навчальних завдань [105]. Подібні підходи досліджувалися в інших областях, зокрема в алгебрі завдань, де завдання описуються як ієрархічні структури з чіткою синтаксичною та семантичною організацією [106], а також у метамоделях завдань (TOOD), що моделюють структуру цілей, декомпозицію завдань та дії користувачів [107].

Дослідження у сфері AI в освіті (AIEd) показують, що хоча AI активно використовується для автоматизації освітніх процесів, формальні методи залишаються недостатньо застосованими для специфікації навчальних завдань на рівні освітніх систем [108]. Наявні підходи до формального моделювання в AIEd здебільшого зосереджені на оцінці та прогнозуванні результатів навчання [109], а не на формальному описі структури завдань, процесу їх виконання чи використання інструментів. Разом з тим з'являються концептуальні моделі інтеграції AI в освітню оцінку, які враховують процесуальні, когнітивні та організаційні аспекти освітньої діяльності [110], а також підходи до організації агентних робочих процесів для виклику інструментів і координації виконання завдань у VLE [111].

Водночас бракує єдиної формальної моделі, яка б описувала PPET, поєднуючи педагогічні вимоги (цілі навчання, обмеження) з вимогами до інфраструктури (конфігурація VLE) та логікою верифікації для складних, недетермінованих станів із специфікацією використовуваних інструментів. Аналіз існуючих підходів виявляє фундаментальне обмеження у формальному описі PPET у стандартних системах електронного навчання. Традиційні моделі базуються на парадигмі «вхід-вихід», яка є ефективною для завдань алгоритмічного програмування, але недостатньою для завдань системної інженерії, DevOps та управління інфраструктурою. Сучасні підходи до моделювання освітніх завдань значною мірою ігнорують процес генерування результатів, зосереджуючись виключно на результаті. Однак у сучасній інженерній освіті практичне завдання зазвичай має два виміри: зміну стану (зміна стану системи – конфігурація сервера) та генерування артефактів (створення програмних компонентів – скриптів, коду, конфігурацій). А поява AI вимагає, щоб

формалізована модель чітко враховувала інструменти, які використовує учень. Результат, досягнутий за допомогою ручного кодування, педагогічно відрізняється від результату, досягнутого за допомогою GenAI, навіть якщо функціональний результат є ідентичним. Тому актуальним є розробка формальної моделі, яка чітко розділяє цілі завдань, методи, критерії якості, інструменти та середовище виконання та забезпечуючи систематичну автоматизацію створення, впровадження та оцінки завдань у DLE.

Формальні підходи до моделювання інтелектуальних агентів представлені переважно в межах BDI-парадигми (Belief–Desire–Intention) [112], яка забезпечує логічне представлення станів агента та механізмів прийняття рішень. В [113] запропоновано узагальнену формальну модель BDI-агентів, що дозволяє описувати асистентів як багатоконтекстні системи з формалізованими правилами взаємодії між компонентами. Такі моделі застосовуються для специфікації рекомендаційних агентів, що підтверджує можливість формального опису поведінки асистентів [113]. У більш сучасних дослідженнях BDI-підхід також використовується для моделювання діалогових агентів, поєднуючи логічні моделі з методами обробки природної мови [114]. Водночас сучасні LLM-базовані AI-асистенти переважно не мають формального представлення, що обмежує можливості їх верифікації, контролю та інтеграції в освітні системи. Отже, існує недостатня формалізація моделей персональних AI-асистентів у навчальних системах. Наявні рішення здебільшого реалізуються як прикладні інструменти без чітко визначеної моделі, що обмежує їх інтеграцію з іншими компонентами освітніх технологій.

Залишається відкритим питання поєднання формальних моделей з можливостями GenAI з метою забезпечення контрольованої і відтворюваної генерації навчальних завдань. Також не вирішено задачу інтеграції процесів генерації, виконання та перевірки завдань у межах єдиної інформаційної технології, що ускладнює автоматизацію повного життєвого циклу практичного завдання. Виявлені обмеження визначають необхідність розробки нових формальних моделей AI-асистентів, які б забезпечували структурну, процесуальну та інструментальну

репрезентацію PPET, були здатні поєднати можливості GenAI з вимогами до структурованості та контрольованості, і дозволяли інтегрувати їх у сучасні DLE. Крім того, існуючі дослідження охоплюють окремі аспекти практичного персоналізованого навчання, проте не забезпечують цілісного підходу до його формалізації та автоматизації.

Для PPET важливо, аби завдання були структурованими, машиночитаними, параметризованими; ці аспекти забезпечується формальним визначенням внутрішньої структури цих завдань [115]. Збалансованість практичних завдань може бути досягнута шляхом використання DSL, за допомогою якої можна спростити розробку навчальних компонентів як викладачами при створенні методичних вказівок до PPET, так і студентами в процесі самостійного опанування навчального контенту. Спроби використати DSL в освітніх цілях фіксуються впродовж останніх років [116-120]. В [121] вказують на наступні переваги DSL: підвищення виразності (*expressiveness*) та простоти використання, зменшення семантичної відстані між проблемою та реалізацією. Однак розробка спеціалізованих мов для формалізації опису шаблонів практичних завдань залишається недостатньо розвиненою. Існуючі DSL мають обмежений функціонал, низку концептуальних і технологічних недоліків щодо опису типів завдань, способів перевірки, рівнів складності, контекстів виконання або взагалі покривають кардинально інший домен (наприклад, сферу вивчення іноземних мов [122]).

Мова, описана в [122] базується на парадигмі Model-Driven Engineering, проте функціонує в іншому домені та на іншому рівні абстракції. Передбачене її використання графічної нотації для опису структури навчального процесу, сценаріїв, навчального контенту, мультимедійних ресурсів та способів представлення матеріалу. Основною метою цієї DSL є формалізація дизайну навчальних активностей та подальша генерація програмного коду для освітніх застосунків, описано “що викладати”, але не описано, “яке завдання згенерувати конкретному студенту”. Незважаючи на активне застосування підходів Model-Driven Engineering у системах Technology-Enhanced Learning, існуючі рішення орієнтовані на

моделювання та генерацію навчальних систем [123]. Платформа MDENet [124] також реалізує підхід до організації навчальних активностей у сфері Model-Driven Engineering у вигляді веб-орієнтованого середовища без необхідності інсталяції інструментів. Навчальні активності описуються декларативно за допомогою конфігураційних файлів у форматах YAML/JSON, що визначають структуру інтерфейсу, доступні інструменти та сценарій виконання. MDENet має низку обмежень: відсутня формалізація практичних завдань як обчислюваних сутностей; не реалізовано механізмів їх параметризації, генерації та автоматичної перевірки; підвищена складність підтримки при редагуванні і масштабуванні. Використання YAML/JSON як основи опису обмежує можливості семантичного контролю та валідації. У результаті система орієнтована також переважно на організацію навчального середовища, а не на формалізацію та автоматизацію змістового рівня навчання. YAML/JSON тут виконують роль формату представлення даних, а не повноцінної мови опису задач.

В роботі [116] розроблено DSL для автоматизованого створення та налаштування віртуальних класів в рамках LMS. Вона забезпечує високий рівень абстракції та візуальність, за рахунок використання графічного режиму в рамках Eclipse Graphical Modeling Framework і дозволяє викладачам створювати класи схематично. Ця мова тісно пов'язана з конкретною платформою, спрямована на структурування середовища навчання, і не моделює сам зміст завдань, не передбачено адаптивних механізмів і не придатна для складних сценаріїв. В [117,118] представлена нова вбудована домен-специфічна мова EDSL, яка є центральним елементом фреймворку для опису параметризованих завдань для курсу з програмування на Haskell-I/O та їх правильних рішень. Автори вказують на наявні обмеження та недоліки: залежність від якості генераторів специфікацій, які повинні бути надані викладачем, ризик створення нерозв'язних завдань, а система генерації тестів стикається з проблемами застрягання при пошуку тестів. Quiz-game DSL [119] створена для вікторин/освітніх ігор і дозволяє описати структуру вікторини: запитання, рівні складності, логіку, але сфокусована лише на вікторинах - не

підтримує складні сценарії, симуляції, інтеракції зі складною логікою. У [120] представлено генератор питань, який використовує власну легку вбудовану домен-специфічну мову для опису шаблонів питань, ключовою особливістю якої є можливість параметризації, визначення діапазонів їх значень та потенційних взаємодій чи залежностей між ними, що дозволяє генерувати цілі родини схожих питань, але якість залежить від розробника, а не генератора. Сама мова має дуже обмежений функціонал і підходить тільки для запитань з однією відповіддю в рамках тільки однієї LMS.

Незважаючи на доцільність запропонованих рішень, вони мають низку концептуальних і технологічних обмежень їх застосування для гнучкої генерації персоналізованих практичних завдань. Існуючі рішення мають такі спільні обмеження: інший домен, залежність від конкретної платформи, недостатня підтримка формального опису логіки перевірки, орієнтація переважно на макрорівень (курси, ресурси), а не мікрорівень (структура і зміст завдань), відсутність механізмів адаптації до опису практичних завдань, низька адаптація для взаємодії із сучасними технологіями на кшталт AI.

З погляду на це дуже перспективним є створення домен-специфічної мови, яка забезпечить викладачів формальним і гнучким інструментом для опису практичних завдань, який підтримуватиме контрольовану генерацію тексту завдання та розгортання VLE, міститиме правила для автоматичної перевірки прогресу виконання завдання і може бути використана в сучасних інформаційних системах автоматизованої генерації та перевірки параметризованих практичних завдань, в тому числі в комбінації із методами GenAI. Запропонований гібридний підхід відповідає передовому напрямку досліджень у галузі комп'ютерних наук та освітніх технологій. В [125] автори пропонують гібридний фреймворк, що поєднує LLM та DSL для генерації коду.

Тому актуальною є розробка формальної DSL для підвищення ефективності програмних засобів генерації та перевірки практичних завдань за рахунок створення моделі завдання, за яким AI-компонент інформаційної технології створить унікальне

завдання для кожного студента необмежену кількість разів, необхідну для якісного засвоєння матеріалу, збільшення швидкодії процесів розгортання VLE за рахунок їх автоматизації, зручності сприйняття шаблону завдань, зменшення кількості помилок при створенні зв'язків між елементами моделі та зменшення часу на модифікацію моделі. Автор дослідження пропонує не просто використовувати AI, а керувати ним за допомогою формальної мови. Пропонується використати DSL не лише як інструмент опису, але й як ключовий компонент для взаємодії з новітніми LLM [126].

Огляд сучасних праць присвячених **функціональним моделям практичного завдання** (його життєвого циклу) показує, що наукове поле вже має вагомі напрацювання щодо активного навчання, персоналізованих траєкторій, навчальної аналітики і автоматизованого зворотного зв'язку, однак ці напрацювання здебільшого описують окремі фрагменти освітнього процесу, а не цілісний життєвий цикл RPET [127-130]. Тому дослідницьке завдання полягає в тому, щоб побудувати функціональну модель RPET, яка формалізує послідовність його етапів, забезпечує наскрізний зв'язок між ними та створює підґрунтя для автоматизації в межах різних інженерних спеціальностей.

Систематичний огляд V.Sukacké та співавторів демонструє, що problem-based, project-based і challenge-based learning в інженерній освіті доцільно аналізувати через етапи інструкційного дизайну за моделлю ADDIE: analysis, design, development, implementation, evaluation [127]. Для нашого дослідження це особливо важливо, оскільки саме така логіка дозволяє інтерпретувати практичне завдання не як одноразову активність, а як процес із чітко визначеними фазами. Практико-орієнтований вимір цієї логіки посилюють дослідження лабораторного й виробничо-симуляційного навчання. Зокрема, в [131] наведено, що ефективна організація віддалених лабораторій потребує структурованого поєднання відкритого завдання, підготовки до лабораторії, командної взаємодії, проміжних консультацій із формувальним зворотним зв'язком і підсумкової рефлексії. В [132] автори, узагальнюючи емпіричні результати щодо навчальних лабораторій, доводять, що

такі середовища здатні розвивати професійні, методологічні, соціальні та компетентності самоорганізації, тобто підтримують універсальний каркас практичної підготовки майбутніх інженерів.

У персоналізованому навчанні вагомою є праця M.L.Bernacki, M.J.Greene та N.G.Lobczowski, які на основі систематичного огляду 376 досліджень запропонували аналізувати персоналізацію за чотирма питаннями: хто персоналізує, що саме персоналізується, яким способом це здійснюється та з якою метою [128]. Така рамка є корисною для побудови функціональної моделі, оскільки дозволяє явно визначити суб'єктів персоналізації, параметри налаштування завдання та очікувані результати.

Ближчим до інженерної освіти є дослідження [133], у якому автори поєднали тривимірну сітку знань і навичок, інтерактивні діагностичні перевірки прогресу та адаптивний пул навчальних ресурсів. Це важливий прецедент, оскільки він показує, як персоналізацію можна вбудувати у формальну структуру кредитних інженерних курсів без втрати ролі викладача.

Значну роль у майбутній моделі має відігравати аналітика навчання. Систематичний огляд [129] показує, що системи зворотнього зв'язку доцільно описувати через чотири виміри: які дані збираються, якими методами вони аналізуються, для яких цілей використовується аналітика та яким стейкхолдерам адресовано результат. Ця рамка фактично задає інформаційну архітектуру для автоматизації життєвого циклу завдання. Подібну логіку розвиває A.Pardo та співавтори в платформі OnTask, де персоналізовані навчальні підтримувальні дії формуються на основі подання про студента, набору правил і каналів адресної комунікації [134]. Разом із цим сучасні праці наголошують, що сам факт автоматизації ще не гарантує якості підтримки. G.Deeva та співавтори, проаналізувавши 109 систем автоматизованого зворотного зв'язку, підкреслюють фрагментованість цього поля та потребу в більш даних орієнтованих, студентоцентричних рішеннях [130]. Емпіричні дослідження підтверджують цінність саме змістовного, персоналізованого й педагогічно вмотивованого зворотного зв'язку [135]: відзначається позитивне сприйняття персоналізованої

навчальної аналітики та зворотного зв'язку студентами, а в [136] автори демонструють, що інформативний зворотній зв'язок, згенерований на основі аналітики навчання, покращує сприйняття його корисності, розуміння навчального прогресу, саморегуляції порівняно з простішими формами повідомлень.

Проведений аналіз дає змогу сформулювати кілька принципових висновків:

- у працях з інженерного навчання добре опрацьовано окремі формати практичної підготовки, але вони не формують єдиної міждисциплінарної моделі RPET або його життєвого циклу [127, 131, 132];
- у дослідженнях персоналізованого навчання вже існують рамки для опису параметрів індивідуалізації, однак вони лише частково охоплюють практичне завдання як цілісний об'єкт педагогічного проєктування [128, 133];
- аналітика навчання та автоматизований зворотний зв'язок дають достатній інструментальний базис для побудови механізмів моніторингу, персоналізованих підказок і адаптації наступних кроків, але ці рішення переважно зосереджені на локальних інтервенціях, а не на повному замкненому циклі [129, 130, 134];
- перспективним напрямком є поєднання інструкційного дизайну, персоналізації та автоматизованого зворотного зв'язку в єдиний ланцюжок, де результати виконання одного завдання слугують вхідними даними для конфігурування наступного [133, 136];
- наявна наукова прогалина полягає не стільки у відсутності окремих рішень для персоналізації чи автоматизації, скільки у відсутності узгодженої функціональної моделі, яка (а) описує повний життєвий цикл RPET від діагностики до рефлексії, (б) задає ролі викладача, студента та цифрової системи на кожному етапі, (в) формалізує вхідні дані, керувальні впливи, механізми виконання та виходи процесу, (г) придатна до використання в різних інженерних доменах без прив'язки до одного навчального формату.

Розробка формальних моделей і мови, що дозволяють змодельовати і описати RPET, створити інформаційну технологію для отримання сталих практичних

навичок в масовому онлайн-навчанні з дотриманням норм академічної доброчесності, є актуальним і обґрунтованим науковим завданням.

1.6 Постановка задач дослідження

У результаті проведеного аналізу встановлено, що підготовка інженерних кадрів ґрунтується на засадничій ролі безперервної практичної діяльності («learning-by-doing»). Для забезпечення необхідної інтенсивності та варіативності практик для набуття сталих практичних навичок в умовах масового дистанційного навчання критично необхідне масштабування процесів через їх повну автоматизацію з підтримкою персоналізації та механізмів контролю академічної доброчесності.

Досягнення масштабованої автоматизації можливе лише за умови комплексного охоплення всього життєвого циклу практичного завдання – від генерації завдання та розгортання VLE до його перевірки. Наявні підходи не забезпечують повної автоматизації процесів генерації, перевірки та розгортання практичних завдань. Виявлено, що головною перешкодою для такої автоматизації є відсутність уніфікованих формальних моделей та спеціалізованої мови, здатних описувати життєвий цикл практичного завдання, що унеможлиблює їх масштабовану автоматизацію.

Обґрунтовано застосування GenAI для оптимізації процесів персоналізації, проте встановлено, що застосування GenAI потребує обережного інтегрування з механізмами контролю, валідації й адаптивності.

В рамках існуючої наукової проблеми, що полягає у забезпеченні набуття студентами інженерних спеціальностей сталих практичних навичок в умовах масового дистанційного навчання, поставлено наукове завдання щодо комплексного забезпечення процесів автоматизованої генерації персоналізованих практичних інженерних завдань, автоматичного розгортання необхідних VLE та автоматичної перевірки результатів виконання цих завдань для набуття студентами сталих практичних навичок.

Метою дисертаційної роботи є підвищення рівня персоналізованого навчання студентів та набуття ними сталих практичних навичок за рахунок масштабованої

автоматизації процесів створення, виконання та оцінювання результатів виконання персоналізованих практичних інженерних завдань з дотриманням норм академічної доброчесності.

Для досягнення мети необхідно вирішити такі задачі дослідження:

- виконати системний аналіз існуючих методів персоналізації практичних завдань у підготовці студентів інженерних спеціальностей;
- виявити обмеження існуючих підходів та визначити вимоги до формалізації практичних завдань і навчальних середовищ;
- визначити потенційні проблеми з контролем академічної доброчесності при дистанційному навчанні;
- розробити систему кількісних показників оцінки персоналізації для наукового аналізу, порівняння, та подальшого вдосконалення персоналізованого навчання;
- розробити функціональну модель PPET, яка формалізує етапи створення, виконання та оцінювання з урахуванням контексту навчання;
- розробити формальну модель PPET, що визначає його структуру, параметри, середовище виконання та критерії оцінювання, придатну до автоматизованої обробки;
- розробити архітектуру персонального AI-асистента, інтегрованого з моделями практичного завдання, для підтримки процесів генерації, адаптації та супроводу виконання завдань;
- сформулювати вимоги до DSL, яка покриває домен PPET;
- розробити формальну граматику DSL для опису PPET;
- розробити парсер та транслятор домен-специфічної мови для опису PPET;
- удосконалити методи автоматизованої генерації PPET, автоматичного розгортання VLE та автоматичної перевірки таких завдань;
- розробити програмні засоби реалізації запропонованих моделей і методів;
- виконати UML-проектування компонентів підсистеми персоналізованого практичного навчання з урахуванням розроблених методів персоналізації;

- розробити архітектуру підсистеми автоматизованої генерації та перевірки персоналізованих практичних завдань;
- розробити інформаційну технологію персоналізації навчання, яка забезпечує інтеграцію моделей, методів та програмних засобів у єдиний комплекс автоматизованої генерації, виконання та перевірки практичних завдань;
- порівняти можливості різних LLM-моделей для автоматизованої генерації;
- оцінити ефективність розроблених моделей, методів та ІТ шляхом проведення експериментів за участю реальних студентів та завдань.

1.7 Висновки до розділу I

У першому розділі здійснено системний аналіз теоретичних і практичних аспектів персоналізованого навчання студентів інженерних спеціальностей, а також визначено наукові лакуни в питанні формалізації практичних завдань, сформульовано передумови створення ІТ автоматизованої генерації та перевірки персоналізованих практичних завдань.

1. Уточнено понятійно-термінологічну базу персоналізованого навчання. Визначено відмінності між персоналізованим, адаптивним, індивідуалізованим та кастомізованим навчанням, що дозволило конкретизувати поняття і місце персоналізації у контексті практичної підготовки інженерів та визначено роль практичних завдань у формуванні компетентностей майбутніх інженерів. Обґрунтовано, що специфіка інженерної освіти вимагає персоналізації саме на рівні практичних завдань як найбільш ефективного механізму підвищення якості навчання та забезпечення академічної доброчесності в умовах дистанційної освіти.

2. Розроблено узагальнену класифікацію рівнів персоналізації практичних завдань, запропоновано індекс персоналізації для кількісної оцінки рівня персоналізації конкретних систем або методів.

3. Проаналізовано сучасні методи підтримки персоналізованого навчання та визначено перспективні напрями їх автоматизації. Показано доцільність інтеграції GenAI для оптимізації процесів персоналізації. Обґрунтовано, що ефективним рішенням для подолання недоліків AI та забезпечення детермінованості перевірки є

використання DSL як посередника між педагогічним сценарієм та інфраструктурною реалізацією.

6. Сформульовано та обґрунтовано вимоги до VLE, серед яких критичними визначено: віддалене розгортання, реалістичність, обфускацію, ізоляцію та інтероперабельність.

7. Виявлено відсутність уніфікованих формальних інструментів для комплексного опису PPET, які б одночасно забезпечували параметризовану генерацію тексту, автоматичне розгортання VLE та верифікацію результатів і могли б бути використані в сучасних автоматизованих інформаційних технологіях, в тому числі в комбінації із методами GenAI. Порівняльний аналіз показав фрагментацію існуючих підходів, що робить їх непридатними для складних інженерних чи інфраструктурних задач.

8. Визначені задачі дослідження і встановлено, що для необхідної інтенсивності та варіативності практик, які забезпечують набуття сталих практичних навичок, в умовах масового дистанційного навчання критично необхідне масштабування процесів через їх повну автоматизацію. Проте досягнення масштабованої автоматизації можливе лише за умови комплексного охоплення всього життєвого циклу PPET – від генерації та розгортання VLE до його перевірки.

Результати досліджень, приведених в розділі, опубліковані в роботах [38, 70].

РОЗДІЛ II. МОДЕЛІ ПЕРСОНАЛІЗОВАНОГО НАВЧАННЯ З ІНЖЕНЕРНИХ СПЕЦІАЛЬНОСТЕЙ

У роботі застосовано підхід централізованої формалізації і всі ключові елементи – модель життєвого циклу, модель практичного завдання, мова його опису та формалізовані структури взаємодії – зведені до єдиного формального базису, представленого у цьому розділі. Такий підхід є обґрунтованим і підсилює наукову складову роботи, оскільки забезпечує узгодженість моделей, методів і засобів їх реалізації. Водночас, з метою уникнення змішування рівнів абстракції, у роботі чітко визначено логіку та ієрархію формальних елементів.

У даному розділі РРЕТ розглядається у двох взаємодоповнюючих аспектах: як процес і як об'єкт. Такий підхід обумовлений необхідністю забезпечення повноти його формалізації для подальшої автоматизованої обробки. З одного боку, завдання розглядається як процес, що реалізується через послідовність етапів його виконання, що відображається у функціональній моделі. З іншого боку, завдання розглядається як структурований об'єкт, що має визначений склад, параметри та взаємозв'язки, що формалізується у вигляді формальної моделі. Поєднання цих підходів дозволяє узгодити динамічні та структурні характеристики завдання, що є необхідним для побудови єдиного формального базису автоматизованої генерації, розгортання та перевірки. Запропоноване поєднання процесного та об'єктного підходів утворює теоретичну основу для розробки нової домен-специфічної мови, яка забезпечує уніфікований формальний опис завдань у контексті їх автоматизованої обробки.

2.1 Функціональна модель РРЕТ

Основою для побудови інформаційної технології персоналізації практичної підготовки інженерних спеціальностей є розробка функціональної моделі, яка описує процеси створення, розгортання, виконання та перевірки навчальних завдань. Функціональна модель РРЕТ формалізує послідовність етапів завдання (описує життєвий цикл) і забезпечує наскрізний зв'язок між ними. Для узагальненого представлення обрано методологію функціонального моделювання IDEF0 [137], що

дозволяє чітко розмежувати вхідні дані, керуючі впливи, результати (виходи) та механізми реалізації на кожному етапі системи.

На відміну від існуючих підходів, де процеси генерації умов завдання та розгортання інфраструктури для його виконання є розрізненими, запропонована функціональна модель формалізує повний життєвий цикл PPEТ як єдиний безперервний конвеєр із замкненим адаптивним циклом зворотного зв'язку. Загальну концепцію подано у вигляді контекстної діаграми (рисунок 2.1), яка описує головний процес – «Персоналізоване практичне інженерне завдання».

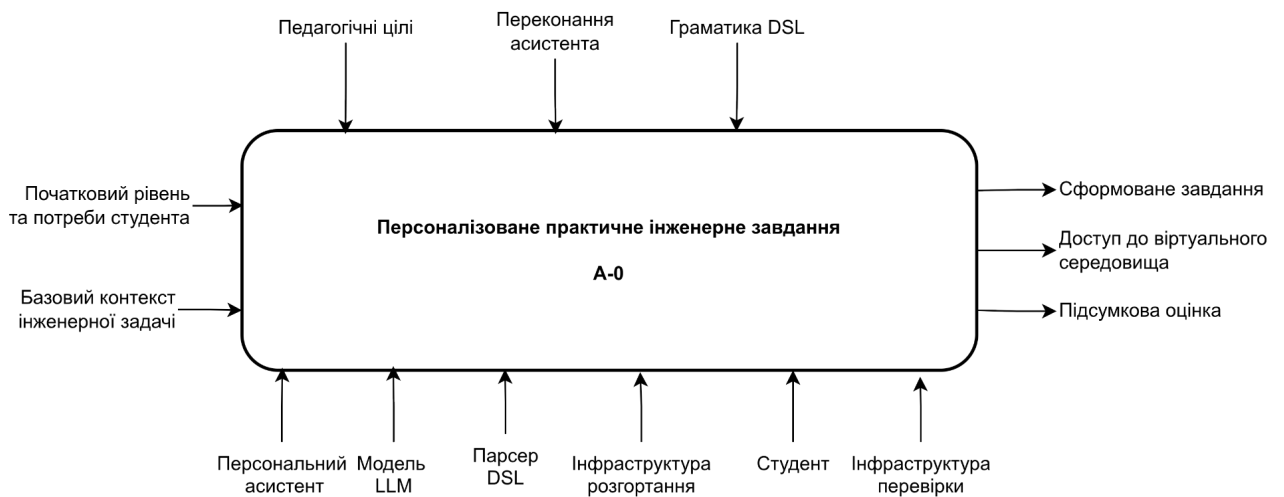


Рис. 2.1. Контекстна діаграма PPEТ.

Взаємодія системи із зовнішнім середовищем на контекстному рівні визначається такими інформаційними та керуючими потоками.

Головний процес: Персоналізоване практичне інженерне завдання.

Входи: Дані, що ініціюють процес і підлягають перетворенню.

1. *Початковий рівень та потреби студента:* Інформація про поточні знання, прогалини та запити студента (профіль).
2. *Базовий контекст інженерної задачі:* Загальна тема, лабораторна робота або тип інфраструктурного завдання, яке необхідно відпрацювати.

Керування: Правила, обмеження та знання, що регулюють процес.

1. *Педагогічні цілі:* Вимоги освітньої програми, стандарти компетентностей.

2. *Граматика DSL*: Синтаксичні та семантичні правила формального опису, яких обов'язково має дотримуватися система при генерації завдань.
3. *Початкові переконання асистента*: Апріорні знання інтелектуального асистента про систему.

Виходи: Кінцеві корисні результати роботи системи.

1. *Сформоване завдання*: Готовий формалізований файл, що зберігається в базі даних (або LMS, наприклад, Moodle) як "єдине джерело правди".
2. *Доступ до VLE*: Точка входу (URL/IP) для студента до розгорнутої та налаштованої інфраструктури.
3. *Підсумкова оцінка*: Результат автоматизованої перевірки практичних РРЕТ.

Механізми: Ресурси (програмні або людські), які безпосередньо виконують роботу.

1. *Персональний асистент*: Інтелектуальний компонент управління життєвим циклом та адаптацією.
2. *Модель LLM*: Генеративна нейромережа для синтезу унікальних умов і параметрів.
3. *Парсер DSL*: Детермінований аналізатор для валідації DSL-граматики та перевірки результатів.
4. *Інфраструктура віртуалізації*: Програмно-апаратне середовище для розгортання ізольованих лабораторних стендів (наприклад, Docker/VirtualBox).
5. *Студент*: Людина, яка виконує завдання.
6. *Інфраструктура перевірки*: Програмний комплекс, який підключається до розгорнутого VLE для виконання перевірки.

Для деталізації внутрішньої логіки та послідовності перетворень контекстну діаграму декомпозитовано на шість взаємопов'язаних етапів (підпроцесів), що формують рівень A0. Діаграму декомпозиції зображено на рисунку 2.2.

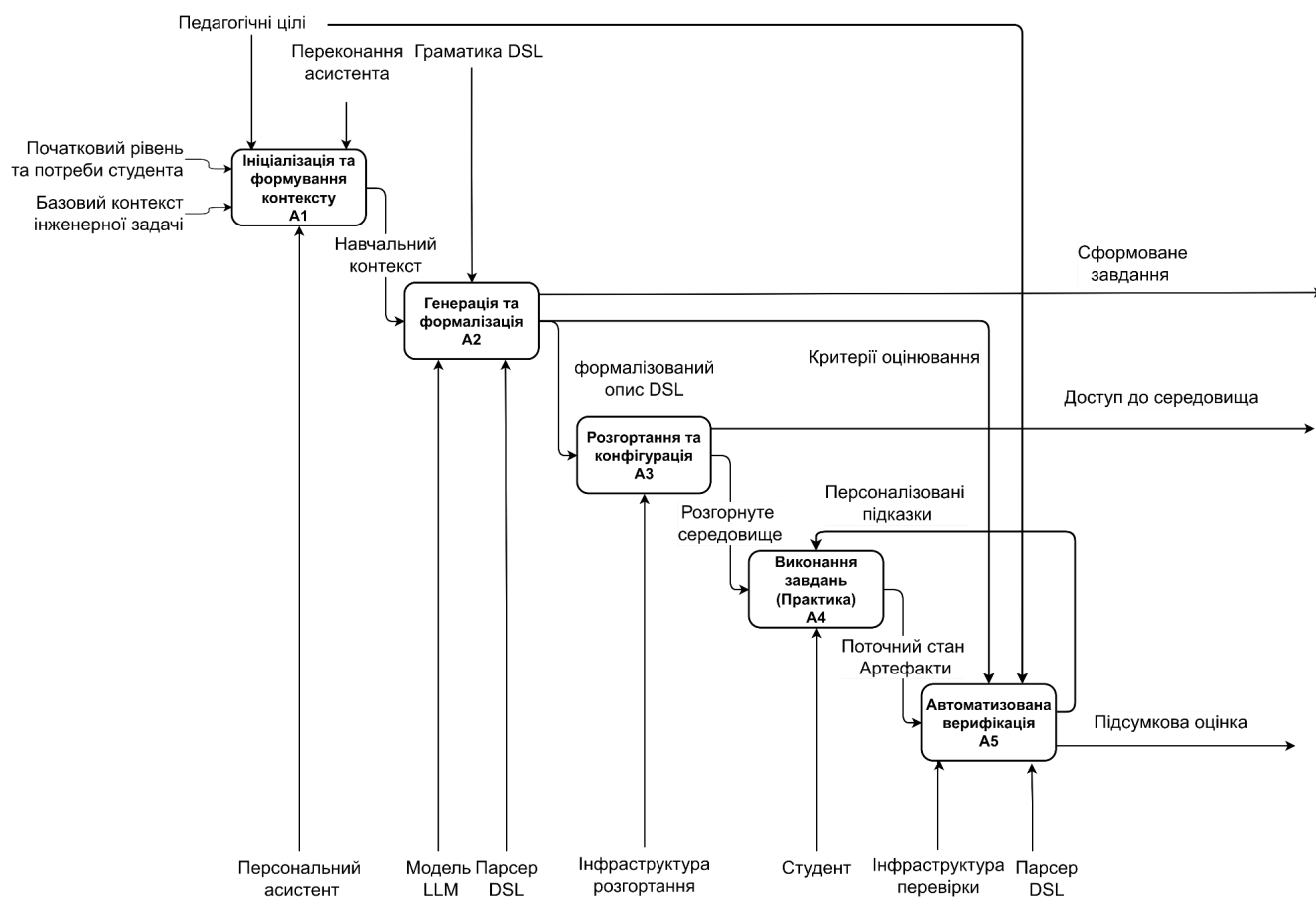


Рис 2.2. Діаграма декомпозиції PPET.

Декомпозиція розкриває головний процес як послідовність п'яти взаємопов'язаних етапів (підпроцесів), що утворюють замкнений конвеєр:

A1. Ініціалізація та формування контексту

Інтелектуальний асистент аналізує початковий рівень студента та базовий запит крізь призму педагогічних цілей і своїх початкових/оновлених переконань. Результатом є конкретизований навчальний контекст, який визначає оптимальну складність та параметри майбутнього завдання.

Входи: Початковий рівень, Базовий контекст.

Керування: Педагогічні цілі, Початкові переконання (для 1-ї ітерації), Оновлені переконання (зворотний зв'язок для наступних ітерацій).

Вихід: Навчальний контекст (конкретизований напрям для генерації).

Механізм: Персональний асистент.

A2. Генерація та формалізація

На основі навчального контексту та суворих правил граматики DSL велика мовна модель (LLM) синтезує унікальний варіант завдання. Згенерований код на льоту валідується парсером. Виходом є формалізований опис, який містить як конфігураційні параметри, так і критерії оцінювання. Сформовані критерії стають внутрішнім керуючим впливом для подальшого етапу верифікації, а конфігурація середовища та параметризація для етапу розгортання VLE.

Вхід: Навчальний контекст.

Керування: Граматика DSL.

Виходи: формалізований опис завдання (передається далі по процесу та виходить назовні як готовий артефакт бази завдань). Крім того, згенеровані критерії оцінювання стають внутрішнім керуванням для етапу A5, а параметри та конфігурація VLE стають внутрішнім керуванням для етапу A3.

Механізми: Модель LLM, Парсер DSL.

A3. Розгортання та конфігурація

Інфраструктура віртуалізації автоматично зчитує технічні параметри та ініціалізує початковий стан ізолюваного готового середовища (лабораторного стенда), надаючи доступ студенту.

Вхід: формалізований опис завдання.

Вихід: Готове середовище (передається студенту та виходить назовні як доступ).

Механізм: Інфраструктура віртуалізації.

A4. Виконання завдань (Практична взаємодія)

Студент взаємодіє з розгорнутим VLE, виконуючи поставлену задачу. У процесі роботи формується поточний стан інфраструктури або проміжні артефакти, які передаються на перевірку. Після успішного завершення завдання персональний асистент отримує підсумкову оцінку та історію виконання завдань студентом.

Вхід: розгорнуте середовище.

Керування: персоналізовані підказки (у випадку дозволу Педагогічними цілями).

Вихід: Артефакти / Фінальний стан середовища.

Механізм: Студент.

A5. Автоматизована верифікація

Створена інфраструктура перевірки у взаємодії з персональним асистентом автоматично зіставляють поточний стан із критеріями, закладеними в формалізованому описі завдання.

Вхід: Артефакти / Стан (від A4).

Керування: Критерії оцінювання (правила, закладені в формалізований опис на етапі A2).

Вихід: Оцінка.

Механізми: Інфраструктура перевірки, Парсер DSL.

Запропонована функціональна модель доводить можливість масштабування та персоналізації практичної підготовки інженерних спеціальностей. В розробленій моделі студент залучається як елемент механізм виключно на етапі практичної взаємодії (A4). Усі інші рутинні процеси – від синтезу унікальних умов (A1-A2) і налаштування інфраструктури (A3) до верифікації результатів (A5), повністю делеговані програмним агентам і відбуваються в автоматичному режимі завдяки використанню єдиних форматів даних.

2.2 Формальна модель PPET

Запропонована модель PPET розглядає практичне завдання як структуровану сутність, що визначає необхідні інструменти, допустимі дії, цільові стани системи та логіку перевірки. Такий підхід дозволяє автоматизувати процеси генерації, розгортання та оцінювання складних PPET у віртуальних навчальних середовищах. Водночас модель формує теоретичну основу для оцінювання результатів, отриманих в освітніх контекстах із використанням AI.

Слід зазначити, що у межах даної інформаційної технології механізми формалізації та контролю використання AI не реалізований. Водночас запропонована модель передбачає можливість такої формалізації шляхом визначення правил взаємодії з інструментами та перетворення неявних припущень щодо

процесу виконання завдання у явні формальні обмеження. Це створює передумови для подальших досліджень, спрямованих на забезпечення коректності та відтворюваності результатів незалежно від способів їх отримання.

Проблема розробки ефективних РРЕТ є міждисциплінарною і лежить на перетині інженерної педагогіки, педагогічного дизайну та інформатики. Формальна модель РРЕТ була розроблена із застосуванням змішаного методологічного підходу, що поєднує емпіричний аналіз, індуктивне узагальнення, теоретичний синтез та методи формальної специфікації. На першому етапі проаналізовано більше 120 практичних завдань з п'яти інженерних дисциплін (електротехніка, міцність матеріалів, термодинаміка, програмна інженерія, чисельні методи) для виявлення повторюваних структурних компонентів завдань. У додатку Д наведені завдання, що демонструють застосування процедури декомпозиції. Категорії стовпців у таблиці не були визначені апріорі, а виникли індуктивно під час початкової фази декомпозиції завдань. Подальше індуктивне узагальнення та структурна декомпозиція дозволили виділити інваріантні елементи завдань, незалежні від дисциплінарного контексту, що стало основою попередньої структури моделі. Узагальнення здійснювалося за ієрархічною схемою переходу від конкретних прикладів до доменно-незалежної формальної структури, придатної для автоматизованого виконання та перевірки. Остаточна модель РРЕТ формалізована з використанням теорії множин, логіки першого порядку та реляційного моделювання, що забезпечує чітку структуру компонентів завдання, формальне визначення умов правильності та явне представлення взаємозалежностей між компонентами.

Формальна модель РРЕТ визначається як структурована 7-ми компонентна модель формально специфікованих компонентів. Кожен компонент визначається як непорожня, добре типізована підмножина, внутрішня структура та обмеження якої залежать від області завдання. Вимоги до завдання та умови правильності виражаються за допомогою предикатів логіки першого порядку, що дозволяє проводити формальну верифікацію та автоматизоване міркування. Ці предикати забезпечують формальну основу для визначення того, чи є специфікація завдання

чітко визначеною та чи відповідає заданий результат заявленим цілям та обмеженням. Взаємодії між компонентами завдання фіксуються за допомогою явних бінарних відносин. Ці відносини роблять залежності явними та дозволяють перевіряти узгодженість між різними реалізаціями завдань.

PPET визначається як впорядкований кортеж 7 компонентів:

$$PPET = \langle C, D, G, M, Q, S, E \rangle$$

де кожен компонент представляє окремий структурний елемент PPET:

C (Контекст) – Специфікація домену та модель системи;

D (Дані) – Вхідні параметри та початкові умови;

G (Ціль) – Цільові результати та завдання;

M (Метод) – Застосовні парадигми вирішення проблем, теоретичні основи;

Q (Якість) – Критерії оцінки та умови прийнятності;

S (Інструменти) – Дозволені та заборонені інструменти досягнення мети;

E (Середовище) – Параметри середовища виконання PPET.

Модель включає шість основних відносин, описаних у Таблиці 2.1.

Таблиця 2.1

Відношення всередині моделі PPET

Назва	Формула	Опис
Dependency relation	$\rho_{dep} \subseteq D \times M$	Вказує, які вхідні дані необхідні для яких методів
Productivity relation	$\rho_{pro} \subseteq M \times G$	Вказує, які методи можуть досягти яких цілей
Validation relation	$\rho_{val} \subseteq Q \times (M \times G)$	Визначає застосовність критеріїв якості до пар «метод-мета»
Tool-method relation	$\rho_{tool} \subseteq S \times M$	Визначає, які інструменти можуть реалізовувати які методи
Environment-Method Relation	$\rho_{env} \subseteq E \times M$	Формалізує сумісність середовища та методу
Environment-Data Relation	$\rho_{prov} \subseteq E \times D$	Визначає сумісність між артефактами, що надаються в компоненті даних (D) (S_{init}) та можливостями обраного середовища.

Відношення валідації ρ_{val} навмисно визначено лише з критеріїв якості, методів та цілей. Інструменти виключені з відношення якості за задумом, щоб зберегти концептуальне розмежування між тим, що оцінюється (результати та їх правильність), і засобами, за допомогою яких ці результати отримуються.

Зауваження щодо взаємодії між середовищем та якістю. У запропонованій моделі навмисно не вводиться пряма бінарна залежність між компонентами «Середовище» (E) та «Якість» (Q). Вплив середовища виконання на оцінку якості є непрямим і опосередковується через сліди виконання та спостережувані артефакти, що утворюються під час виконання завдання. Формально, середовище обмежує виконання методів ($\rho_{env} \subseteq E \times M$), тоді як критерії якості оцінюються за результатами та слідами, пов'язаними з конкретним виконанням. Таким чином, будь-який вплив E на Q фіксується через екземпляр виконання, а не через структурну залежність між компонентами.

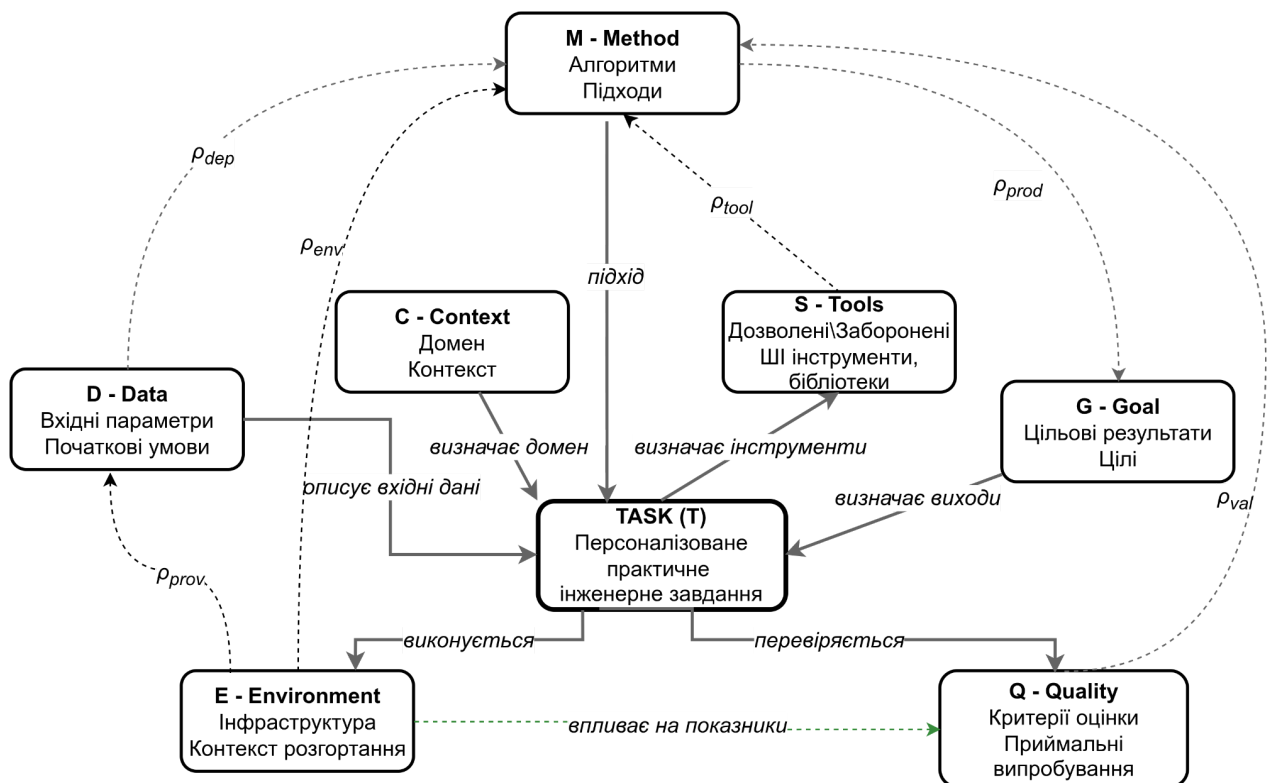


Рис. 2.3. Формальна модель PPET.

На рисунку 2.3 показано формальну модель завдання, що складається з семи компонентів. Компонент S (Інструменти) чітко відображає засоби досягнення результату, встановлюючи зв'язок інструменту ρ_{tool} з компонентом Метод, це має вирішальне значення для сучасної освіти, де інструменти GenAI кардинально змінюють простір доступних шляхів реалізації завдань. Компонент S чітко показує те, що раніше було неявним: засоби та інструменти, за допомогою яких отримуються результати, мають таке ж значення, як і самі результати, особливо в освітньому контексті, де навчальні цілі наголошують на розумінні та процесі, а не лише на правильному результаті.

Кожен компонент моделі PPET формалізовано або як впорядкований кортеж, або як скінченну множину кортежів. Перелічені внутрішні елементи складають мінімально достатнє незмінне ядро і не претендують на вичерпність. Умови семантичної валідності виражаються окремо як логічні предикати. Опис компонентів моделі наведений в додатку Е.

Валідність та застосовність запропонованої PPET-моделі оцінювались шляхом емпіричного інстанціювання на вибірці з 20 практичних завдань дисциплін «Операційні системи Unix», «Алгоритми» та «Бази даних». Під час маппінгу завдань до структури моделі виявлено типові неоднозначності традиційних текстових формулювань, зокрема змішування методів та інструментів, неявно визначене середовище виконання та недостатньо формалізовані критерії оцінювання. Це підтвердило здатність моделі формалізувати приховані структурні залежності та уточнювати припущення, що використовуються у практичних завданнях.

Валідація проводилась за чотирма критеріями: перевірка покриття компонентів, структурної узгодженості, реалізованості та операціоналізованості оцінювання. У результаті встановлено основні формальні властивості моделі: структурну повноту, узгодженість обмежень, відсутність циклічних залежностей між компонентами, семантичну ізольованість складових, реалізованість (існування

допустимої конфігурації виконання) та перевірюваність, що означає можливість формалізованої оцінки результатів виконання завдання. Попри позитивні результати, валідація має певні обмеження. По-перше, емпірична вибірка охоплює лише ІТ-орієнтовані дисципліни, тому подальші дослідження можуть розширити її на інші інженерні галузі. По-друге, проведена перевірка має структурно-концептуальний характер і не включає масштабного педагогічного експерименту щодо впливу моделі на результати навчання. По-третє, формалізація окремих традиційних завдань вимагає уточнення неявних припущень щодо середовища виконання та критеріїв оцінювання, що відображає особливості початкових описів, а не обмеження моделі.

2.3 Формальна мова опису PPET

2.3.1 Вимоги до розробки домен-специфічної мови LTDL

Для реалізації автоматизованих процесів генерації, розгортання VLE та перевірки виконання PPT пропонується DSL LTDL для опису завдань, яка базується на формальній моделі PPET. Не всі компоненти запропонованої моделі PPET потребують явного машинно-читаного представлення для автоматизації процесів генерації, розгортання VLE та перевірки виконання завдань. Частина компонентів не потребує прямої інтерпретації інформаційною системою. З цієї причини мова LTDL включає лише ті компоненти та атрибути, які необхідні для машинної обробки завдань у межах інформаційної технології. Підхід, за якого LTDL є операційною реалізацією частини моделі PPET, забезпечує компактність і практичну придатність мови для автоматизованої генерації, розгортання VLE та перевірки результатів виконання завдання, при цьому сама модель зберігає концептуальну повноту на рівні теоретичного опису завдань і відрізняється таким чином від існуючих представлень освітніх завдань, які базуються переважно на неформальних описах або тестуванні вхідних-вихідних даних.

Сфера застосування запропонованої в рамках роботи DSL полягатиме в описі PPET для автоматизованого генерування та автоматичної перевірки, тому має охоплювати:

1. Проектування завдань – визначення структури, змінних та обмежень

PPET для автоматичного генерування завдань з декількома еквівалентними варіантами.

2. Механізми персоналізації – адаптація змісту, складності та контексту завдань відповідно до рівня навичок, попередніх знань та потреб студентів (створення PPET).

3. Автоматизовану оцінку – забезпечення систематичної та об'єктивної перевірки рішень завдань, включаючи негайний зворотній зв'язок.

4. Кросплатформну інтеграцію – забезпечення можливості безперебійного виконання PPET у LMS та інших освітніх інструментах.

Цільова аудиторія – викладачі, розробники навчальних матеріалів та студенти, які хочуть швидко створювати інтерактивні PPET. LTDL у співпраці з інструментами GenAI задовольняє потреби викладачів та розробників навчальних програм у створенні адаптивних, параметризованих та орієнтованих на навички PPET, які можна адаптувати до індивідуальних профілів студентів та надавати на різних освітніх платформах без необхідності володіння просунутими навичками програмування.

2.3.2 Опис домену розробленої домен-специфічної мови

Практичне завдання (*task*) створюється з метою розвитку або оцінювання навичок студента в межах певної навчальної дисципліни. Для розроблення такого завдання необхідно чітко визначити його основну мету і, що найважливіше, конкретну навчальну ціль, якої має бути досягнуто після виконання завдання. Зазвичай ця ціль пов'язана з досягненням певного стану VLE або створенням артефакту, наявність якого можна перевірити. Наприклад, щоб навчитися створювати каталоги в ОС Unix, студентам може бути запропоновано завдання, що передбачає створення каталогу.

У багатьох випадках досягнення загальної мети практичного завдання потребує виконання кількох етапів. Тому кожен етап (*stage*) має бути пов'язаний із власною метою, досягнення якої підтверджує успішне завершення етапу та може бути формально перевірене. Етапи можуть також бути взаємозалежними.

Послідовності таких взаємопов'язаних етапів, які мають виконуватися у суворій послідовності, об'єднуються у конструкцію вищого рівня, трек (*track*).

Автоматичне розгортання кількох варіацій одного й того самого завдання шляхом модифікації окремих елементів реалізується за допомогою рандомізованих елементів, змінних (*variables*). Такі змінні можуть використовуватися для визначення назв файлів чи каталогів, імен користувачів, назв функцій або інших параметрів завдання. Це забезпечує унікальність кожного індивідуального варіанта завдання при збереженні однакової дидактичної складності між усіма версіями. Область видимості змінних може відрізнятися – вони можуть бути застосовані на рівні всього завдання, треку або окремого етапу. Важливою складовою завдання є *метадані*, які використовуються поза межами завдання, наприклад, для визначення, кому і за яких умов завдання слід призначати, або для оцінки очікуваного часу його виконання.

Виконання PPET зазвичай вимагає розгортання спеціального VLE. Для цього необхідно надати інструкції з розгортання середовища, які визначають, як саме створюється відповідне VLE. Ці інструкції поділяються на два типи: *інструкції створення*, які визначають, як саме ініціалізується середовище; та *інструкції налаштування* – визначають, як воно налаштовується. Обидва типи інструкцій використовують параметри, що включають специфічні для завдання персоналізовані змінні. Крім того, інструкції містять відомості про тип VLE, бекенд, який використовується, а також про протокол доступу та його параметри конфігурації. Приклад структури PPET зображено на рисунку 2.4. Ця структура ілюструє можливі елементи та зв'язки між ними. Існує три типи зв'язків “*використовує*” (використання змінної в межах сутності, де вона застосовується), “*включає*” (для вказання того, що сутність є частиною іншої), “*залежить*” (для позначення логічної послідовності етапів).

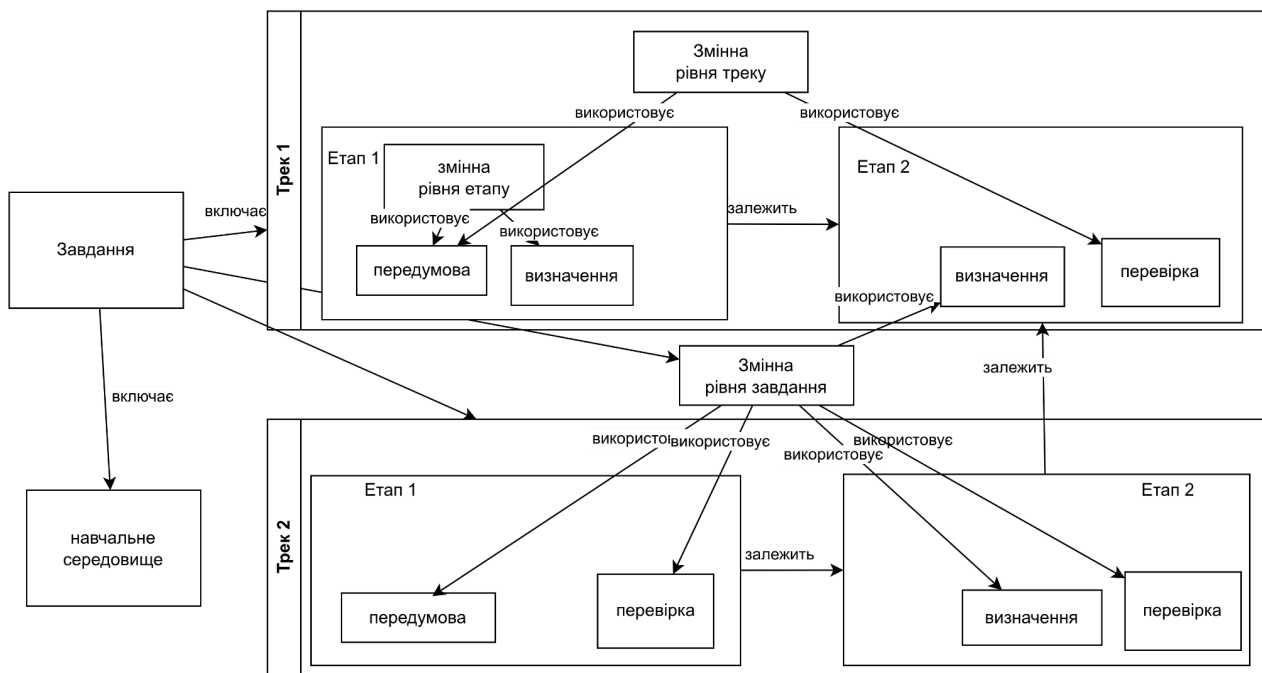


Рис.2.4. Приклад структури PPET.

Кожен етап може також містити “Попередні Інструкції” (*Precondition Instructions*), “Визначення Завдання” (*Task definitions*) та “Інструкції Перевірки” (*Check instructions*). “Попередні Інструкції” виконуються як завершальний крок під час налаштування VLE після його успішного розгортання; “Визначення Завдання” містить текстову інформацію для студентів, у якій пояснюється, що саме потрібно виконати на цьому етапі; “Інструкції Перевірки” застосовуються для відстеження прогресу виконання завдання та надання зворотного зв'язку в реальному часі (наприклад, для відображення поточного прогресу у відсотках).

2.3.3 Формальна граматики LTDL

На сьогодні теорія формальних мов для текстових мов програмування досліджена на високому рівні [138, 139]. Зразки формальних граматики для впорядкованих наборів символів описано у [140], а математичні засади формальних граматики – у [141]. Чітке формальне визначення граматики мови дає змогу уникнути неоднозначності при інтерпретації виразів і гарантує однозначне перетворення мовної конструкції в алгоритм.

Будь-яка формальна граматика описується алфавітом термінальних і нетермінальних символів, початковим символом і правилами виведення, які використовують для розпізнавання певних послідовностей символів. Алфавіт мови - це сукупність термінальних та нетермінальних символів. Із термінальних символів згідно з правилами виведення (граматики мови) будуться множини слів, які можуть бути в подальшому інтерпретовані як модель обчислень.

Формальна мова, окрім алфавіту, має ще синтаксис та семантику. Синтаксис визначає допустимі конструкції, а семантика пояснює їхній зміст. Синтаксис визначає набір правил, за якими складаються мовні конструкції. Розбір мовного виразу, складеного за синтаксичними правилами, виконується синтаксичним аналізатором мови у відповідності до правил виведення (продукцій), що задані граматикою мови. Семантика – це зв'язок між синтаксисом мови та моделлю обчислення. Формальна семантика задається математичною моделлю, що описує обчислення у відповідності до виразу, заданого мовою. Розробка LTDL здійснювалась згідно із принципів, закладених в [142-145]. Формальною породжувальною граматикою називають кортеж [146]:

$$G = (N, T, P, S),$$

де N – множина нетермінальних символів;

T – множина термінальних символів ; $N \cap T = \emptyset$;

P – множина правил виведення;

S – початковий символ.

Введемо алфавіт нетермінальних символів для опису сутностей PPET:

TC – вміст завдання,

LE – середовище навчання,

LEC – його вміст,

CI – інструкції створення навчального середовища,

PI – інструкції налаштування навчального середовища,

V – параметризована змінна,

TR – послідовність етапів завдання,

TRC – вміст послідовності,

ST – етап послідовності,

STC – вміст етапу,

I – інструкція, або дія яку треба виконати в навчальному середовищі,

A – сама дія,

D – визначення та опис завдання,

C – перевірка та фіксування результатів етапу.

Доповнимо алфавіт допоміжними нетерміналами, які задають списки сутностей:

VL – список параметризуючих змінних,

TRL – список треків,

STL – список етапів,

DL – список визначень завдання,

CL – список перевірок,

IL – список інструкцій.

Алфавіт термінальних символів складається з набору зарезервованих слів під кожен сутність ("task", "learning_environment", "create_instructions", "provision_instructions", "variable", "track", "stage", "instruction", "definition", "check"), знаків пунктуації ("=", "{", "}", "[", "]", ":", "(", ")", "\$", ".") та загальної буквено-цифрової послідовності (string_literal).

Правила виведення граматики представлені у Таблиці 2.2.

Таблиця 2.2

Правила виведення граматики (синтаксис мови)

№	Правило	Зміст правила
1	$S \rightarrow \text{"task" string_literal ":" " \{ " TC " \} "}$	Створення сутності практичного завдання з іменем.
2	$TC \rightarrow LE \mid \text{"stages" "=" "[" STL "]"}$ $\mid \text{"tracks" "=" "[" TRL "]"}$ $\mid \text{"variables" "=" "[" VL "]"}$	Завдання може складатися з опису навчального середовища (НС) та списків: етапів, послідовностей етапів, змінних.
3	$LE \rightarrow \text{"learning_environment" string_literal ":" " \{ " LEC " \} "}$	Створення сутності НС з іменем.

4	LEC \rightarrow VL CI PI	НС складається зі списків: змінних, інструкцій створення та налаштування.
5	CI \rightarrow "create_instructions" "=" "[" IL "]"	Створення списку інструкцій створення НС.
6	PI \rightarrow "provision_instructions" "=" "[" IL "]"	Створення списку інструкцій налаштування НС.
7	IL \rightarrow I I "," IL	Список інструкцій може складатися з однієї інструкції або списку інструкцій розділених комою.
8	I \rightarrow "instruction" string_literal ":" "{" A "}"	Створення інструкції з іменем.
9	A \rightarrow "action" "=" (string_literal "\$" ".")	Безпосередня дія, яка є послідовністю звичайних символів або спеціальною послідовністю для інтерполяції значення змінної у вигляді: \$scope.name.
1 0	TRL \rightarrow TR TR "," TRL	Список послідовностей може складатися з однієї послідовності або декількох послідовностей розділених комою.
11	TR \rightarrow "track" string_literal ":" "{" TRC "}"	Створення послідовності з іменем.
1 2	TRC \rightarrow "stages" "=" "[" STL "]" "variables" "=" "[" VL "]"	Послідовність складається зі списків: змінних та етапів.
1 3	STL \rightarrow ST ST "," STL	Список етапів може складатися з одного етапу, або декількох послідовностей розділених комою.
1 4	ST \rightarrow "stage" string_literal ":" "{" STC "}"	Створення етапу з іменем.
1 5	STC \rightarrow "variables" "=" "[" VL "]" "definitions" "=" "[" DL "]" "checks" "=" "[" CL "]" "preconditions" "=" "[" IL "]"	Етап складається зі списків: змінних, визначень, перевірок, інструкцій.
1 6	VL \rightarrow V V "," VL	Список змінних може складатися з однієї змінної або декількох змінних розділених комою.
1 7	V \rightarrow "variable" string_literal ":" "{" "}"	Створення змінної з іменем.
1 8	DL \rightarrow D D "," DL	Список визначень може складатися з одного визначення або декількох визначень розділених комою.
1 9	D \rightarrow "definition" string_literal ":" "{" "}"	Створення визначення та опису з іменем.
2 0	CL \rightarrow C C "," CL	Список перевірок може складатися з однієї перевірки або декількох перевірок розділених комою.
2 1	C \rightarrow "check" string_literal ":" "{" A "}"	Створення перевірки з іменем.

З метою створення транслятора формальна граматика LTDL була описана за допомогою розширеної нотації Бекуса-Наура (Extended Backus-Naur form - EBNF)

[147]. Для текстового представлення персоналізованих практичних завдань обрано синтаксис, подібний до HCL [148], оскільки він забезпечує простіший вигляд, ніж JSON. Приклад опису завдання в текстовій нотації LTDL з відповідним описом EBNF наведено на рисунку 2.5.

```

13 tracks = [
14   track "find_file": {
15     description = "String"
16     variables = [
17       variable "file_name": {
18         type = "RandomString"
19         length = 6
20         description = "File to be created by student"
21       }
22     ]
23   }
24   stages = [
25     stage "find_file": {
26       description = "Find file by pattern"
27       variables = [
28         variable "pattern": {
29           type = "RandomString"
30           length = 10
31           description = "String to be used to find file"
32         }
33       ]
34       preconditions = [
35         precondition "create_file": {
36           description = ""
37           action = "echo ${find_file.pattern} > ${find_file.file}"
38         }
39       ]
40       definitions = [
41         definition "find_file": {
42           template_type = "html"
43           template = "Please find file with text: ${find_file.pattern}"
44         }
45       ]
46     },
47     stage "copy_file": {
48       description = "Copy file to the destination"
49       variables = [
50         variable "source": {
51           type = "RandomString"
52           length = 10
53           description = "Source file name"
54         }
55         variable "destination": {
56           type = "RandomString"
57           length = 10
58           description = "Destination file name"
59         }
60       ]
61       preconditions = [
62         precondition "source_exists": {
63           description = "Source file exists"
64           action = "test -f ${source}"
65         }
66       ]
67       definitions = [
68         definition "copy_file": {
69           template_type = "html"
70           template = "Copy file ${source} to ${destination}"
71         }
72       ]
73     }
74   ]
75 }

```

```

60 trackDefinition* RBRACKET;
61 trackDefinition: TRACK STRING COLON LBRACE trackBody RBRACE;
62 trackBody: trackProperty*;
63 trackProperty: descriptionProperty
64 |variablesProperty
65 |stagesProperty;
66 stagesProperty: STAGES EQUALS LBRACKET
67 | stageDefinition* (COMMA stageDefinition)* RBRACKET;
68 stageDefinition: STAGE STRING COLON LBRACE stageBody RBRACE;
69 stageBody: stageProperty*;
70 stageProperty: descriptionProperty
71 |variablesProperty
72 |dependOnProperty
73 |preconditionsProperty
74 |definitionsProperty
75 |checksProperty;
76 dependOnProperty: DEPEND_ON EQUALS LBRACKET identifierList? RBRACKET;
77 identifierList: IDENTIFIER (COMMA IDENTIFIER)*;
78 preconditionsProperty: PRECONDITIONS EQUALS LBRACKET preconditionDefinition* (COMMA preconditionDefinition)* RBRACKET;
79 preconditionDefinition: PRECONDITION STRING COLON LBRACE preconditionBody RBRACE;
80 preconditionBody: preconditionProperty*;
81 preconditionProperty: descriptionProperty|actionProperty;
82 actionProperty: ACTION EQUALS stringValue;
83 definitionsProperty: DEFINITIONS EQUALS LBRACKET definitionDefinition* (COMMA definitionDefinition)* RBRACKET;
84 definitionDefinition: DEFINITION STRING COLON LBRACE definitionBody RBRACE;
85 definitionBody: definitionProperty*;
86 definitionProperty: descriptionProperty|templateTypeProperty|templateProperty;
87 templateTypeProperty: TEMPLATE_TYPE EQUALS templateTypeValue;
88 templateTypeValue: STRING|PLAIN_TEXT|HTML|MARKUP|LATEX;
89 templateProperty: TEMPLATE EQUALS stringValue;
90 checksProperty: CHECKS EQUALS LBRACKET checkDefinition* (COMMA checkDefinition)* RBRACKET;
91 checkDefinition: CHECK STRING COLON LBRACE checkBody RBRACE;
92 checkBody: checkProperty*;
93 checkProperty: descriptionProperty|resultProperty|resultTypeProperty|actionProperty;
94 resultProperty: RESULT EQUALS booleanValue;

```

Рис. 2.5. Приклад опису завдання в текстовій нотації LTDL

На тлі досліджених інструментів запропонована мова LTDL концептуально виділяється своїм комплексним підходом до життєвого циклу PPET. В [149] DSL використовується для опису архітектурних компонентів навчальних систем (модулів LMS) і орієнтована на генерацію програмних компонентів LMS, тоді як LTDL функціонує в домені інженерної освіти та призначена для формалізації практичних навчальних завдань як параметризованих і виконуваних сутностей. Описана в [122] DSL функціонує в іншому домені (вивчення іноземних мов) та на іншому рівні абстракції і фокусується на моделюванні навчальних сценаріїв. Основною метою цієї DSL є формалізація дизайну навчальних активностей та подальша генерація програмного коду для освітніх застосунків, описано “що викладати”, але не описано, “яке завдання згенерувати конкретному студенту”. Незважаючи на активне застосування підходів модельно-орієнтованої інженерії (*Model-Driven Engineering*) у

системах технологічного навчання (*Technology-Enhanced Learning*), існуючі рішення орієнтовані на моделювання та генерацію навчальних систем [123], а не на формалізацію практичних завдань як обчислюваних об'єктів. LTDL усуває цей розрив, забезпечуючи формальний опис, генерацію, перевірку та персоналізацію практичних завдань у домені інженерної освіти. Універсальні формати серіалізації даних YAML та JSON також широко використовуються для опису задач і процесів у вигляді конфігураційних структур, проте не забезпечують вбудованих механізмів типізації, валідації та семантичного контролю. Їх конфігурації складні, важко відлагоджуються, не мають строгої типізації [124]. На відміну від цих форматів, LTDL містить нативні засоби параметризації та перевірки коректності. Це дозволяє генерувати унікальні валідні екземпляри практичних завдань для кожного здобувача освіти, забезпечуючи високий рівень варіативності та академічної доброчесності.

Окремо варто відзначити адаптованість запропонованої мови до сучасних методів автоматизації на базі AI. Завдяки строго структурованому синтаксису, LTDL значно знижує ймовірність помилок при генерації контенту LLM. Таким чином, розроблена DSL виступає інтеграційною ланкою, що об'єднує процеси генерації завдання, опису вимог до інфраструктури та формальної автоматичної перевірки, пропонуючи комплексне забезпечення, яке наразі відсутнє в існуючих рішеннях. Результат порівняння представлено в Таблиці 2.3.

Таблиця 2.3

Порівняння LTDL з існуючими засобами формалізації практичних завдань.

Критерій	IMS QTI [98]	Code Runner[103]	Haskell DSL [117,118]	VClassroom DSL [116]	SCQ Template Generator [120]	YAML/JSON описи	LTDL
Опис змісту завдання	+	-	+-	+-	+-	+-	+
Опис середовища виконання (VLE)	-	-	-	+	-	+-	+
Параметризація завдань	+-	+	+	-	+	-	+
Автоматичне розгортання середовища	-	-	-	+	-	-	+
Формалізація критеріїв перевірки	+	+	+-	-	+-	-	+

Візуальний режим	+	-	-	+	-	+-	+
Інтеграція з AI (GenAI)	-	-	+-	-	+-	+-	+
Підтримка персоналізації	+-	+-	+-	+-	+	-	+
Комплексне забезпечення (генерація + виконання + перевірка)	+-	-	-	-	-	-	+

2.4 Висновки до розділу II

Важливим результатом дослідження є перехід від неформального опису практичних завдань до їх формалізованого представлення. Розроблено функціональну модель PPET, яка описує процеси створення, розгортання, виконання та перевірки PPET і формалізує їх повний життєвий цикл як єдиний безперервний конвеєр із замкненим адаптивним циклом зворотного зв'язку, що формує уніфікований підхід до організації практичної підготовки з інженерних спеціальностей. Також запропоновано формальну модель PPET, яка подає завдання як структурований об'єкт із чітко визначеними компонентами: описом змісту, структурою етапів виконання, параметрами персоналізації, правилами перевірки та контекстом виконання та механізмами контролю коректності. Запропонована модель забезпечує формалізоване представлення внутрішньої структури завдання, що дозволяє забезпечити однозначність інтерпретації завдання інформаційною системою; створити основу для параметризації та унікалізації варіантів; реалізувати автоматичну перевірку коректності структури завдання та забезпечити інтеграцію із програмними компонентами системи персоналізованого навчання.

Розроблені функціональна та формальна моделі PPET виступають концептуальним і структурним базисом для створення домен-специфічної мови LTDL, оскільки відображають практичне завдання відповідно як процес (життєвий цикл) і як об'єкт (структура та компоненти). Такий підхід дозволив визначити вимоги до мови та забезпечити її здатність формалізовано описувати всі аспекти практичного завдання.

Також у цьому розділі представлено нову домен-специфічну мову LTDL, граматика якої, на відміну від існуючих, дозволяє формалізувати життєвий цикл РРЕТ і поєднати його педагогічні складові з технічними характеристиками середовища виконання, що забезпечує розгортання VLEs та перевірку результатів виконання завдань в автоматичному режимі. LTDL фокусується на мікрорівні завдань, глибокій параметризації, описі структури, залежностей, розгортанні VLE та гнучких перевірках. Розроблена мова може поєднуватися з генеративними AI для контрольованого процесу автоматичної генерації РРЕТ для інженерних спеціальностей, і дозволяє описувати складні навчальні сценарії з залежностями, перевірками та параметризацією.

Запропоновані моделі і мова реалізуються за допомогою методів автоматизованої генерації, автоматичного розгортання та перевірки, та дозволяють забезпечити узгодженість між процесами генерації завдань, їх виконання та оцінювання.

Результати досліджень, приведених в розділі, опубліковані в роботах [126, 150].

РОЗДІЛ ІІІ. МЕТОДИ ПЕРСОНАЛІЗОВАНОГО НАВЧАННЯ З ІНЖЕНЕРНИХ СПЕЦІАЛЬНОСТЕЙ

У межах даного дослідження удосконалено 4 взаємопов'язані методи автоматизації РРЕТ, які разом із відповідними програмними засобами їх реалізації є основою для інформаційного забезпечення процесів автоматизованої генерації РРЕТ з інженерних дисциплін, автоматичного розгортання необхідних VLE та перевірки результатів виконання завдань для набуття студентами сталих практичних навичок, що відображено в Таблиці 3.1.

Таблиця 3.1

Методи автоматизації процесів персоналізації практичного навчання.

Метод	Суть	Вхід	Вихід	Представлення	Місце в ІТ
Метод автоматизованої генерації РРЕТ	контрольованої змістовної AI-генерації формального опису РРЕТ	запит+контекст+проміт+приклад+граматика	персоналізоване “базове” завдання у вигляді LTDL-документу	формальна граматика LTDL у EBNF-нотації $G = (N, T, P, S)$, функція генерації	етап генерації
Метод параметризації РРЕТ	генерація варіантів завдання	персоналізоване “базове” завдання у вигляді LTDL-документу + набір параметрів	конкретний індивідуальний персоналізований варіант у текстово-графічному у вигляді	стохастичний генератор рядкових структур, функція параметризації	етап виконання
Метод автоматичного розгортання VLE	безпосереднє розгортання VLE (запуск і контроль, вимкнення)	LTDL + бекенд	ізольоване середовище S_i	відображення LTDL \rightarrow API backend	етап виконання
Метод автоматичної перевірки результатів виконання РРЕТ	поетапний контроль прогресу, перевірка успішності виконання	стан середовища $S_{env}(t)$, еталонні стани H_{target}^j	результат успішності: вектор правильності етапів $\{V_j\}$ та підсумкова оцінка V_{task}	булева верифікаційна функція	етап контролю

3.1 Архітектура інтелектуального асистента

Одним із ключових компонентів запропонованої інформаційної технології є персональний AI-асистент [150], який виступає інтелектуальним посередником (чатботом) [151] між студентом та DLE. Для формалізації поведінки цього інтелектуального асистента використано формальне визначення PPET та BDI-парадигму (Belief–Desire–Intention), що є добре дослідженим і популярним підходом до моделювання раціональних агентів [112-114, 152, 153].

Реалізація парадигми BDI в інтерпретації персоналізованого навчання дозволяє формалізувати процес прийняття рішень AI-асистентом, зокрема під час генерації PPET, адаптації складності та формування рекомендацій (рисунок 3.1)

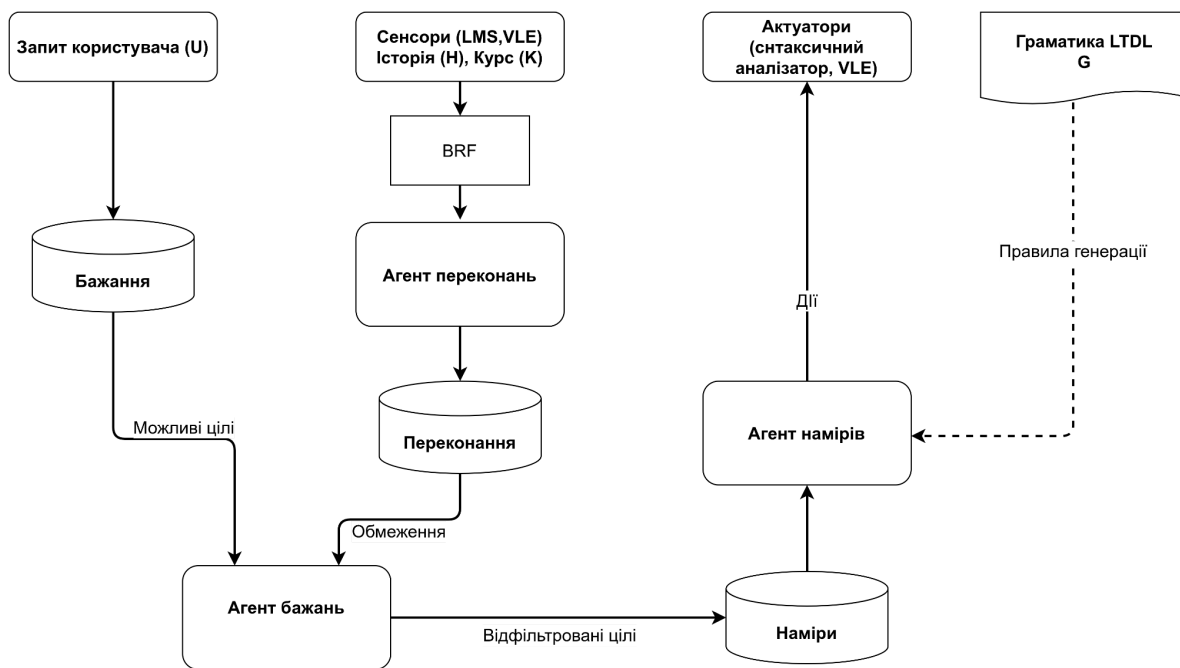


Рис. 3.1. Реалізація парадигми BDI в інтерпретації персоналізованого навчання

Запропонована архітектура інтелектуального асистента базується на мультиагентній системі AI агентів, що реалізує BDI-парадигму в інтерпретації персоналізованого навчання, і показана на рисунку 3.2. На відміну від класичних

підходів, модель поєднує формалізацію процесу прийняття рішень із генерацією формалізованих навчальних артефактів та їх виконанням у VLE.

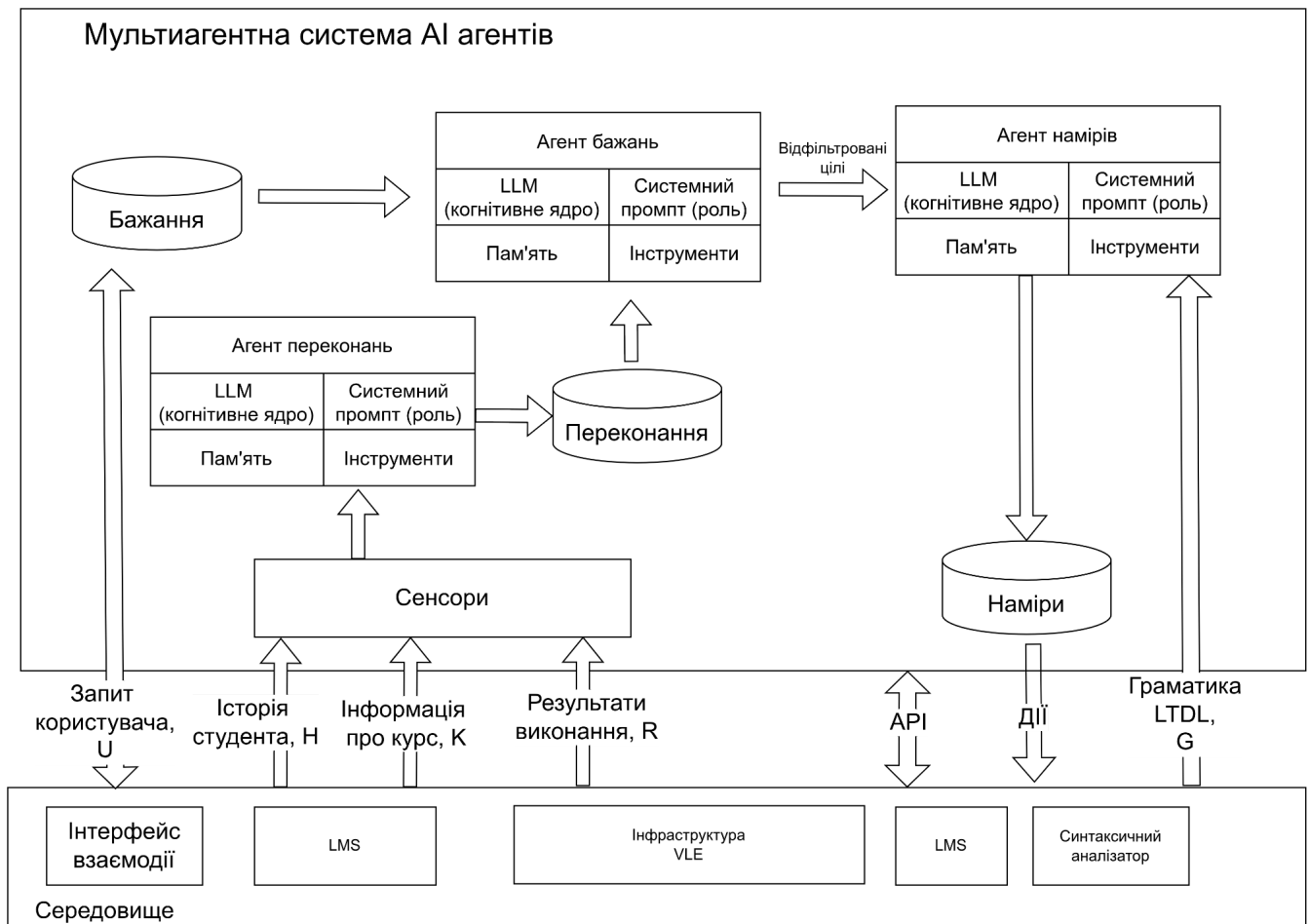


Рис. 3.2. Архітектура інтелектуального асистента

Інтерфейс взаємодії – забезпечує діалоговий канал між студентом і системою, приймає запити, відображає відповіді асистента, специфікації завдань, підказки та результати оцінювання;

Агент бажань – виконує первинне опрацювання запиту користувача: аналізує текстове повідомлення, виокремлює тему, бажаний рівень складності, можливі обмеження та наміри користувача, нормалізує вхідні дані та перетворює їх у структуроване внутрішнє подання, придатне для подальшої роботи асистента;

Агент переконань – формує початковий стан переконань на основі інформації

про курс та історію студента, і підтримує актуальність переконань, реалізуючи функцію перегляду переконань (Belief Revision Function – BRF).

Агент намірів – обирає наміри та визначає, який сценарій дій є доцільним для конкретного студента в поточному контексті, формує формальну специфікацію практичного завдання на основі обраного наміру, параметрів курсу, історії студента та інтерпретованого запиту, перетворюючи педагогічну мету на структурований артефакт виконання;

Інфраструктура VLE – виступає середовищем фактичного виконання РРЕТ, у якому студент взаємодіє з розгорнутими ресурсами, а система отримує об'єктивні результати для подальшого оновлення переконань агента. Відповідає за розгортання відповідного VLE, передає специфікацію завдання до VLE, збирає телеметрію та результати, а також обчислює показники успішності виконання;

LMS – задає предметну область, методичні вимоги, обмеження, правила формування завдань і критерії оцінювання, які впливають на поведінку асистента та зміст згенерованих завдань, накопичує дані про курс, історію попередніх спроб, вхідні параметри генерації та результати виконання, забезпечуючи збереження контексту та основу для персоналізації наступних рішень агента;

LLM – використовується як зовнішній інтелектуальний сервіс для інтерпретації запитів, уточнення контексту та синтезу фрагментів змісту завдання, однак його робота підпорядковується логіці рішень, сформованій у межах архітектури агента.

У запропонованому підході архітектура інтелектуального асистента базується на BDI-парадигмі, проте, на відміну від класичних реалізацій, вона має ієрархічну мультиагентну організацію з дворівневою інтерпретацією когнітивних станів. Перший рівень представлений мультиагентною системою як цілісним агентом, який функціонує по BDI і для якого визначено глобальні переконання, бажання та наміри, що відображають педагогічні цілі навчального процесу. Другий рівень складається з множини спеціалізованих агентів, які реалізують окремі функціональні аспекти системи. Ключовою відмінністю запропонованого підходу є узгодження когнітивних

станів між рівнями: внутрішні переконання, бажання та наміри окремих агентів не суперечать глобальним цілям системи, а формуються як їх локалізовані проекції. Це забезпечує цілісність поведінки системи та узгодженість прийняття рішень у процесі персоналізованого навчання. Крім того, на відміну від класичної BDI-моделі, інформаційний стан агента поділяється на дві складові: динамічні переконання про зовнішнє середовище, що змінюються в процесі взаємодії (наприклад, дані про прогрес студента), та апріорні переконання агентів, що визначають їх функціональну роль і залишаються незмінними протягом життєвого циклу агента. Такий поділ дозволяє забезпечити адаптивність системи без втрати стабільності її функціонального призначення.

Переконання (Beliefs – B) – це інформаційний стан агента. В запропонованому підході *Переконання* складаються з двох різних за своєю природою частин $B_t = \{B_{env}, B_a\}$,

де

- B_{env} - глобальні динамічні переконання про середовище. Це знання про те, що відбувається зовні: який рівень студента, які оцінки він отримав, який курс зараз проходить. Ці знання зберігаються в централізованому сховищі (сховище LMS та векторна база системи). Ці дані змінюються впродовж функціонування асистента за допомогою функції BRF;
- $B_a = \{B_b, B_d, B_i\}$ множина локальних (внутрішніх) переконань агентів. Це знання агента про самого себе та його роль, які зберігаються в системному промті. Ці знання не змінюються в межах життєвого циклу агента - агент не може (і не повинен) самостійно змінювати своє базове призначення під час роботи зі студентом.

Бажання (Desires – D) – це множина всіх можливих цілей, яких хоче досягти AI-асистент. В запропонованому підході *Бажання* складаються з:

- глобальні бажання системи D_{sys} – це стратегічні, педагогічні цілі AI-асистента.

Наприклад: "Студент повинен опанувати курс 'Комп'ютерні мережі'", "Студент має закрити прогалини в розумінні протоколу MQTT". Це закладено в архітектуру системи як її глобальна мета і виходить з навчального плану;

- локальні бажання агентів $D_a = \{D_b, D_d, D_i\}$ – це множина вузьких, функціональних цілей конкретних AI-агентів. Наприклад, бажання Агента намірів – «згенерувати синтаксично правильний код LTDL», яке зберігається в системному промті.

Намір (Intentions – I) – це те бажання, яке пройшло фільтрацію через *Переконання* і до виконання якого суб'єкт вже зобов'язався. Існують:

- глобальний намір системи I_t – це конкретний навчальний крок, обраний для студента в поточній ітерації t . Наприклад: "Система вирішила видати студенту лабораторну роботу №4 середньої складності з обмеженням часу 20 хвилин". Це затверджений план дій щодо студента, який формалізується набором конкретних дій;
- локальні наміри агента $I_a = (f_i, t_i)$ – це безпосереднє обчислювальне зобов'язання агента виконати певну функцію прямо зараз. Задається як пара функція-інструмент.

На кожному кроці агент отримує вхідний запит користувача $U_t \in \mathcal{U}$, де \mathcal{U} простір текстових запитів. Запит інтерпретується за допомогою оператора семантичного розбору $\tilde{U}_t = \phi(U_t)$ та інтегрується у стан переконань агента, впливаючи на подальший процес прийняття рішень. Оператор ϕ реалізує семантичну інтерпретацію запиту користувача та відображає його з простору текстових повідомлень у структуроване внутрішнє подання, що включає тему, бажаний рівень складності, обмеження та наміри користувача та є частиною функції сприйняття агента. Запит студента моделюється як частина динамічних переконань про стан середовища, але використовується як фактор формування множини бажань, тобто впливає на генерацію цілей, не будучи самостійною ціллю системи.

Таким чином, представимо глобальні динамічні переконання за формулою:

$$B_{env} = \{H_t, K, G_{ltdl}, \tilde{U}_t\},$$

де

H_t – історія студента,

K - контекст знань,

G_{ltdl} – граматика мови LTDL, яка задає правила генерації для агента намірів.

\tilde{U}_t – інтерпретований запит студента.

Історія студента представлена у вигляді кортежу трійок елементів:

$$H_t = \langle (\tilde{U}_1, A_1, R_1), (\tilde{U}_2, A_2, R_2), \dots, (\tilde{U}_n, A_n, R_n) \rangle,$$

де

\tilde{U}_n – інтерпретований запит користувача,

A_n – згенеровані практичні завдання,

R_n – результати виконання практичних завдань.

R_t – результат виконання завдання на етапі t в розгорнутому VLE представимо

$$R_t = \{P_t, T_t\},$$

де

$P_t \in [0,100]$ – оцінка виконання завдання у відсотках, значення якого розраховується за допомогою визначених вагових коефіцієнтів, помножених на результат виконання етапів завдання,

T_t – час, витрачений на виконання завдання у попередньо розгорнутому віртуальному навчальному середовищі.

Контекст знань, що визначає предметну область навчання, задається як опис курсу K і є множиною

$$K = \{C_{domain}, M, N\},$$

де

C_{domain} – визначає предметну область (наприклад, «структури даних», «мережеві протоколи», «системи баз даних», «криптографія»),

M – набір допустимих методів (наприклад, методичні рекомендації до виконання практичних, самостійних чи лабораторних робіт),

N – нотатки викладача, які повинні впливати на результат генерації завдання (наприклад, вказання конкретного бекенду для розгортання VLE).

Переконання агента формуються на основі історії навчання, результатів виконання та контексту знань: $B_{t+1} = BRF(B_t, H_t, R_t)$. Таким чином, B_t відображає актуальний стан знань про студента, його прогрес та контекст навчання, що забезпечує адаптивність та замкнений цикл навчання.

У запропонованій архітектурі функція BRF реалізується як композиція трьох операторів:

$$B_{t+1} = \mathcal{F}(\mathcal{E}(\mathcal{A}(B_t, H_t), R_t)),$$

де

\mathcal{A} – агрегація нових даних,

\mathcal{E} – оцінювання характеристик студента,

\mathcal{F} – фільтрація та механізм забування.

Така структура забезпечує інкрементальність оновлення, обмеження зростання історії та адаптивність до змін навчального контексту.

Множина кандидатних цілей агента формується та ранжується $D = \{D_{sys}, D_a\}$ на основі переконань: $D_t = options(B_t, \tilde{U}_t)$, де D_t є множиною потенційних цілей для поточного кроку прийняття рішення, що відповідають поточному рівню студента та його намірам.

Для забезпечення раціонального вибору наміру вводиться функція оцінювання, що визначає доцільність кожного кандидата з d множини D_t з урахуванням стану переконань та запиту користувача. Функція U – це функція

корисності, яка оцінює кожен кандидатний варіант з точки зору відповідності поточному стану студента та педагогічним цілям. У мультиагентній архітектурі функція корисності реалізується як розподілений механізм оцінювання, у якому окремі субагенти обчислюють часткові оцінки кандидатних рішень відповідно до своєї функціональної ролі. Зокрема, агент переконань оцінює відповідність кандидата поточному стану студента, агент бажань – відповідність запиту користувача, а агент намірів – виконуваність і коректність сформованого плану.

Нехай $d^* = \arg \max_{d \in D_t} U(d | B_t, \tilde{U}_t)$ - оптимальний кандидатний намір. Тоді намір

асистента формується як результат відображення $I_t = Plan(d^*, B_t)$, де функція *Plan* перетворює вибрану ціль у послідовність виконуваних дій. У такому випадку намір може бути представлений як

$$I_t = \langle I_a^1, I_a^2, \dots, I_a^n \rangle,$$

де

$I_a = (f_i, t_i)$ - локальний намір агента,

$f_i \in F$,

$t_i \in T$,

F – множина доступних функцій,

T – множина доступних інструментів.

Таким чином, глобальний намір є послідовністю атомарних обчислювальних дій, виконання яких приводить до реалізації обраного навчального сценарію. Результатом виконання наміру $I_t \rightarrow A_t$ є артефакт - формальний опис практичного завдання у вигляді $A_t = \{C, D, G, E\}$,

де

C – компонент контексту, який складається з предметної області (наприклад, «структури даних», «мережеві протоколи», «системи баз даних»,

«криптографія»), теми, що аналізується (наприклад, «реляційна база даних», «стек TCP/IP», «бінарне дерево пошуку»), опис змісту завдання, набору обмежень (наприклад, «ідеальні умови», «максимальна тривалість»), складності завдання.

D – компонент даних, представляє типізовані вхідні параметри з відповідними обмеженнями і визначається кортежем незмінних параметрів, спільних для всіх випадків завдання, параметризованих вхідних даних, які використовуються для параметризації екземпляра завдання, декларативні артефакти конфігурації, необхідні для переведення середовища в початковий стан.

G – компонент цілей, що визначає бажані результати згенерованого завдання та набір обов'язкових вимог, яким має відповідати рішення, щоб вважатися дійсним, кожна з яких є кортежем з предикатів, що описують обов'язковий стан середовища (*E*), структурне визначення артефакту, створеного в рамках цієї підділі, процедуру оцінки, що застосовується для визначення відповідності цільового стану або артефакту вимогам.

E – компонент специфікації середовища, що визначається як кортеж, який задає межі технології віртуалізації або виконання, стек і базову конфігурацію, обчислювальні обмеження для забезпечення справедливості та відтворюваності (наприклад, обмеження процесора, квоти пам'яті, пропускна здатність диска), операційні політики, включаючи правила доступу до мережі та необхідні канали телеметрії для спостереження.

Детальний опис компонентів *C*, *D*, *G*, *E* наведений в додатку Е.

У розробленій архітектурі запропоновано дворівневу доменно-орієнтовану інтерпретацію BDI для задач персоналізованого навчання, яка відрізняється від класичних підходів структурною декомпозицією когнітивних станів та інтеграцією з процесом генерації практичних завдань.

Множина переконань B_t розділена на апіорні (внутрішні) та динамічні

(середовищні). Апріорні переконання B_a імплементовані на рівні системних промтів кожного субагента, визначаючи його рольові обмеження та правила виведення. Натомість динамічне глобальне переконання системи про стан середовища та студента B_{env} формується як єдине централізоване онтологічне сховище. Окремі агенти не мають ізольованих локальних баз знань про студента, що зменшує ймовірність виникнення епістемічних конфліктів між ними.

Стан системи, що визначає глобальні бажання D_{sys} , спрямований на досягнення педагогічних цілей курсу, тоді як локальні бажання суб-агентів D_a обмежені їхніми вузькоспеціалізованими когнітивними завданнями. Агенти генерують локальні цілі на основі підмножини глобальних переконань, а зміна переконань асистента відбувається через єдину функцію перегляду переконань BRF за результатами валідації практичного завдання.

Запропоновано формалізацію результату діяльності агента у вигляді навчального артефакту $A_t = \{C, D, G, E\}$, що забезпечує уніфіковане представлення контексту, даних, цілей та параметрів середовища виконання PPET. На відміну від класичних BDI-моделей, у яких наміри розглядаються як внутрішні плани дій, у запропонованій архітектурі встановлено відображення $I_t \rightarrow A_t$, що дозволяє інтерпретувати намір як формалізований освітній результат, придатний до автоматизованого виконання у VLE. Додатково архітектура включає замкнений цикл адаптації, у якому результати виконання завдань у середовищі VLE використовуються для оновлення переконань агента через функцію перегляду переконань, що забезпечує контекстно-залежну персоналізацію та безперервне уточнення навчальної траєкторії студента.

3.2 Метод автоматизованої генерації PPET

GenAI має значний потенціал у сфері персоналізованого навчання, особливо у контексті автоматизації створення, адаптації та перевірки практичних завдань. Проте ефективне застосування потребує розробки архітектурних рішень, що поєднують

гнучкість генерації з формальною структурою DSL, що забезпечує контроль коректності результатів. Саме така інтеграція дозволить досягти балансу між автоматизацією, персоналізацією та академічною доброчесністю.

В межах дослідження пропонується модифікувати метод автоматизованої генерації завдань [154-156] шляхом поєднання методів штучного інтелекту з механізмами DSL. Методи AI використовуються для змістовної генерації чи регенерації завдання, нова розроблена мова LTDL (описана в розділі II) – для формального визначення структури завдання, його перевірки і автоматичного виправлення синтаксичних і лексичних помилок, параметризації (закладається можливість подальшої унікалізації завдання виданого конкретному студенту). Формальна граматики мови LTDL забезпечує єдиний механізм опису, відтворення та контролю освітніх сценаріїв, тоді як використання GenAI має локальний та контрольований характер і обмежується лише змістовою складовою і функцією персоналізації змісту завдання. Такий гібридний підхід (комбінація GenAI та DSL в освіті) є передовим науковим напрямком у галузі комп'ютерних наук та освітніх технологій [125] і забезпечує вищу точність генерації, ніж одноосібне використання LLM, що буде експериментально підтверджено в наступному розділі дисертації.

Процес генерації починається із запиту від зовнішнього учасника. Запит повинен містити інформацію про тему завдання (за бажанням – рівень складності та додаткову інформацію з деталізацією теми). Вхідний запит об'єднується в потоці AI з додатковою інформацією, яка представляє контекст для GenAI. Контекст GenAI складається з файлів граматики LTDL в нотації EBNF, підготовлених прикладів з бажаною структурою LTDL та системного запиту для створення вихідних даних. Запропонований метод надання AI правил граматики є науково обґрунтованою та передовою технікою промт-інжинірингу. В [125] для досягнення точності генерації використовується техніка промт-інжинірингу, заснована на метамоделях.

Після отримання згенерованого опису завдання у вигляді структурованого документа LTDL AI-агент використовує спеціальний інструмент для його збереження на сервері та пересилає його до LTDL-парсера. Парсер виконує

лексичний аналіз і синтаксичний аналіз для перевірки згенерованої структури на відповідність правилам граматики. У разі виникнення синтаксичних помилок неправильний документ із переліком помилок надсилається до моделі AI для повторного генерування. Кількість ітерацій налаштовується на рівні агента, експериментальним шляхом визначена кількість дорівнює 5, після якої результати генерації не змінюються. Успішна генерація призводить до створення валідного файлу LTDL з описом PPET. Цей файл можна зберегти в сховищі LMS для подальшого аналізу та використання. PPET можна переглянути для отримання додаткових деталей та виправити за допомогою графічного режиму в LTDLEditor. Оцінку PPET можна остаточно виконати після розгортання завдання у VLE.

У даному дослідженні задача генерації опису PPET формалізується як задача синтезу структури у домен-специфічній мові LTDL.

Ведемо $L(G)$ як множину коректних описів завдань за допомогою формальної граматики G . GenAI (M) формує можливий опис завдання $L = M(C)$ на основі контексту C . Оскільки L може не належати $L(G)$, вводиться функція валідації $V: \Sigma^* \rightarrow \{0, 1\}$, яка перевіряє відповідність опису правилам граматики LTDL.

Таким чином задача формулюється як задача обмеженої генерації

$$\text{maximize } P_M(L \vee C), \text{ subject } \rightarrow L \in L(G)$$

GenAI використовується для формування текстових інструкцій та інших семантичних компонентів елементів завдання, тоді як структура завдання та допустимі зв'язки між його елементами визначаються формальною граматиною LTDL. Це забезпечує поєднання гарантованої структурної коректності та динамічної персоналізації змісту.

Схема методу контрольованої генерації опису завдання, що поєднує генерацію змісту генеративною моделлю AI, синтаксичну валідацію за граматиною LTDL та ітеративне повторне генерування у разі порушення формальних обмежень, представлена на рисунку 3.3.

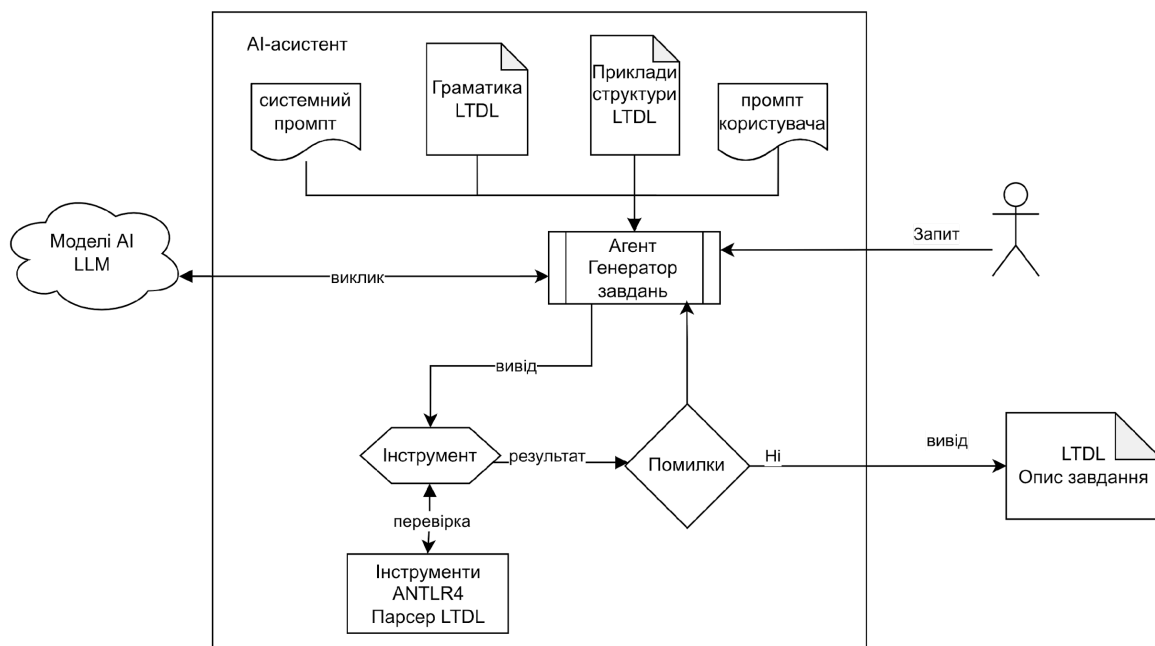


Рис. 3.3. Схема методу контрольованої генерації опису завдання.

У процесі автоматизованої генерації PPET важливим є не лише створення унікальних варіантів, але і контроль їх складності. Відсутність формального опису складності призводить до низки проблем, таких як неможливість адаптації завдань під рівень підготовки студента. Студенти з різним рівнем базових знань потребують відмінних траєкторій навчання: надто прості завдання не формують компетентностей, а надто складні – знижують мотивацію та підвищують ризик звернення до нечесних методів виконання з порушенням академічної доброчесності, складність автоматичного масштабування процесу персоналізації та обмежена ефективність AI-генерації без контролю складності: GenAI може продукувати занадто прості або надмірно складні завдання, якщо відсутній параметр, який дозволяє алгоритмічно керувати глибиною та варіативністю завдання. В якості такого параметру пропонується ввести **індекс структурної комплексності завдання** (Task Structural Complexity Index – TSCI) як формалізований критерій порівняння та адаптивного контролю рівня складності PPET. Метрика дозволяє узгодити процеси AI-генерації змісту, оскільки надає кількісну оцінку структурної складності завдання, і виступає ключовим елементом управління динамічними навчальними траєкторіями та забезпечує баланс між доступністю завдань для

студента і вимогами до формування професійних компетентностей. TSCI відіграє ключову роль у функціонуванні розробленої інформаційної технології персоналізованого практичного навчання, забезпечує кількісно вимірювальний механізм контролю складності навчальних завдань у процесі їх автоматизованої генерації, параметризації та перевірки:

1. TSCI дозволяє здійснити адаптивний підбір завдань відповідно до рівня підготовки студента. Оскільки студенти мають різні стартові знання і швидкість засвоєння матеріалу, одні й ті самі завдання не можуть бути ефективними для всіх. Використання TSCI дозволяє системі автоматично формувати завдання потрібного рівня складності, підвищуючи точність персоналізації і зменшуючи ризик когнітивного перевантаження або, навпаки, недостатнього навчального виклику.

2. TSCI завдання забезпечує масштабованість процесу персоналізації. Формальна метрика дозволяє автоматизувати процес: інтелектуальні модулі генерації та параметризації можуть самостійно підвищувати або знижувати складність шляхом зміни структури графа завдання (зокрема, кількості гілок та глибини критичного шляху), не залучаючи викладача до рутинних дій.

3. TSCI забезпечує контрольовану інтеграцію GenAI. Неврегульоване використання моделей типу LLM призводить до непередбачуваності результатів: завдання можуть бути надто простими, неточними або надмірно складними. Визначення цільового значення TSCI дає змогу використовувати LLM як семантичний генератор змісту, але під контролем формальної структури та цільового рівня складності, що забезпечує відтворюваність і стабільність результатів.

4. TSCI дозволяє зберігати порівнянність та баланс між завданнями в курсі. Це особливо важливо для оцінювання навчальних результатів та забезпечення академічної доброчесності: студенти мають отримувати завдання різні за наповненням, але рівноцінні за складністю, що унеможливорює ситуації, коли окремі студенти виконують суттєво простіші або складніші варіанти, ніж інші.

5. TSCI є необхідним елементом динамічного управління навчальною траєкторією. Значення індексу може змінюватися під час навчання: наприклад, після

успішного завершення кількох завдань система може автоматично запропонувати складніше завдання, що сприяє поступовому розвитку компетентностей, а не їх випадковому накопиченню.

Розрахунок індексу TSCI базується на теорії графів і використовує цикломатичну складність як основу [157]. Він математично обґрунтований і машиночитаний, що дозволяє AI-асистенту динамічно маніпулювати складністю.

PPET, описане на LTDL, визначається як орієнтований ациклічний граф (DAG)

$$T = (V, E, K),$$

де:

V (Вершини) - множина всіх етапів, які студент має виконати (усі блоки stage).

|V| – це загальна кількість етапів.

E (Ребра) - множина всіх жорстких залежностей між етапами (усі зв'язки depend). |E| – це загальна кількість залежностей.

K (Компоненти) - множина паралельних шляхів виконання (усі блоки track).

|K| – це загальна кількість треків.

Базовий розрахунок складності графу:

$$TSCI = |E| - |V| + 2|K|$$

Приклад розрахунку TSCI наведено в Таблиці 3.2.

Таблиця 3.2

Приклад розрахунку індексу TSCI

Опис завдання	Приклад розрахунку індексу TSCI
Просте лінійне завдання (базовий рівень складності).	1 track (K=1), 3 stage (V=3), 2 depend (E=2) TSCI = 2 - 3 + 2(1) = 1
Просте паралельне завдання (завдання складніше, бо вимагає паралельного мислення).	2 track (K=2), 3 stage (V=3), 1 depend (E=1) TSCI = 1 - 3 + 2(2) = 2
Складне злиття (завдання значно складніше, бо треки сходяться в одній точці).	2 track (K=2), 5 stage (V=5), 5 depend (E=5) TSCI = 5 - 5 + 2(2) = 4

Механізм керування складністю для AI є чіткою інструкцією. По запиті студента "дай мені складніше/простіше завдання" AI-асистент проводить визначені

маніпуляції з графом. Приклади механізму керування TSCI (збільшення або зменшення) наведені в Таблиці 3.3.

Таблиця 3.3

Приклад механізму керування TSCI

Механізми збільшення складності	Механізми зменшення складності
Збільшити Заплутаність (\uparrow TSCI через \uparrow E). Інструкція для AI: Додати нові ребра depend між раніше незалежними етапами stage. Це створює "вузькі місця" і злиття шляхів, що збільшує TSCI.	Зменшити Заплутаність (\downarrow TSCI через \downarrow E). Інструкція для AI: Видалити ребра depend. Це "розблоковує" етапи, дозволяючи студенту виконувати їх у довільному, більш простому порядку.
Збільшити Паралелізм (\uparrow TSCI через \uparrow K). Інструкція для AI: Розбити один довгий track на два паралельні track. Це змушує студента утримувати в пам'яті кілька контекстів одночасно.	Зменшити Паралелізм (\downarrow TSCI через \downarrow K). Інструкція для AI: Злити два паралельні track в один послідовний. Це зменшує когнітивне навантаження.

Обмеження методу контрольованої AI-генерації.

Запропонований метод має низку обмежень, що необхідно враховувати при його практичному застосуванні в DLE:

- Залежність якості змістовної генерації від моделі AI. Хоча LTDL забезпечує структурний контроль, семантична якість змісту залишається залежною від можливостей конкретної LLM-моделі;
- Необхідність підтримки граматики LTDL у процесі масштабування. Розширення множини допустимих структур потребує редагування граматики, що передбачає участь фахівця, який володіє навичками формального моделювання;
- Ймовірність надмірної формалізації завдань. Оскільки LTDL вимагає чітких структурних блоків, створені завдання можуть мати більш стандартизовану форму, ніж завдання, сформульовані викладачем вручну. Це підвищує відтворюваність, але може зменшувати варіативність стилістичних формулювань;
- Вимоги до обчислювальних ресурсів. Генерація завдань може бути ресурсомісткою при масовому використанні у великих групах студентів. Це

особливо критично для навчальних закладів, які застосовують локальні або ізольовані серверні середовища з обмеженою продуктивністю;

- Потреба у валідації семантичної коректності складних завдань. Хоча структурна перевірка гарантує правильність форми, семантична логіка, наприклад, відповідність навчальній меті або коректність прикладів, потребує періодичної експертної перевірки для складних міждисциплінарних сценаріїв.

3.3 Метод параметризації PPET

Під час формування контрольного практичного завдання для студента необхідно забезпечити унікальність конкретного варіанта завдання при збереженні однакового рівня дидактичної складності для всіх студентів. Саме за рахунок фінальної параметризації на етапі виконання збільшується варіативність і унікальність завдань – важливий фактор у масштабній онлайн-освіті та оцінюванні. Параметризація дозволяє автоматично генерувати різні варіанти одного і того ж завдання, змінюючи певні його елементи. Для цього треба виділити частини, які потребують параметризації для кожного етапу: імена змінних, назви файлів або директорій, ім'я користувача, функцій, порядок параметрів, тощо. Такий підхід є сучасним і затребуваним [158, 159].

Параметри завдання розглядаються як структурні токенні патерни (*structural token patterns*), що описують текстові елементи середовища виконання завдання (назви директорій, імена змінних, ідентифікатори користувачів, назви функцій тощо), тому для параметризації застосовано контрольований стохастичний генератор рядкових структур, що формує допустимі рядки відповідно до правил формальної граматики та заданого ймовірнісного розподілу [160, 161]. Допустимість синтаксису кожного патерну гарантується формальною граматиною G мови LTDL. Простір допустимих значень параметра позначається як Ω . Його елементи формуються відповідно до синтаксичних правил мови LTDL, що гарантує коректність рядкових структур параметрів.

Тоді вибір параметра здійснюється як випадкова величина:

$$p_i \sim P_{param}(\cdot \vee S, C),$$

де $P_{param}(\cdot \vee S, C): \Omega \rightarrow [0, 1]$ - умовний розподіл на множині допустимих значень параметра, що залежить від профілю студента S (його рівень підготовки: початковий, середній, високий) та контексту завдання C .

Значення параметра p_i вибирається випадковим чином із розподілу P_{param} , який визначено на множині допустимих значень параметра та залежить від профілю студента S і контексту навчальної задачі C .

Параметрів існує кілька, тому вводимо вектор параметрів $\theta = (p_1, p_2, \dots, p_n)$, де $p_i \in \Omega$, а генерація кожного параметра виконується за тим самим правилом.

Перед прийняттям згенерованого параметра виконується синтаксична перевірка та семантична валідація. Перевіряється відсутність заборонених символів, допустима довжина рядка та унікальність значення параметра в межах поточної сесії генерації. Якщо згенероване значення параметра не проходить синтаксичну або семантичну валідацію, виконується повторна генерація параметра згідно того ж розподілу P_{param} до отримання допустимого значення.

Випадковість є керованою: генерація параметра відбувається в межах допустимого простору, але різноманітність варіантів контролюється навчальними обмеженнями. Структурні властивості завдання визначаються граматикою, а варіативність – стохастичною моделлю параметрів, що гарантує баланс між персоналізацією і контрольованістю. Таким чином, параметризація є випадковою на рівні вибору конкретного значення, але детермінованою на рівні структури та типу параметра. Це забезпечує баланс між унікальністю та контрольованістю.

Параметризоване практичне завдання не є статичним об'єктом, а є результатом виконання генеруючої функції, що застосовує унікальні параметри до шаблону завдання, описаного мовою LTDL, і генерує унікальний екземпляр завдання на основі унікальних параметрів студента.

Нехай T_{base} – шаблон практичного завдання, описаний мовою LTDL. Тоді

параметризоване завдання визначається як

$$P = F(T_{base}, \theta),$$

де $\theta = (p_1, p_2, \dots, p_r)$ – вектор параметрів,

F – функція параметризації, що виконує підстановку параметрів у шаблон завдання.

Це забезпечує контрольовану стохастичність параметризації: з одного боку, створюється велика кількість унікальних варіантів завдань, з іншого – зберігається відповідність навчальним цілям, оскільки вибір параметра обмежений структурними правилами LTDL та семантичними обмеженнями навчального сценарію. Саме формальна граMATика LTDL виступає механізмом обмеження простору параметризації, забезпечуючи коректність структури завдання незалежно від кількості згенерованих варіантів.

Оскільки значення параметрів p_i належать допустимому простору Ω , а шаблон завдання T_{base} є коректним LTDL-описом, результат параметризації також залишається коректним описом завдання:

$$P \in L(G).$$

В дослідженні удосконалено метод параметризації PPET за рахунок поєднання стохастичної генерації параметрів із формальними обмеженнями мови LTDL, що дозволяє, на відміну від існуючих, формувати великий простір унікальних варіантів завдань при збереженні їх структурної коректності.

3.4 Метод автоматичного розгортання VLE

Для забезпечення академічної доброчесності, моніторингу процесу та усунення можливостей реверс-інжинірингу застосовано механізм ізольованих та одноразових VLE. Вимоги до VLE сформульовані в першому розділі. Формальна граMATика LTDL описує конфігурацію VLE, яка транслюється у команди на рівні backend-сервера. Вхідними даними цього методу є набір інструкцій для розгортання індивідуального VLE, які зберігаються у вигляді LTDL-документа. Вихідними даними методу автоматичного розгортання є статус запущеного VLE та його

параметри.

Відображення LTDL-опису у сценарій розгортання подається як функціональне відображення:

$$\tau: L(G) \rightarrow B,$$

де $L(G)$ - множина коректних описів завдань, що породжуються граматиною LTDL; B – множина конфігураційних інструкцій для контейнеризації або віртуалізації VLE.

Будь-яка зміна опису завдання автоматично породжує відповідні зміни у середовищі без ручного втручання.

Відображення τ реалізується у вигляді послідовності етапів аналізу LTDL-документа, трансляції конфігурації VLE та виконання сценарію розгортання. Послідовність виконання методу автоматичного розгортання VLE представлена у вигляді алгоритмічної схеми на рисунку 3.4.

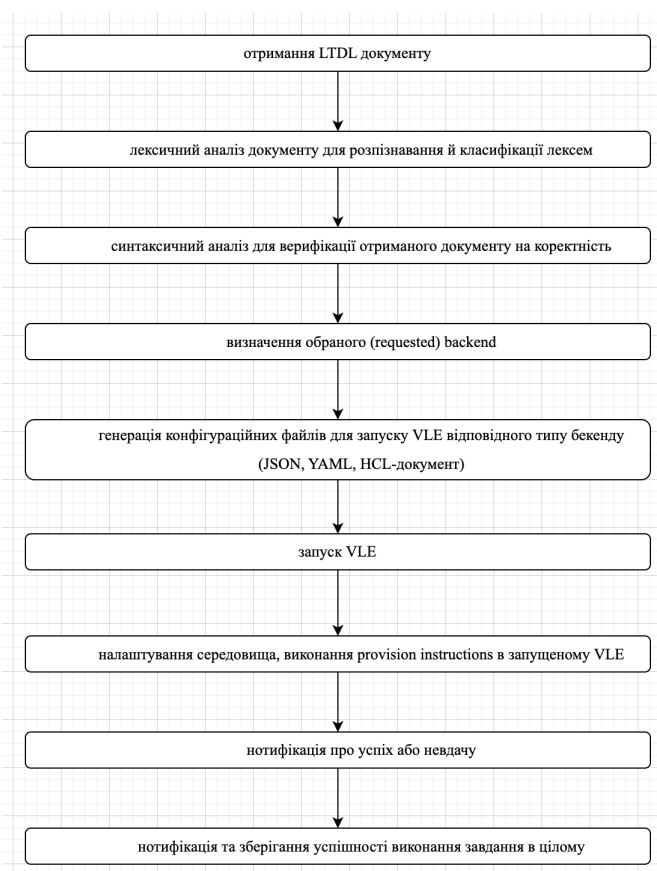


Рис. 3.4. Послідовність виконання методу автоматичного розгортання.

Загальний час розгортання VLE використовується для оцінки масштабованості системи при різній кількості студентів і оцінюється як:

$$T_{total} = T_{parse} + T_{translate} + T_{provision} + T_{validation}, \quad (3.1)$$

де

T_{parse} – час аналізу LTDL-опису;

$T_{translate}$ – генерація конфігураційних файлів;

$T_{provision}$ – час створення контейнера/VM;

$T_{validation}$ – час первинної перевірки готовності середовища.

В дослідженні удосконалено метод автоматичного розгортання ізольованих VLE на основі LTDL-опису завдання, який, на відміну від існуючих підходів до організації VLE, забезпечує автоматичну трансляцію формального опису завдання у конфігурацію середовища виконання.

3.5 Метод автоматичної перевірки результатів виконання PPET

В даному дослідженні пропонується удосконалити метод автоматичної перевірки результатів виконання PPET шляхом формалізації стану VLE у вигляді впорядкованого вектора артефактів, що дозволяє визначати відповідність поточного стану VLE цільовому стану виконання етапів завдання.

Вхідними даними цього етапу є LTDL документ зі списком команд для перевірки цілей кожного етапу завдання та еталонний статус VLE для кожного етапу завдання. Схема використання методу автоматичної перевірки результатів виконання PPET зображена на рисунку 3.5.

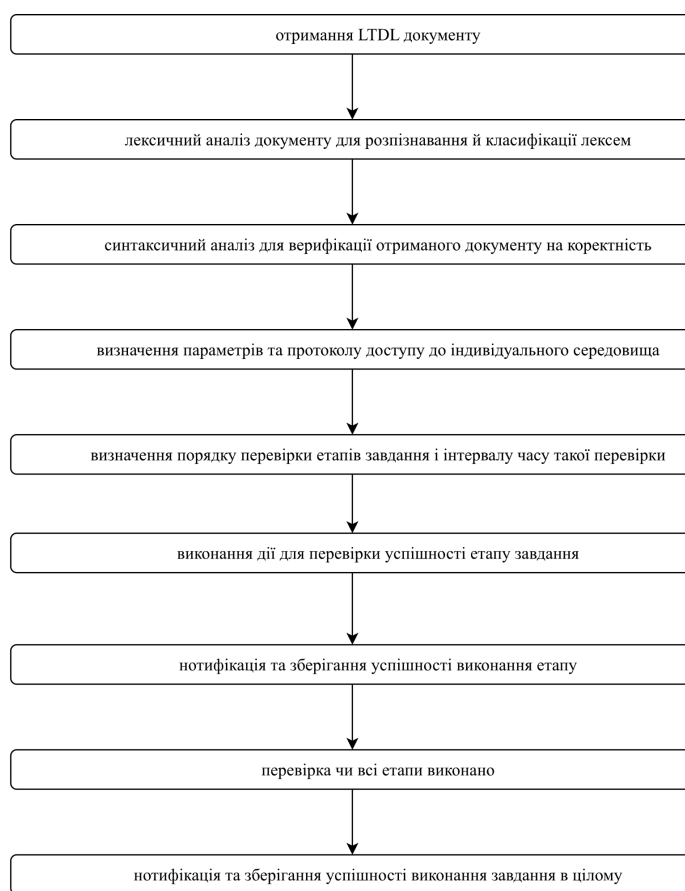


Рис. 3.5. Послідовність виконання етапів методу автоматичної перевірки результатів

Результатом виконання дії перевірки може бути значення true/false, конкретне очікуване значення поля чи параметру або певний стан VLE, який важко детермінувати. В такому випадку пропонується використання hash-функції (md5, sha тощо) для опису потрібного стану. Конкретний стан, який є результатом роботи однієї або кількох дій (команд) фіксується і хешується за допомогою попередньо обраної hash-функції. В якості цілі такого етапу завдання треба вказати значення hash-функції для фіксації потрібного стану середовища.

Стан VLE S_{env} у момент часу t можна визначити як впорядкований набір (вектор) ключових артефактів, що підлягають перевірці:

$$S_{env}(t) = a_1(t), a_2(t), \dots, a_n(t),$$

де a_k – це стан конкретного артефакту (наприклад, вміст файлу, права доступу, статус процесу, наявність каталогу).

Метою j -го етапу завдання є досягнення цільового стану $S_{target,j}$. Процедура автоматичної поетапної перевірки V_j є булевою функцією, що порівнює поточний стан з еталонним:

$$V_j = \begin{cases} 1, \text{ якщо } H(S_{env}(t)) = H_{target,j} \\ 0, \text{ інакше} \end{cases}$$

де

H – хеш-функція, що обчислюється над конкатенацією всіх артефактів a_k , які входять до S_{env} ;

$H_{target,j}$ – заздалегідь обчислений еталонний хеш для цільового стану j -го етапу, що зберігається у LTDL-документі.

Для кожного етапу функція набуває значення 1 у випадку відповідності поточного стану VLE еталонному стану, та 0 – у протилежному випадку.

Так як завдання складається з m кроків / етапів, для узагальненої кількісної оцінки виконання практичного завдання вводиться інтегральний показник V_{task} :

$$V_{task} = \sum_{j=1}^m w_j \cdot V_j,$$

де

m – кількість етапів у завданні;

V_j – індикаторна функція коректності виконання j -го етапу;

w_j – вага j -го етапу.

Так як в реальних РРЕТ завданнях етапи мають різну важливість, використано зважене оцінювання. Тобто студент отримує частку виконаної роботи відповідно до важливості етапів. Дане визначення дозволяє коректно відображати вплив ключових кроків на загальний результат і слугує основою для автоматизованої перевірки

завдань у VLE з підтримкою поетапної валідації.

Вихідними даними методу автоматичної перевірки є статус успішності виконання завдання по кожному етапу та кількісна оцінка виконання всього завдання. Вихідні дані передаються на подальшу обробку та зберігання в LMS.

У запропонованому підході перевірка сталості навичок реалізується через параметризацію практичних завдань та автоматизовану верифікацію результатів їх виконання. Навичка вважається сталою, якщо студент демонструє коректне виконання задачі на множині варіацій задачі. Для кількісної оцінки сталості практичних навичок запропоновано метрику **Sustainability Index (SI)**. Введемо

$S = \{t_1, t_2, \dots, t_n\}$ – множина варіацій практичного завдання (PPET-інстанси),
 $r_i \in [0, 1]$ – інтегральний результат виконання задачі t_i , який може включати коректність, повноту, якість рішення (визначається автоматично),
 $a_i \in [0, 1]$ – коефіцієнт автономності (1 – без допомоги, <1 – з підказками/AI),
 w_i – вага задачі (складність/важливість)

Тоді SI визначається як зважена сума результатів виконання множини параметризованих варіацій практичного завдання з урахуванням коректності виконання, автономності студента та складності задач.

$$SI = \frac{\sum_{i=1}^n w_i * r_i * a_i}{\sum_{i=1}^n w_i}$$

Запропонована метрика дозволяє формалізувати поняття сталості практичних навичок та може бути використана для оцінювання ефективності персоналізованих практичних завдань у подальших дослідженнях. На відміну від традиційного підходу, який базується на оцінці окремого результату виконання завдання, запропонований алгоритм передбачає аналіз множини параметризованих варіацій практичного завдання. На основі агрегованих даних обчислюється показник сталості навички, який використовується як доповнення до традиційної оцінки результату виконання завдання. Це дозволяє перейти від оцінювання окремих відповідей до

оцінювання сформованості практичної навички як стійкої здатності до виконання задачі в різних умовах. В запропонованому методі автоматичної перевірки на кожному кроці виконується автоматизована верифікація результату виконання, в ході якої визначається коректність рішення, а також фіксуються додаткові параметри виконання, зокрема рівень автономності (наявність або відсутність підказок, використання допоміжних інструментів) та характеристики VLE. Запропонована метрика відображає сталість практичної навички через узагальнення результатів виконання множини параметризованих варіацій завдання, що відповідає визначенню сталості як здатності коректно виконувати задачу за різних умов. Врахування коефіцієнта автономності дозволяє оцінити ступінь самостійності виконання, а вагові коефіцієнти — вплив складності окремих варіацій задачі на загальний результат. Таким чином, значення SI інтерпретується як інтегральна оцінка стійкості навички до зміни умов виконання.

Додатково може враховуватись часовий фактор, що характеризує стійкість навички у часі. Реалізація та експериментальна валідація запропонованої метрики із урахуванням часового фактору становить перспективний напрям подальших досліджень.

3.6 Варіанти використання методів

Діаграми варіантів використання методів автоматизації персоналізованого навчання дозволяють представити основні сценарії використання запропонованих методів, визначити ролі учасників процесу навчання та окреслити ключові функції, пов'язані з генерацією, параметризацією, розгортанням VLE та автоматичною перевіркою виконання PRET. Нижче наведено діаграми варіантів використання, що ілюструють основні сценарії роботи.

На рисунку 3.6 представлено Use Case діаграму попереднього налаштування мультиагентної системи з боку надавача освітніх послуг. Основним актором є адміністратор, який виконує підготовку системи до використання в навчальному процесі. Діаграма відображає ключові дії адміністратора, зокрема: встановлення та налаштування мультиагентної системи, встановлення і конфігурацію модуля PAIA, а

також розгортання підсистем. Взаємодія з усіма варіантами використання є централізованою, що відображає керований характер початкового етапу впровадження. Представлений набір варіантів використання формує базовий операційний контур розгортання системи, забезпечуючи узгоджене функціонування її компонентів у межах DLE та готовність до генерації, розгортання і перевірки PPET.

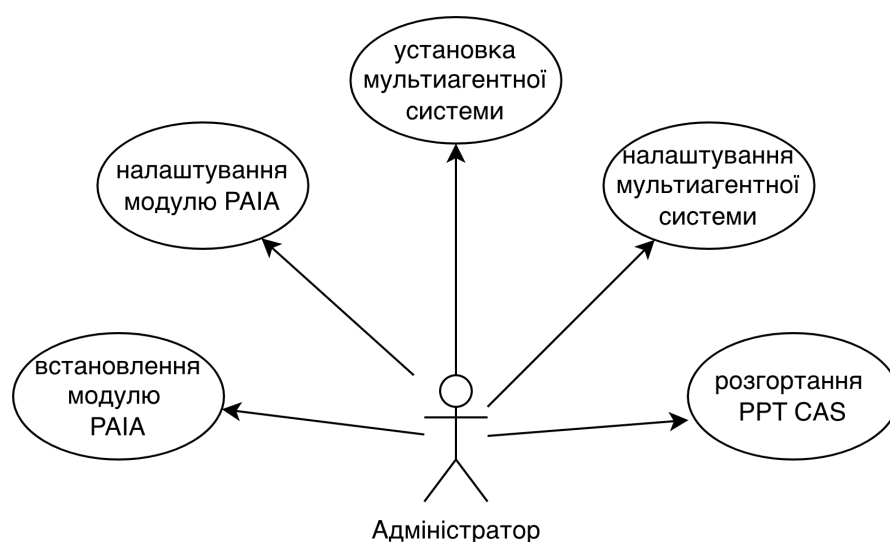


Рис. 3.6. Use Case діаграма попереднього налаштування мультиагентної системи адміністратором.

Варіанти використання методів викладачем починається після попереднього розгортання всіх компонентів. На рисунку 3.7 представлено Use Case діаграму налаштування курсу в LMS. Основним актором є викладач, який виконує підготовку навчального курсу до подальшого використання. Діаграма відображає ключові дії викладача, зокрема створення курсу, додавання модуля PAIA та його налаштування. Взаємодія з варіантами використання відповідає логіці послідовного формування VLE. Представлений набір варіантів забезпечує ініціалізацію курсу і його готовність до інтеграції інструментів генерації та перевірки PPET.

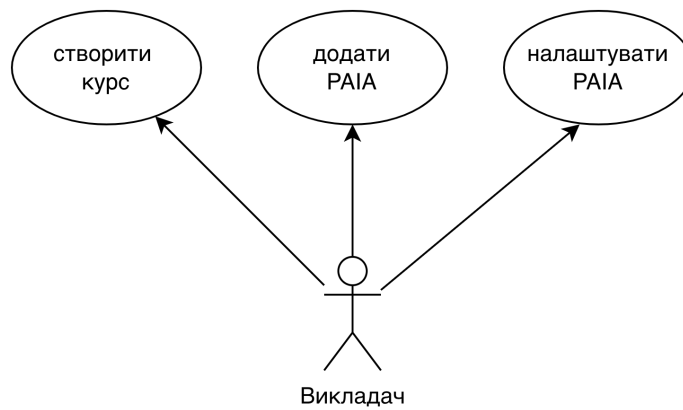


Рис. 3.7. Use Case діаграма попереднього налаштування курсу викладачем

На рисунку 3.8 представлено Use Case діаграму створення практичного іспиту в LMS. Основним актором також виступає викладач, який формує та публікує практичні завдання для студентів. Діаграма відображає основні дії, що включають створення завдання, його перегляд і редагування, запуск у VLE, а також створення відповідної активності в LMS Moodle. Така структура відображає повний цикл підготовки практичного іспиту – від формування змісту до його інтеграції в освітню DLE.

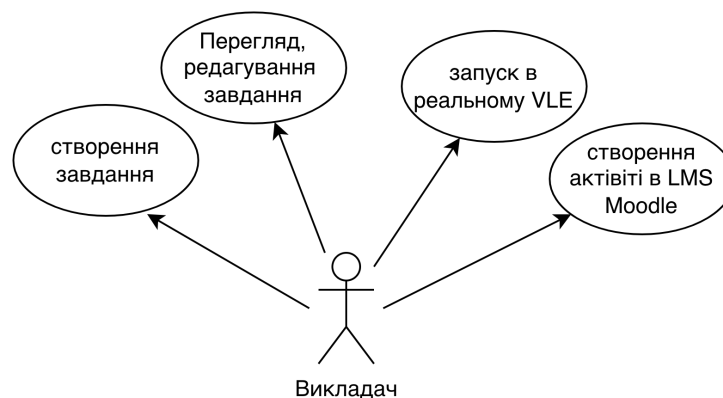


Рис. 3.8. Use Case діаграма створення практичного іспиту в LMS.

Рисунок 3.9 узагальнює потоки даних у запропонованій IT, відображаючи повний цикл створення PPET викладачем – від генерації та валідації LTDL-опису до його візуалізації, збереження та розгортання у VLE через LMS. Це демонструє

узгоджену взаємодію компонентів системи та забезпечує контрольованість і автоматизацію процесу формування практичних завдань.

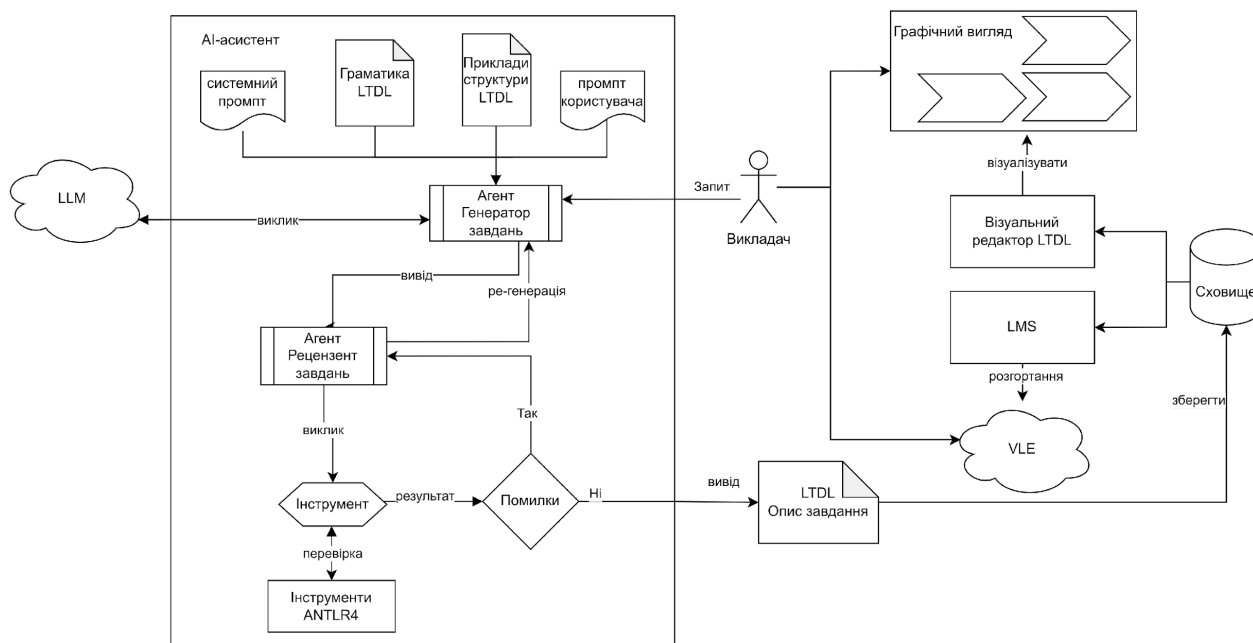


Рис. 3.9. Потіки даних при створенні PPET викладачем.

Варіанти використання методів студентом відрізняється: вони можуть генерувати завдання та виконувати їх у власному VLE з автоматичною перевіркою, але без доступу до повного опису завдання в форматі LTDL. Запропонований метод дозволяє як самостійну генерацію студентом необмеженої кількості PPET, яку можна використовувати для досягнення індивідуальних навчальних цілей (більш глибоке вивчення конкретної теми, повторне вивчення теми для кращого розуміння, вивчення додаткових тем або виконання завдань різної складності), так і генерацію фінального параметризованого практичного іспиту для отримання оцінки.

На рисунку 3.10 представлено Use Case діаграму набуття студентом практичних навичок з певної теми. Така структура забезпечує гнучкість процесу, у якому студент виступає активним суб'єктом керування, та підтримує індивідуальну траєкторію навчання.

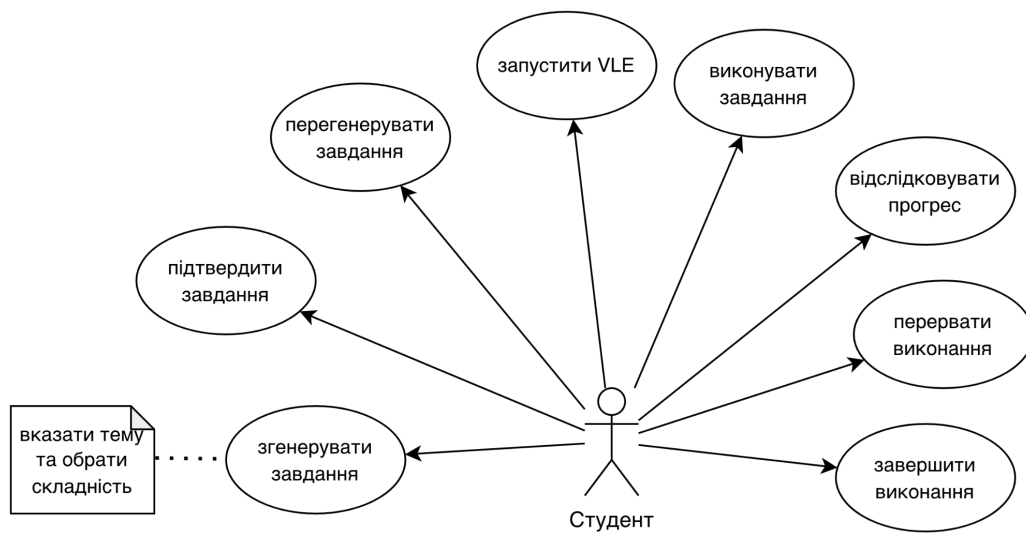


Рис. 3.10. Use Case діаграма виконання студентом PPET в адаптивному режимі.

На рисунку 3.11 представлено Use Case діаграму проходження практичного іспиту в LMS. Основним актором є студент, який виконує завдання у контрольованому VLE. На відміну від режиму навчання, процес є більш структурованим і не передбачає зміни або регенерації завдання, що забезпечує контрольованість оцінювання. Діаграма відображає проведення практичного контролю знань, спрямоване на об'єктивне оцінювання сформованих навичок у стандартизованих умовах.

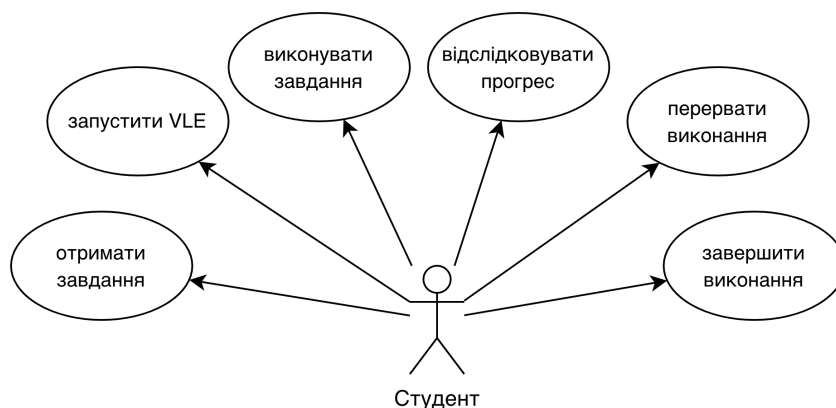


Рис. 3.11. Use Case діаграма проходження практичного іспиту студентом в контрольному режимі.

Процес виконання PPET конкретним студентом показано на рисунку 3.12. Діаграма відображає взаємодію студента з AI-асистентом в рамках LMS, який забезпечує формування запиту, отримання опису завдання, параметрів VLE та відображення прогресу виконання. Генерація та обробка завдання здійснюється у середовищі, де відповідні агенти із залученням GenAI формують персоналізоване завдання, а також забезпечують його валідацію. Подальше розгортання VLE та автоматична перевірка результатів здійснюються через модулі підсистеми, що взаємодіють із VLE. Таким чином, діаграма відображає повний цикл виконання PPET – від запиту студента до отримання результату з автоматизованим контролем процесу виконання.

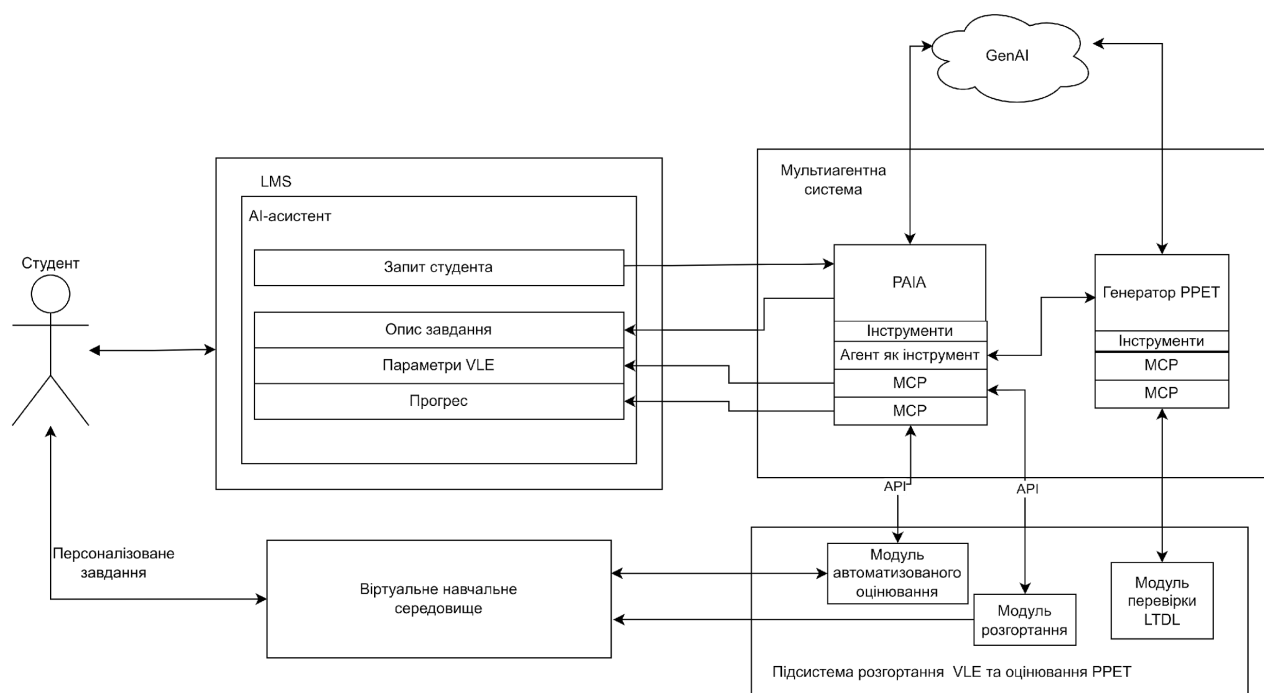


Рис. 3.12. Процес виконання PPET конкретним студентом.

Виконання PPET студентами також здійснюється в рамках підсистеми розгортання VLE та оцінювання PPET, яка інтегрована з LMS Moodle. Для забезпечення інтерактивної взаємодії з користувачами розроблено додатковий модуль РАІА, розгорнутого в системі Flowise. Він складається з AI-агента, набору

спеціалізованих інструментів та MCP-серверів (Model Context Protocol Servers) [169]. З точки зору користувача, PAIA виглядає як AI-асистент, інтегрований у відкрите вікно завдань в LMS Moodle. Студент взаємодіє (формує запит) безпосередньо з асистентом, вказуючи тему, тип завдання або аспект, який він бажає дослідити більш глибоко.

Цей запит за допомогою API передається до мультиагентної системи, де AI-агент поєднує його з попередньо визначеним системним запитом та історичними даними цього конкретного студента про те, завдання якої складності і в якій кількості студент вже виконував. Поєднаний запит надсилається до агента PPETGenerator, який за допомогою GenAI та правил граматики LTDL створює практичне завдання. Успішно згенероване завдання зберігається в сховищі, а персональному AI-асистенту повертається його унікальний ідентифікатор PPETID. Якщо студенту підходить створене завдання, AI-агент надсилає запит API до підсистеми розгортання VLE та оцінювання PPET, щоб ініціювати розгортання VLE, конфігуруючи порти, логіни, змінні середовища, каталоги.

Після розгортання VLE AI-асистент повертає студенту параметри доступу та згенероване PPET. Студент може безпосередньо підключитися до VLE і почати виконувати завдання. Під час виконання завдання AI-асистент періодично надсилає API-запити до підсистеми розгортання VLE та оцінювання PPET для виконання перевірок та моніторингу прогресу. Коли завдання або його частина успішно виконані, прогрес автоматично реєструється та відображається студенту. Діаграма послідовності взаємодії студента з платформою Flowise представлена на рисунку 3.13.

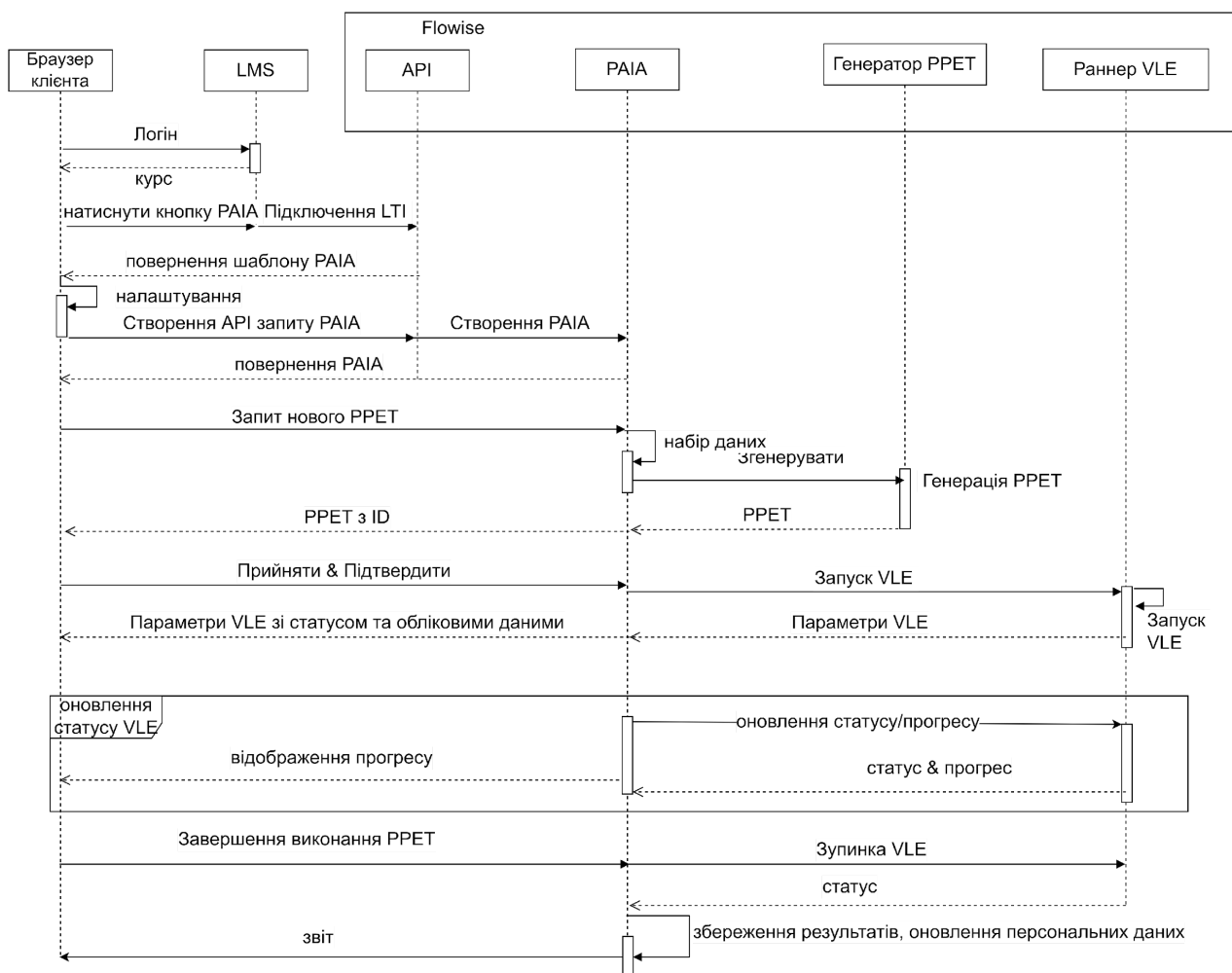


Рис. 3.13. Діаграма послідовності взаємодії студента з платформою Flowise

Підхід до організації виконання практичних завдань у DLE набув подальшого розвитку за рахунок розділення процесу на два операційні режими – адаптивний навчальний та регламентований контрольний, що, на відміну від існуючих рішень, забезпечує поєднання персоналізації навчання з об’єктивністю оцінювання. У навчальному (адаптивному) режимі реалізується гнучкий сценарій виконання завдань із можливістю багаторазових спроб, генерації нових варіантів та адаптації складності, що сприяє формуванню практичних навичок. Після завершення навчальних сесій студент переходить до контрольного режиму, у якому виконує попередньо згенероване завдання в умовах фіксованих параметрів і обмежень. Такий підхід забезпечує розмежування процесів навчання та оцінювання, що є принципово

важливим для досягнення балансу між індивідуалізацією освітнього процесу та стандартизацією контролю результатів.

3.7 Висновки до розділу III

У цьому розділі вперше запропоновано архітектуру інтелектуального асистента, яка, на відміну від існуючих, базується на мультиагентній системі, що реалізує BDI-парадигму в інтерпретації персоналізованого навчання, базуючись на формальному визначенні PPET та його ітераційному покращенні з урахуванням рівня підготовки студента.

В роботі удосконалено 4 взаємопов'язані методи автоматизації PPET:

(1) метод автоматизованої генерації PPET на базі GenAI, який, на відміну від існуючих, забезпечує формальну контрольовану генерацію практичних завдань на основі розробленої мови LTDL та перевірку коректності сформованого завдання за допомогою валідаційних модулів, що мінімізує ризики генерації некоректних описів PPET. Пропонується інтеграція GenAI з обов'язковими механізмами контролю якості. Формальна граматика мови LTDL забезпечує єдиний механізм опису, відтворення та контролю освітніх сценаріїв, тоді як використання GenAI має локальний та контрольований характер і обмежується лише змістовою складовою і функцією персоналізації змісту PPET;

(2) метод параметризації PPET за рахунок поєднання стохастичної генерації параметрів із формальними обмеженнями мови LTDL, що дозволяє, на відміну від існуючих, формувати великий простір унікальних варіантів завдань при збереженні їх структурної коректності;

(3) метод автоматичного розгортання VLE на основі LTDL-опису завдання, який, на відміну від існуючих підходів до організації VLE, забезпечує автоматичну трансляцію формального опису завдання у конфігурацію VLE і дозволяє динамічно створювати унікальні параметризовані VLE для кожного студента, забезпечуючи ізольованість, контроль, повторюваність і масштабованість таких VLE у рамках DLE;

(4) метод автоматичної перевірки результатів виконання PPET шляхом

формалізації стану VLE у вигляді впорядкованого вектора артефактів, що дозволяє, на відміну від існуючих, визначати відповідність поточного стану середовища цільовому стану виконання етапів завдання.

Інтегруючи принципи розробленої DSL LTDL (розділ II) із генерацією завдань на основі AI, пропонується збалансований гібридний підхід: хоча DSL забезпечують формальну точність, педагогічну узгодженість та перевірюваність, вони часто є жорсткими та специфічними для певної галузі. З іншого боку, генератори завдань на основі AI пропонують адаптивність та масштабованість, але часто працюють як «чорний ящик», що викликає занепокоєння щодо прозорості та академічної доброчесності. Запропоновано синтезувати ці підходи, забезпечуючи компроміс, за якого завдання можуть бути як автоматично генеровані, так і формально перевірені.

Запропоновані методи реалізують формальний базис, визначений моделями та мовою LTDL, і виступають основою для побудови інформаційної технології.

Результати досліджень, приведених в розділі, опубліковані в роботах [70, 172].

РОЗДІЛ IV. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРСОНАЛІЗОВАНОГО НАВЧАННЯ З ІНЖЕНЕРНИХ СПЕЦІАЛЬНОСТЕЙ

4.1 Складові інформаційної технології

Складовими запропонованої інформаційної технології є описані в розділі III методи та відповідні програмні засоби, які реалізують ці методи. Пропонується автоматизація на 3 основних етапах: етапі створення, етапі виконання і етапі перевірки завдань. Хоча певні концепції (персоналізація, автоматизоване оцінювання, використання Docker/VM) вже відомі – важливо, що вони інтегровані в єдину технологію з новим комплексним рівнем автоматизації, тому запропонований підхід має інноваційний характер. Автоматизовані підходи додають до даної інформаційної технології масштабованість, вони спрощують процес підготовки матеріалів для великих груп студентів, знижуючи навантаження на викладачів, при цьому забезпечуючи контроль академічної доброчесності. Загальна концепція нової інформаційної технології представлена на рисунку 4.1.

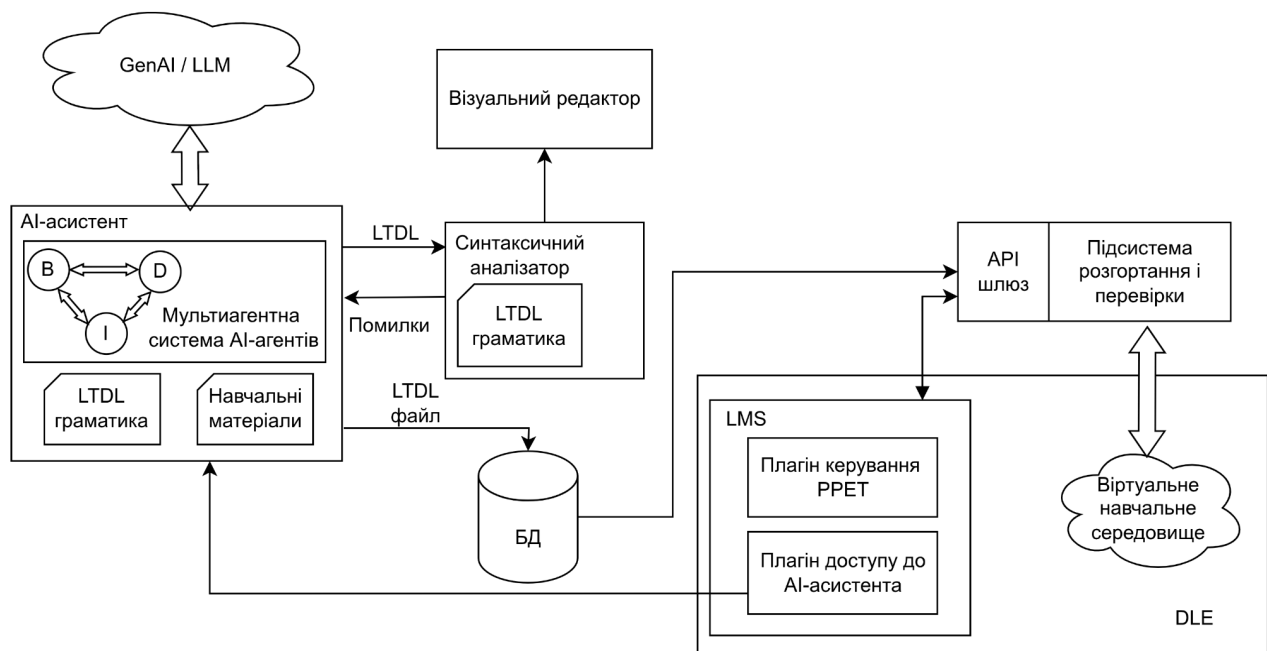


Рис. 4.1. Загальна концепція інформаційної технології.

Для реалізації запропонованих методів у складі нової інформаційної технології

розроблено наступні компоненти:

1) Мультиагентна система оркестрації AI-агентів, побудована на основі технологій Docker та Flowise [162] та LangChain [163]. Docker було обрано як платформу контейнеризації з таких міркувань: забезпечення відтворюваності середовища виконання незалежно від хост-системи, спрощення розгортання та масштабування сервісів, ізоляція залежностей компонентів системи, а також широке застосування у виробничих середовищах та наявність розвиненої екосистеми готових образів.

LangChain є однією з найбільш зрілих та широко використовуваних бібліотек для побудови застосунків на основі LLM. Вибір цього фреймворку зумовлений наявністю готових абстракцій для роботи з ланцюжками запитів, вбудованою підтримкою агентів та інструментів, інтеграцією з широким спектром векторних сховищ і зовнішніх API, а також активною спільнотою та регулярними оновленнями.

Flowise – це платформа з відкритим вихідним кодом, побудована поверх LangChain, що надає графічний інтерфейс для візуального конструювання конвеєрів обробки даних та агентних потоків. Її використання дозволяє скоротити час на розробку прототипів, спростити налаштування та модифікацію логіки без змін у вихідному коді, а також забезпечити наочне представлення архітектури системи.

Сукупність зазначених інструментів утворює збалансований стек, що поєднує гнучкість низькорівневого програмування, зручність візуального налаштування та надійність інфраструктурного рівня. Розроблена система реалізована відповідно до принципів BDI-архітектури, що передбачає розподіл когнітивних функцій між спеціалізованими агентами: агентом бажань, агентом переконань та агентом намірів. Усі агенти використовують LLM Gemini-2.5 як когнітивне ядро та отримують рольову специфікацію через системний промт, що визначає їх поведінку та межі відповідальності.

Агент бажань виконує роль оркестратора системи та є єдиною точкою входу для обробки запитів. Він координує взаємодію між агентом намірів та агентом переконань, делегуючи їм відповідні підзадачі залежно від поточного стану

виконання.

Агент намірів реалізує роль генератора і відповідає за синтез екземплярів РРЕТ у вигляді конструкцій мови LTDL. Для формування коректних та семантично узгоджених виразів агент звертається до векторної бази даних Qdrant, яка виступає як довгострокова пам'ять системи і містить документацію мови LTDL, приклади валідного LTDL-коду та навчальні матеріали й онтології предметної області. Семантичний пошук у цьому сховищі реалізовано за принципом доповненого пошуком генерування (Retrieval-Augmented Generation – RAG), що дозволяє агенту отримувати релевантний контекст для кожного конкретного запиту на генерацію. Після формування чергового варіанту LTDL-виразу агент намірів ініціює його перевірку, передаючи згенерований код синтаксичному аналізатору, та отримує від нього результат у вигляді підтвердження валідності або переліку синтаксичних помилок. У разі виявлення помилок агент намірів виконує корекцію та повторює генерацію. Після успішної валідації згенерований LTDL-код зберігається засобами відповідного інструменту агента. Реалізація агента представлена на рисунку 4.2.

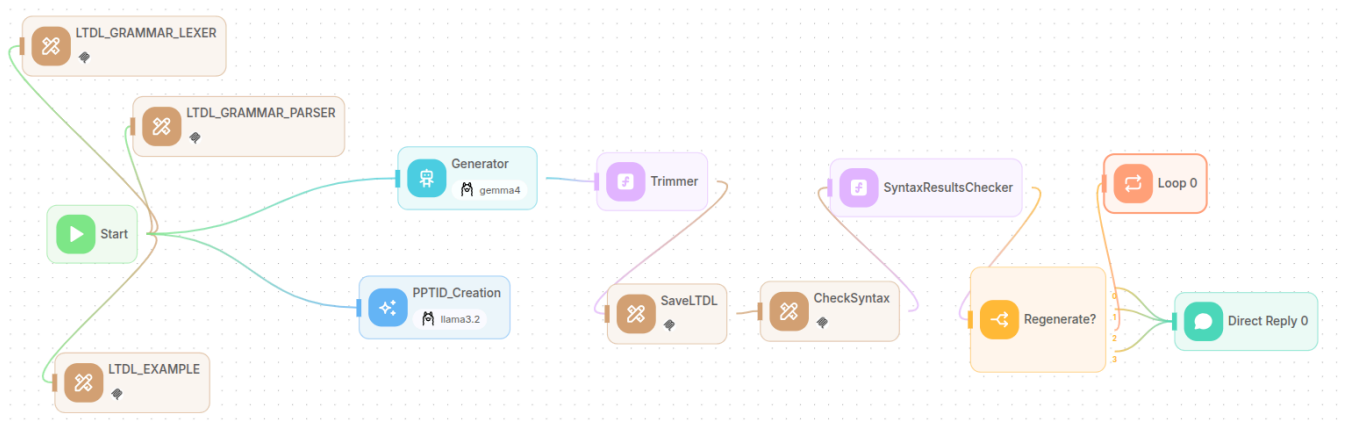


Рис. 4.2. Реалізація агента намірів для генерації РРЕТ

Агент переконань функціонує як актуалізатор стану зовнішнього середовища. За допомогою інструменту виконання HTTP-запитів він взаємодіє з LMS через РАІА-плагін, отримуючи відомості про актуальність навчального контенту, стан розгортання віртуального навчального середовища (VLE), результати студентів та наявність необхідних ресурсів. На основі зібраних даних агент формує актуалізоване

уявлення про поточний стан навчального процесу та при необхідності ініціює виклик підсистеми автоматичного розгортання VLE і перевірки PPEТ через HTTP-інтерфейс.

Архітектура системи (рисунок 4.3) підтримує дворівневу організацію пам'яті. Короткострокова сесійна спільна пам'ять (*short-term memory*) забезпечує обмін контекстом між агентами в межах поточної сесії та реалізована як спільне сховище, доступне всім агентам одночасно. Довгострокова пам'ять (*long-term memory*) забезпечує семантичний пошук у базі знань [164] предметної області. Взаємодія між компонентами системи здійснюється через HTTP-запити у форматі JSON, що забезпечує слабку зв'язаність компонентів та спрощує їх незалежне масштабування.

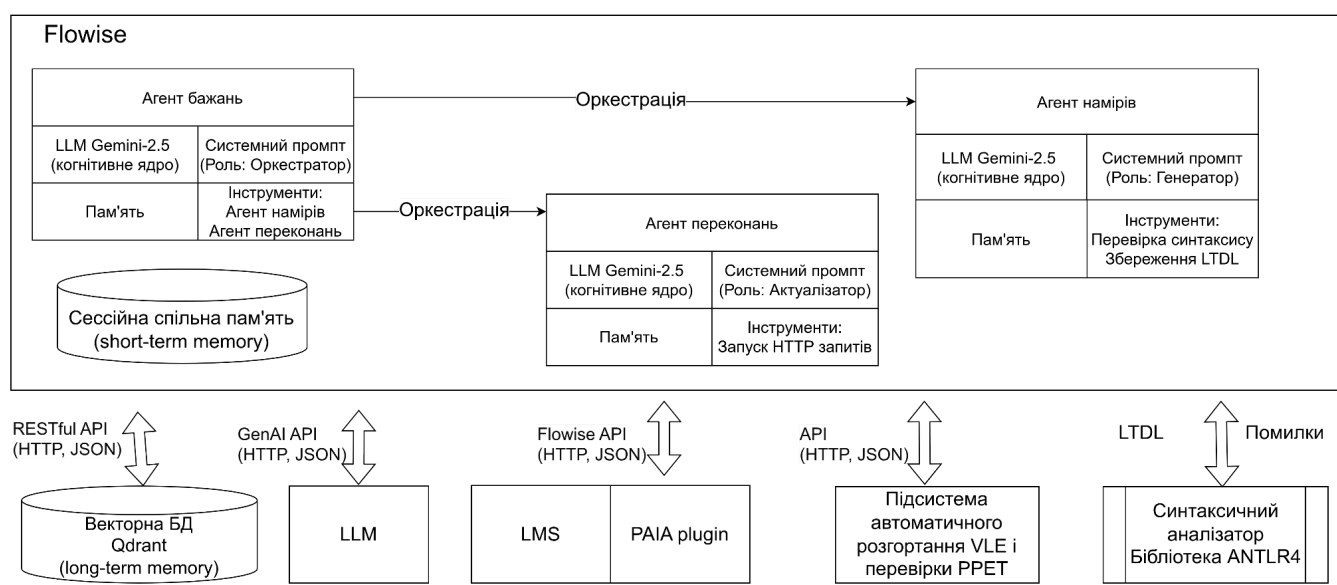


Рис. 4.3. Архітектура мультиагентної системи.

Приклади створених колекцій для векторної бази даних показані на рисунку 4.4.

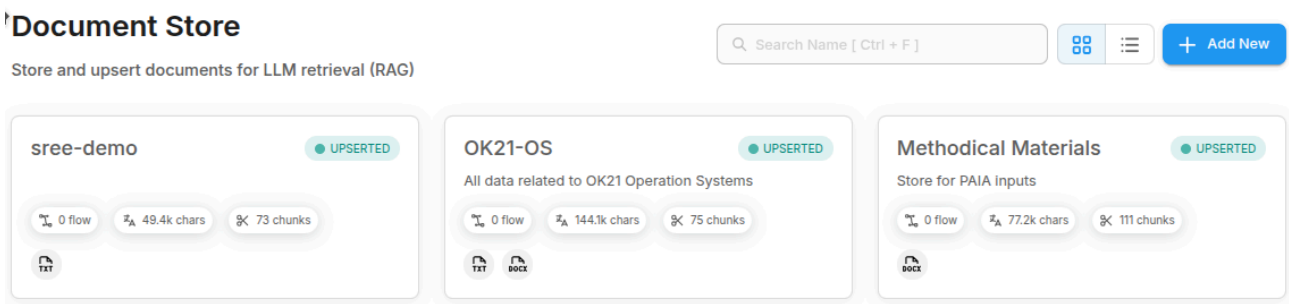


Рис. 4.4. Приклади колекцій у векторній базі даних.

При виборі векторної бази даних розглядалися рішення, які підтримуються Flowise, такі як Pinecone, Weaviate, Chroma та Qdrant. Pinecone є хмарним сервісом без можливості локального розгортання, а Chroma орієнтована переважно на прототипування і поступається за продуктивністю у виробничих середовищах. Weaviate є функціонально насиченим рішенням, проте має вищу складність налаштування. Qdrant вирізняється оптимальним балансом між продуктивністю, простотою інтеграції та можливістю повністю локального розгортання, що й зумовило його вибір у межах розробленої систем

Запропонована архітектура забезпечує інтеграцію формалізованих компонентів курсу із механізмами семантичного пошуку та генерації, що дозволяє розглядати персонального AI-асистента як інтелектуальний інтерфейс доступу до навчального контенту та як складову частину загальної системи персоналізації практичних інженерних завдань.

2) Програмний засіб PAIA-moodle-plugin розширює можливості LMS Moodle для інтеграції з платформою Flowise, і дозволяє створювати персонального AI-асистента та виступає інтерфейсом взаємодії з AI-асистентом для створення PPET та зберігання результатів їх виконання. Інтеграція персонального асистента в курс в LMS Moodle продемонстрована на рисунку 4.5.

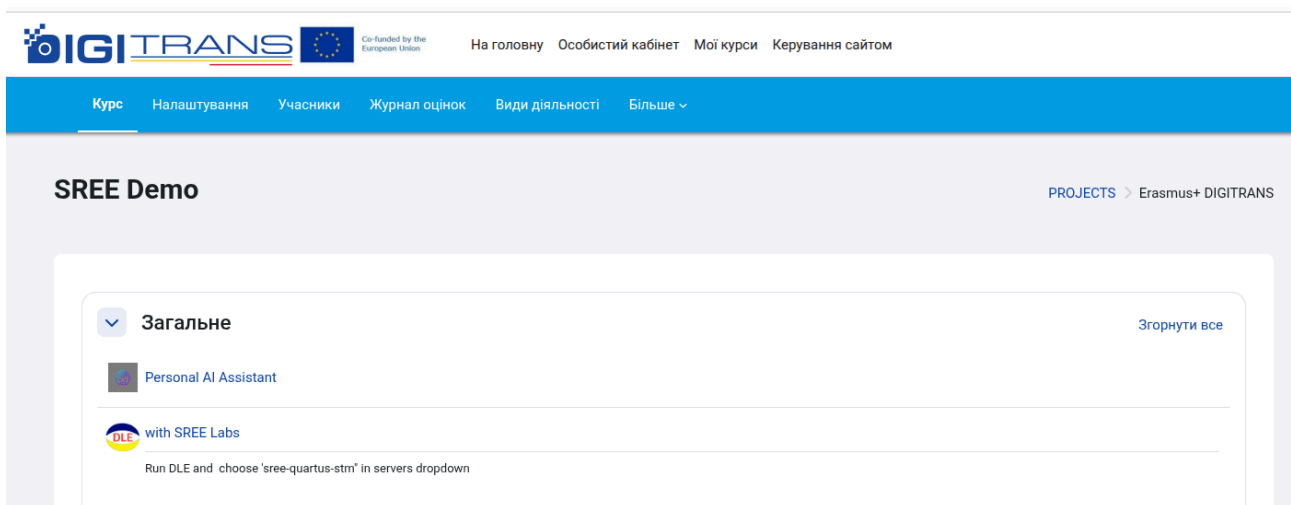


Рис. 4.5. Інтеграція персонального асистента в курс в LMS Moodle.

Вибір мов програмування для реалізації плагіну визначається архітектурою самої платформи Moodle. Оскільки Moodle є PHP-застосунком із реляційною базою даних, використання PHP та SQL є обов'язковою умовою сумісності. Інтерфейсна частина реалізована стандартними засобами веб-розробки - HTML, CSS та JavaScript - що відповідає внутрішнім конвенціям платформи та забезпечує коректну роботу в межах її шаблонної системи. Інтерфейс взаємодії з персональним AI-асистентом у LMS Moodle представлено на рисунку 4.6.

Персональний ШІ асистент студента

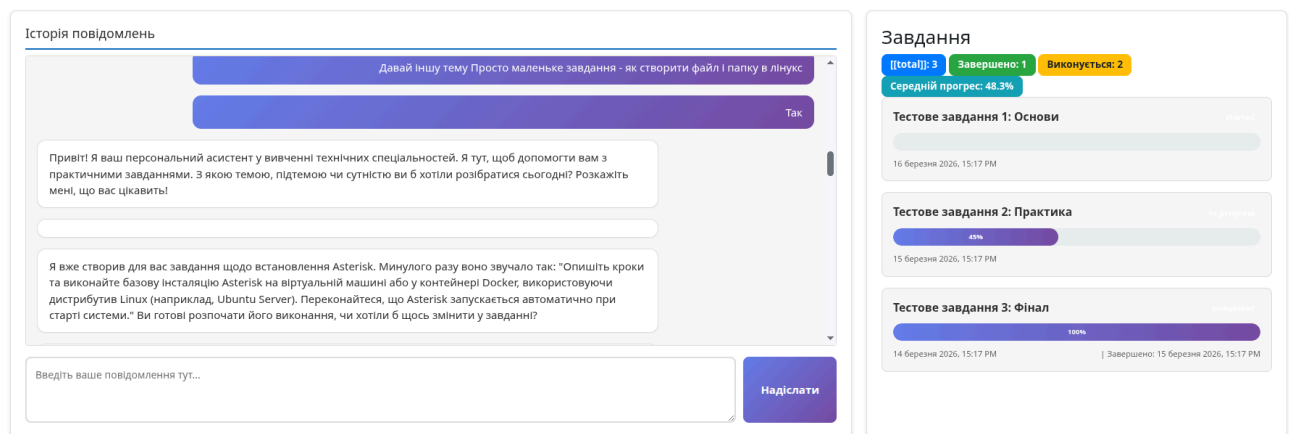


Рис.4.6. Інтерфейс доступу до персонального AI-асистента

3) Синтаксичний аналізатор LTDL. Для розробки граматики LTDL використовувалося середовище VSCode у поєднанні з плагіном ANTLR4, яке обрано з огляду на можливість інтерактивного налагодження граматики, підсвічування синтаксису та візуалізації дерева розбору безпосередньо під час розробки – на відміну від запуску ANTLR4 через командний рядок. Формальний опис граматики LTDL виконано у нотації EBNF, яка є загальноприйнятим стандартом для специфікації синтаксису формальних мов. На основі цих описів генератор ANTLR4 автоматично створює лексичний та синтаксичний аналізатори. Вибір Python 3 як цільової платформи генерації обумовлено його інтеграцією з іншими компонентами системи та зрілістю відповідного середовища виконання. Лексичний аналізатор

розбиває вхідний текст на окремі лексеми, що відповідають конкретним правилам мови. Синтаксичний аналізатор перевіряє структуру вхідного тексту на відповідність граматиці, створюючи дерево синтаксичного аналізу. За допомогою згенерованих програмних засобів це дерево синтаксичного аналізу можна обробити для виконання необхідних перетворень коду LTDL в будь-які інші описи чи формати, наприклад для інтеграції з іншими елементами DLE, наприклад, SMSE [167].

Ключові можливості налагодження та тестування розробленої граматики включають:

- (а) побудову абстрактного синтаксичного дерева (AST), яке показує граматичну структуру без конкретних синтаксичних деталей, що дозволяє зрозуміти логіку виразів та виявити помилки, конфлікти, неоднозначності в коді;

- (б) синтаксичні діаграми Railroad (Nora), які допомагають візуалізувати структуру мови, перевіряти узгодженість, виявляти неоднозначності або нечіткі повторення та забезпечують візуальний контроль над граматиною;

- (в) генерацію речень для правил, щоб перевірити дійсні та недійсні послідовності термінальних символів.

Для тестування створеної граматики та згенерованих лексичного та синтаксичного аналізатора створено та проведено наступні тести: парсинг тести, лексичне тестування, тестування генерації коду, побудова синтаксичного дерева, визначення непродуктивних та недосяжних символів. Опис процедури тестування LTDL наведено в додатку Є. Розроблена граматика не містить зайвих символів та всі визначені правила мають практичне застосування і можуть бути досягнуті під час парсингу.

4) Онлайн-редактор LTDLEditor. Для спрощення сприйняття та можливості охоплення всіх аспектів розробленого PPET автором пропонується можливість графічного представлення елементів завдання. У мові візуального програмування зображення, складене з взаємопов'язаних графічних елементів, трансформується у змістові конструкції, що запускаються на обчислення. Мова візуального програмування може бути визначена формальною граматиною, так само, як і

текстова мова програмування, за умови, що визначені зв'язки між графічними елементами. Сутності PPET представлені у вигляді графічних зображень символів мови. Повний перелік параметрів елементів не виводиться у графічному представленні, окрім параметру *name*, який призначений для відображення назви відповідного елементу. Графічні зображення символів алфавіту мови LTDL наведені в додатку Ж.

LTDLEditor реалізує графічний рівень взаємодії з мовою LTDL, орієнтований на кінцевого користувача – викладача. Вибір React як основи для розробки інтерфейсу обумовлено його компонентною архітектурою, що спрощує побудову складних інтерактивних інтерфейсів та забезпечує ефективне керування станом застосунку. Панель діаграм дозволяє створювати об'єкти та пов'язувати їх між собою за допомогою логічних відносин з підтримкою вкладеності. Користувачі можуть інтерактивно працювати з діаграмою, включаючи масштабування, вибір і переміщення елементів у режимі перетягування. Панель налаштувань дозволяє визначати та змінювати властивості об'єктів LTDL. Завдання, створені за допомогою візуального редактора, автоматично зберігаються у текстовому форматі LTDL і можуть бути використані для подальшого розгортання VLE або імпортовані у редактор для редагування або розширення. Зовнішній вигляд панелей показано на рисунку 4.7.

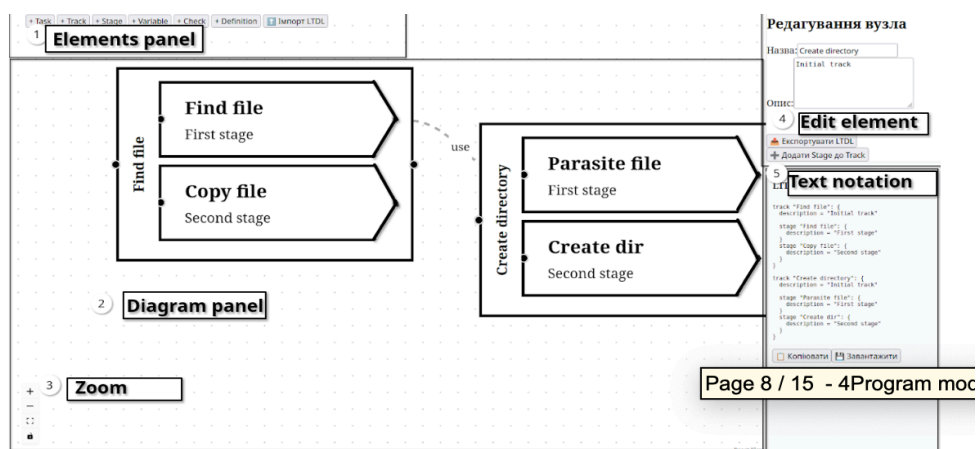


Рис.4.7. Інтерфейс візуального редактора LTDLEditor

На рисунку 4.8 показано приклад представлення завдання за допомогою графічного варіанту мови.

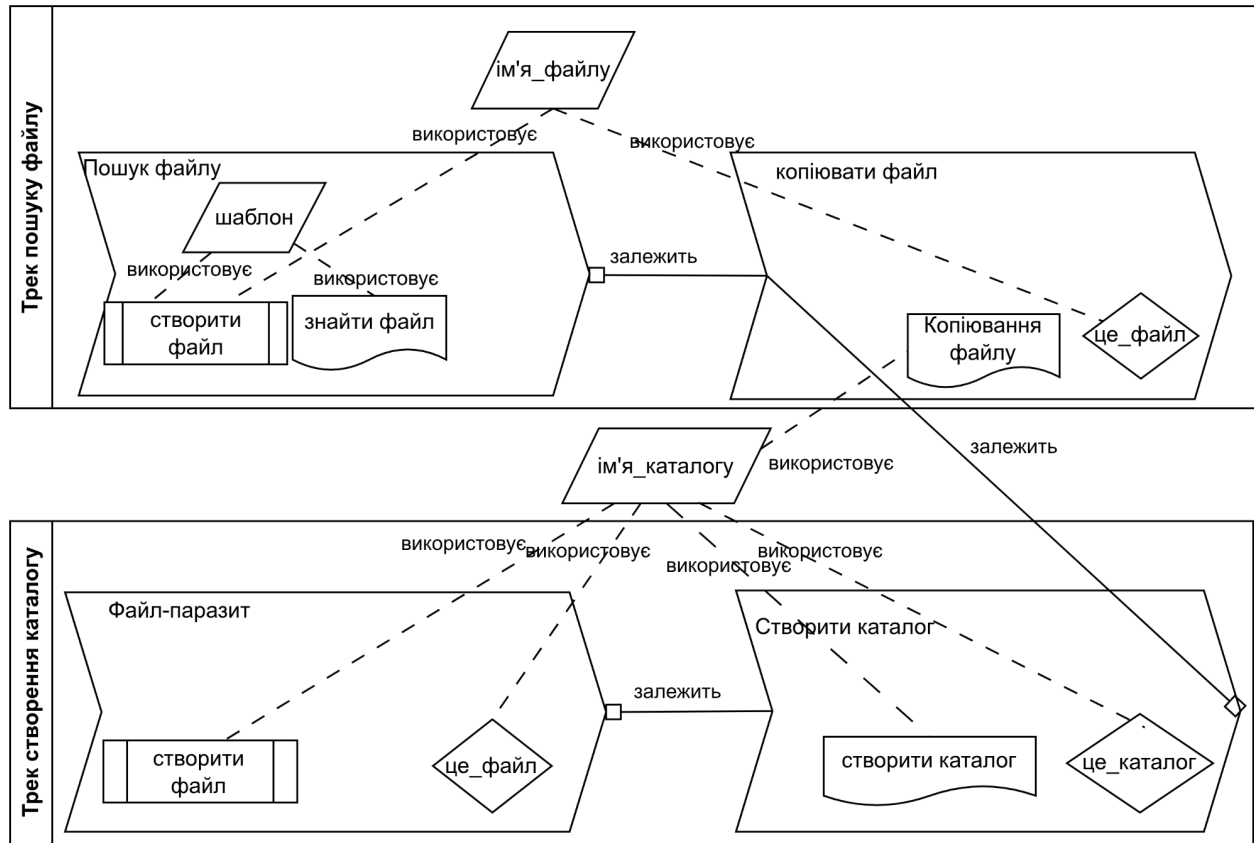


Рис. 4.8. Приклад представлення PPET графічним варіантом мови LTDL.

5) Програмний засіб **LTDL-moodle-plugin** розширює можливості LMS Moodle для інтеграції з підсистемою автоматичного розгортання і перевірки PPET і дозволяє викладачу створювати контрольні PPET та зберігати результати їх виконання. Плагін створено з використанням мов HTML, CSS, JavaScript, PHP, SQL.

Вигляд LTDL Practical Task модуля у LMS Moodle представлено на рисунку 4.9.

Рис. 4.9.

Налаштування модуля LTDL Practical Task

Приклад виводу результатів виконання завдань студентами показано на рисунку 4.10.

LTDL Practical Task

Налаштування

Більше

Модульний контроль 1

Модульний контроль 1

Attempts report

Collapse/Expand all

Last refresh: 15:40

☒ Auto refresh

↺

Швець Євгеній Віталійович

🏆 100%

⌚ 2 хв 48 сек

⌵

User Exists

Group Exists

User In Group

Primary Group Is Username

Secondary Group Listed In Group File

📅 14 квітня 2026

📄

📄

» details

Хижняк Андрій Васильович

🏆 100%

⌚ 3 хв 3 сек

⌵

User Exists

Group Exists

User In Group

Primary Group Is Username

Secondary Group Listed In Group File

📅 14 квітня 2026

📄

📄

» details

Швець Євгеній Віталійович

🏆 100%

⌚ 1 день 14 години

⌵

User Exists

Group Exists

User In Group

Primary Group Is Username

Secondary Group Listed In Group File

📅 9 квітня 2026

📄

📄

» details

Рис. 4.10 Приклад виводу результатів контрольного завдання

Вигляд запущеного практичного завдання студентом показана на рисунку 4.11.

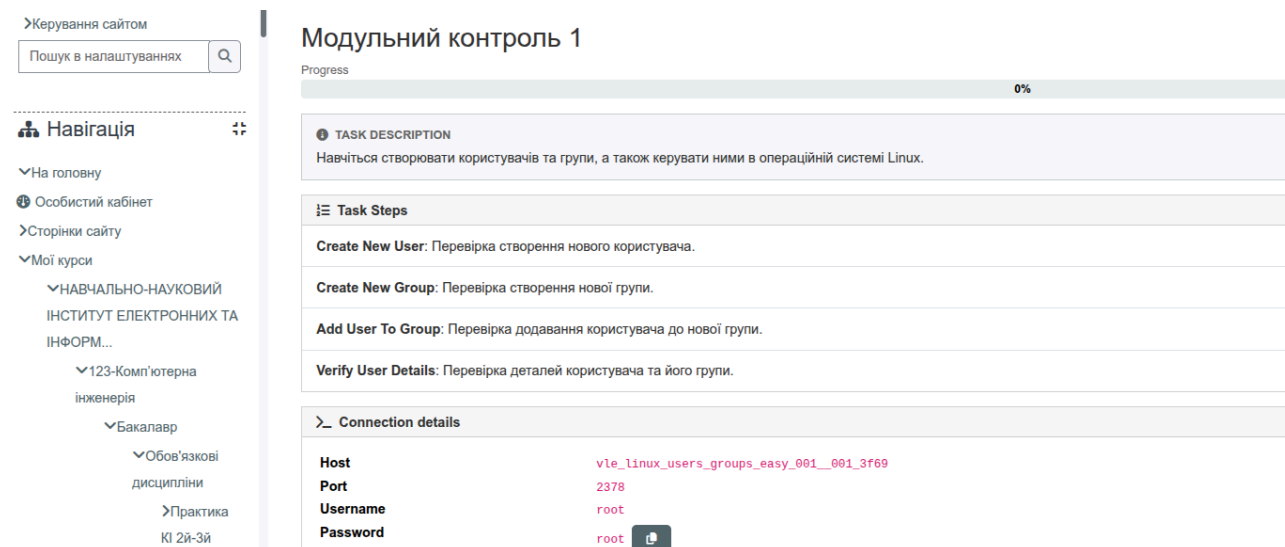


Рис. 4.11. Вигляд запущеного практичного завдання в LMS Moodle.

б) Підсистема автоматичного розгортання VLE і перевірки PPET.

В рамках практичної реалізації ІТ створено підсистему для автоматичної перевірки PPET з підтримкою функції автоматичного розгортання ізолюваних та контрольованих індивідуальних VLE [168]. Архітектуру підсистеми побудовано навколо API Gateway, реалізованого на FastAPI - сучасному Python-фреймворку, що поєднує високу швидкодію з мінімальними витратами на розробку завдяки декларативному опису маршрутів та вбудованій підтримці типізації через Pydantic. П'ять модулів підсистеми функціонують у ізолюваних середовищах виконання, що забезпечується спільним застосуванням VirtualBox та Docker. Використання гіпервізора VirtualBox доцільне для завдань, що потребують повної апаратної ізоляції або специфічного операційного середовища, тоді як Docker застосовується для завдань із типовими вимогами до оточення, забезпечуючи швидке розгортання та ефективне використання ресурсів.

4.2 Архітектура підсистеми автоматичного розгортання VLE

Для визначення компонентів, їх структурування і об'єднання в підсистему, а також для зниження складності системи шляхом абстракції і розмежування повноважень розроблена архітектура підсистеми, представлена на рисунку 4.12.

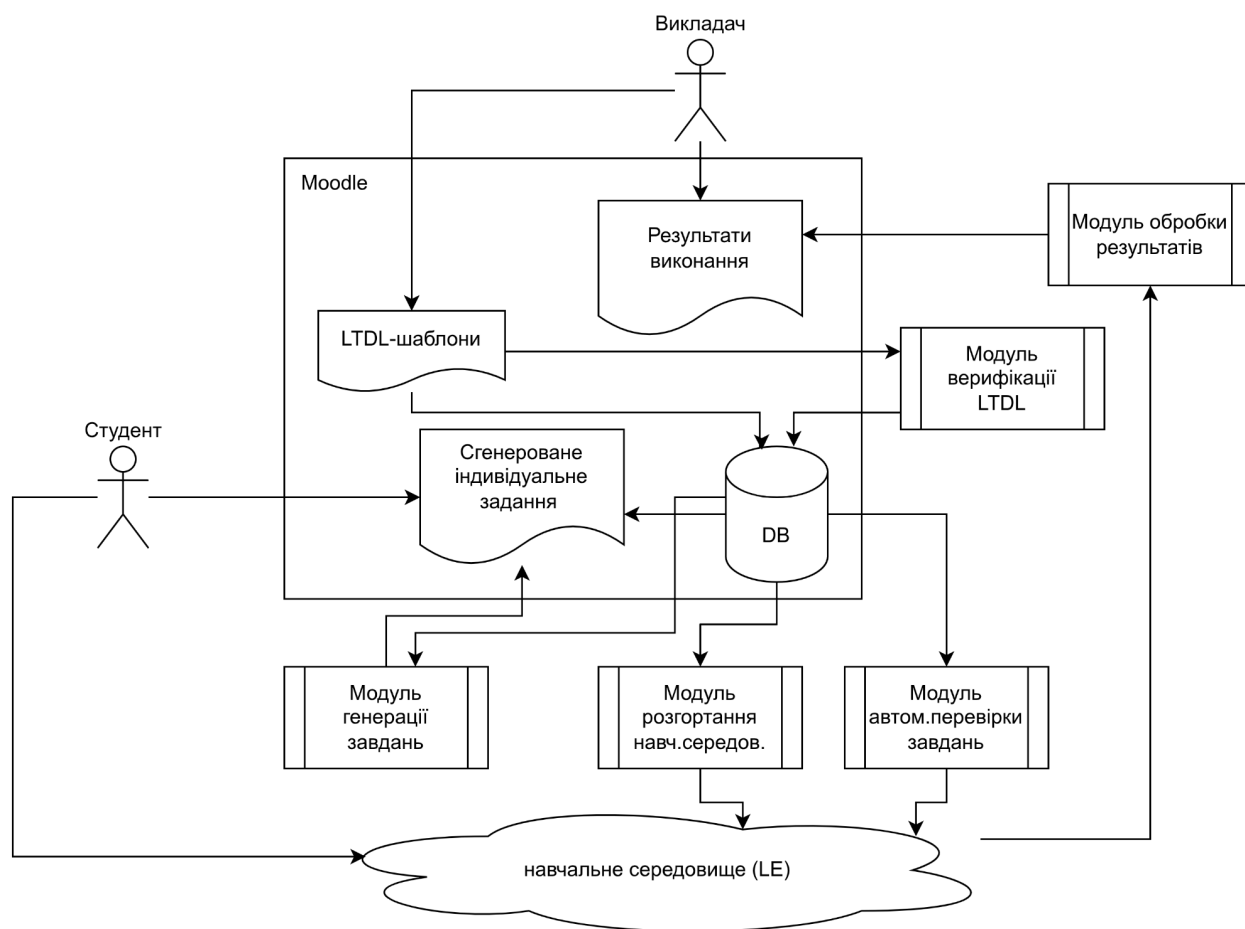


Рис. 4.12. Архітектура підсистеми автоматичного розгортання PPET

Підсистема складається із 5 модулів: модулю генерації завдань, модулю верифікації LTDL, модулю розгортання VLE, модулю автоматичної перевірки завдань та модулю обробки результатів виконання. Розглянемо детальніше кожен модуль, його функціональне призначення, місце в архітектурі та взаємозв'язки з іншими компонентами системи.

Модуль генерації завдань (№1). Модуль інтегрується з наявною LMS і дозволяє створити PPET в рамках вивчення якоїсь теми з обраної дисципліни. За допомогою модуля генерації і розробленої мови створюється LTDL документ, який містить наступні елементи:

a) *student assignment template* – шаблон завдання, яке буде використане для генерації параметризованого завдання для конкретного студента з певним набором

унікальних змінних параметрів.

б) *learning environment* – опис VLE та набір інструкцій для модуля розгортання №3. В цьому елементі вказується тип обраного VLE та перелік дій або команд, які повинні бути виконані модулем для правильного розгортання і налаштування відповідного VLE. Оскільки для виконання параметризованого завдання студент повинен отримати не тільки опис завдання, але і параметри доступу до VLE, то перед видачею завдання студентові його VLE запускається.

в) *PPET verification instructions* – набір інструкцій для модуля №4 автоматичної перевірки PPET, згідно яких фіксується успішність виконання завдання у відповідному VLE.

Модуль верифікації LTDL (№2). Для створення вказаних шаблонів пропонується використовувати спеціально розроблену і описану в розділі II домен-орієнтовану мову, LTDL. Документ LTDL в подальшому буде перетворено на опис конкретного унікального завдання, конфігураційні файли систем розгортання для хмарних сервісів або конфігураційних файлів Docker-контейнерів чи віртуалізаторів на кшталт VirtualBox, VMWare, Vagrant. Створені документи зберігаються в існуючій базі даних LMS та можуть бути в подальшому змінені, оновлені чи експортовані в інші формати. Також модуль дозволяє імпорт шаблонів з підтримуваних форматів.

Для функціонування системи модуль генерації завдань повинен створити коректний LTDL-документ, який в подальшому буде використовуватися іншими модулями: модулем розгортання VLE та модулем автоматичної перевірки. Для цього всі модулі системи будуть використовувати модуль верифікації LTDL, задача якого провести розбір та верифікацію документу на відповідність специфікації LTDL.

Модуль розгортання VLE (№3). Задача модуля - розгорнути VLE для виконання PPET конкретним студентом згідно інструкцій. Ці інструкції визначають, який тип VLE треба розгорнути: окрему VM, LXC\docker контейнер, хмарний сервіс, JupyterLab тощо. В залежності від обраного типу модуль запускає екземпляр VLE і виконує перелік дій, які налаштовують конкретний екземпляр VLE для

конкретного студента: створить або завантажить потрібні файли чи директорії, інсталує необхідне ПЗ, сконфігурує підсистему аутентифікації та авторизації, налаштує мережу.

Для виконання цих задач модуль повинен отримати інструкції від LMS у вигляді LTDL-документу і провести верифікацію цього документу за допомогою відповідного модулю. Коли студент створить запит на виконання завдання, модуль розгортання отримає LTDL-документ з необхідними для створення VLE параметрами. Результат розгортання VLE модуль повертає у вигляді відповідного статусу і в подальшому виконує його контроль. Конфігураційні параметри VLE будуть використані модулем верифікації при підключенні до екземпляру VLE для проведення верифікації і формування звіту про прогрес виконання завдання та його фінальні результати. Модуль також виконує функції згортання або знищення непотрібних VLE, для цього він взаємодіє з модулем автоматичної перевірки завдань та з модулем обробки результатів, від яких в певний момент отримує сигнал на завершення роботи відповідного VLE.

Модуль автоматичної перевірки РРЕТ (№4). Задача модуля - перевірити правильність виконання студентом його власного завдання в окремому VLE згідно отриманих інструкцій. Спочатку цей модуль також використовує модуль верифікації LTDL для перевірки коректності отриманих інструкцій. Далі модуль виконує підключення до конкретного VLE з використанням відповідних параметрів для аутентифікації та авторизації та перевіряє чи виконане завдання. Підключення та перевірка виконується з певною, заданою наперед в LTDL-інструкціях, періодичністю, наприклад, кожні 5 або 10 хвилин. В разі успішного виконання завдання (чи переліку завдань) модуль виконує наступні дії: сповіщає студента про успішність виконання завдання, передає дані перевірки модулю обробки результатів та сповіщає модуль розгортання про необхідність завершити роботу конкретного VLE. Якщо модуль не зміг підключитися до VLE, він сповіщає про це модуль обробки результатів.

Модуль №5, для обробки результатів. Задача модуля полягає у відстеженні і

відображенні результатів виконання завдання. Загальний вигляд результату виконання завдання залежить від налаштувань конкретної LMS і може мати як фіксований набір значень: “виконане”, “спроба провалена”, “термін виконання сплив” так і прогрес виконання у відсотках. Результати виконання завдання даний модуль отримує від модулю автоматичної перевірки завдань.

Також модуль відображає поточний статус VLE, який він отримує від модулю розгортання або від модулю автоматичної перевірки завдань. Можливі статуси VLE: “не створене”, “створене”, “помилка створення”, “запущене”, “скасоване”, “завершене”.

4.3 Програмний інтерфейс керування VLE

Для забезпечення взаємодії між компонентами розробленої підсистеми розгортання VLE та оцінювання PPET та зовнішніми інтерфейсами користувача та плагінами LMS спроектовано та реалізовано API на базі архітектурного стилю REST. Архітектура API побудована за принципом мікросервісів, де маршрутизація запитів здійснюється через єдину точку входу (веб сервер Nginx), що забезпечує ізоляцію внутрішніх сервісів та спрощує масштабування.

Функціонально API розподілено на два ключові блоки:

- Сервіс розгортання та керування (*Deployment Service*) – відповідає за життєвий цикл VLEs та менеджмент навчального контенту. До його основних функцій належать завантаження та парсинг сценаріїв у форматі LTDL, асинхронний запуск процесу розгортання VLE для конкретного студента, моніторинг активних сесій та агрегація результатів навчання для формування звітності (*scoreboard*) у форматах JSON та CSV.
- Сервіс верифікації (*Verification Service*) – спеціалізований модуль для динамічної перевірки стану виконання практичних завдань. Сервіс забезпечує підключення до розгорнутих VLEs за протоколом SSH та ініціює послідовне виконання перевірок, визначених у моделі задачі.

Взаємодія із системою реалізована через набір кінцевих точок (endpoints), що підтримують стандартні методи передачі даних (GET, POST, DELETE), що наведені

в Таблиці 4.1. Для гарантування цілісності даних доступ до критичних операцій, таких як видалення сценаріїв або примусове знищення віртуальних машин, обмежений префіксом `/control/`, що дозволяє розмежувати права доступу між студентами та адміністративним інтерфейсом викладача. Документування API виконано з використанням специфікації OpenAPI (Swagger), що забезпечує автоматизовану генерацію клієнтських бібліотек та спрощує інтеграцію з іншими компонентами DLE.

Таблиця 4.1

Набір кінцевих точок і опис їх функціонального призначення

Категорія	Метод	Шлях (Endpoint)	Опис функціонального призначення
Моніторинг	GET	/status	Перевірка працездатності сервісу та стану з'єднання з базою даних MongoDB
	GET	/active_sessions	Отримання переліку поточних навчальних сесій та їх статусів у реальному часі.
	GET	/scoreboard	Формування агрегованих даних про прогрес студентів для викладацького дашборду.
Керування контентом	POST	/control/scenarios/upload	Завантаження файлів описів PPET у форматі LTDL для подальшої генерації середовищ.
	GET	/scenarios	Перегляд бібліотеки доступних навчальних сценаріїв у системі.
	DELETE	/control/scenarios/{id}	Видалення конкретного сценарію або очищення всієї бази сценаріїв.
Керування VLE	POST	/control/deploy/{id}	Ініціалізація розгортання VLE для конкретного студента за вказаним сценарієм.
	POST	/control/destroy/{id}	Завершення сесії та деструкція (видалення) обчислювальних ресурсів VLE.
Верифікація	POST	/control/run_check/{id}	Проксі-запит на запуск процедури автоматизованої перевірки виконання етапів завдання.

	POST	/verify/{session_id}	Безпосереднє підключення до VLE по SSH для виконання скриптів перевірки та повернення результатів.
Адміністрування	GET	/control/export/csv	Експорт результатів проходження практичних робіт у форматі CSV для подальшого аналізу.
	POST	/control/system/cleanup_db	Повне очищення історії сесій та тимчасових даних із бази даних.

Розроблений алгоритм верифікації базується на принципі послідовного підтвердження станів (*state validation*). На відміну від статичного аналізу коду, такий підхід дозволяє перевіряти реальну конфігурацію систем (наприклад, стан мережевих інтерфейсів, активність сервісів або вміст бази даних), що є критично важливим для підготовки фахівців з комп'ютерної інженерії. Використання SSH як транспорту забезпечує універсальність сервісу, дозволяючи йому працювати з будь-якими Linux-подібними операційними системами без необхідності встановлення додаткових агентів всередині VLE.

На рисунку 4.13 зображено архітектуру API за принципом мікросервісів.

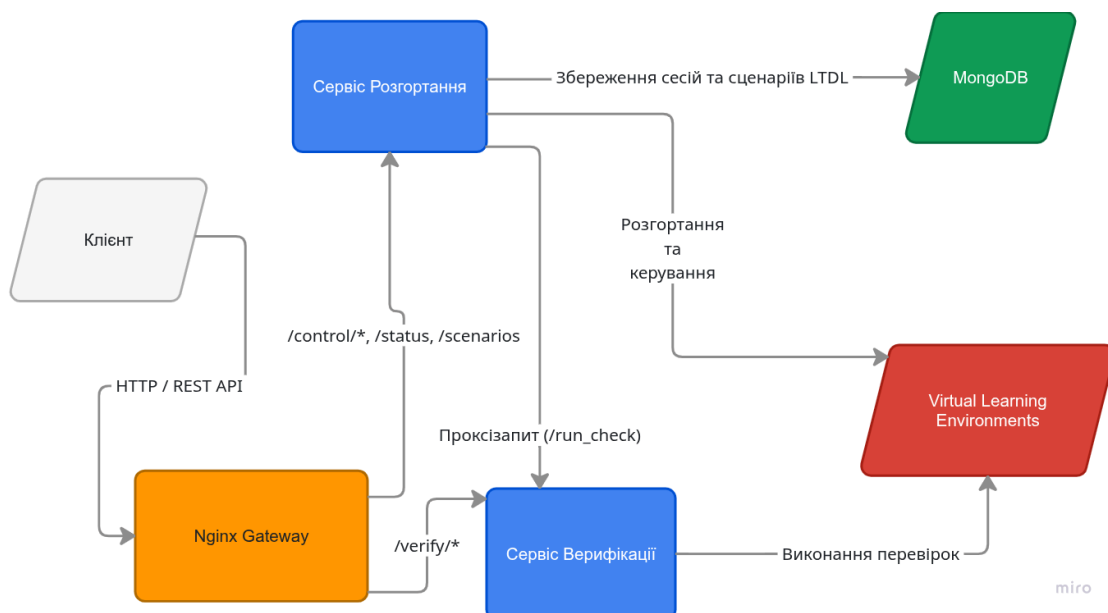


Рис. 4.13. Архітектура API побудована за принципом мікросервісів.

Nginx Gateway: Виступає єдиною точкою входу (шлюзом) для всіх клієнтських запитів на порту 80. Він виконує зворотнє проксіювання та

маршрутизацію трафіку до відповідних мікросервісів (/control/ до сервісу розгортання, /verify/ до сервісу верифікації).

Deployment Service (Сервіс розгортання): Ядро системи управління. Відповідає за обробку файлів сценаріїв (зокрема у форматі LTDL), збереження даних про активні сесії та результати (*scoreboard*) у базі даних MongoDB, а також безпосереднє управління життєвим циклом віртуальних машин (VLE).

Verification Service (Сервіс верифікації): Ізольований компонент, що фокусується виключно на процесі перевірки. Після ініціалізації запиту (безпосередньо через шлюз або через проксі-запит від сервісу розгортання), цей сервіс встановлює SSH-з'єднання з відповідним VLE для виконання кроків перевірки практичного завдання.

MongoDB: база даних, що використовується сервісом розгортання для зберігання стану системи, завантажених сценаріїв та статистики виконання завдань студентами.

Virtual Learning Environments (VLE): Цільові VLE, які динамічно створюються та знищуються за запитом, і в яких студенти виконують PPET.

На рисунку 4.14 зображено діаграму послідовності (*Sequence Diagram*), яка деталізує інформаційну взаємодію між користувачами та компонентами програмного комплексу під час життєвого циклу практичного завдання.

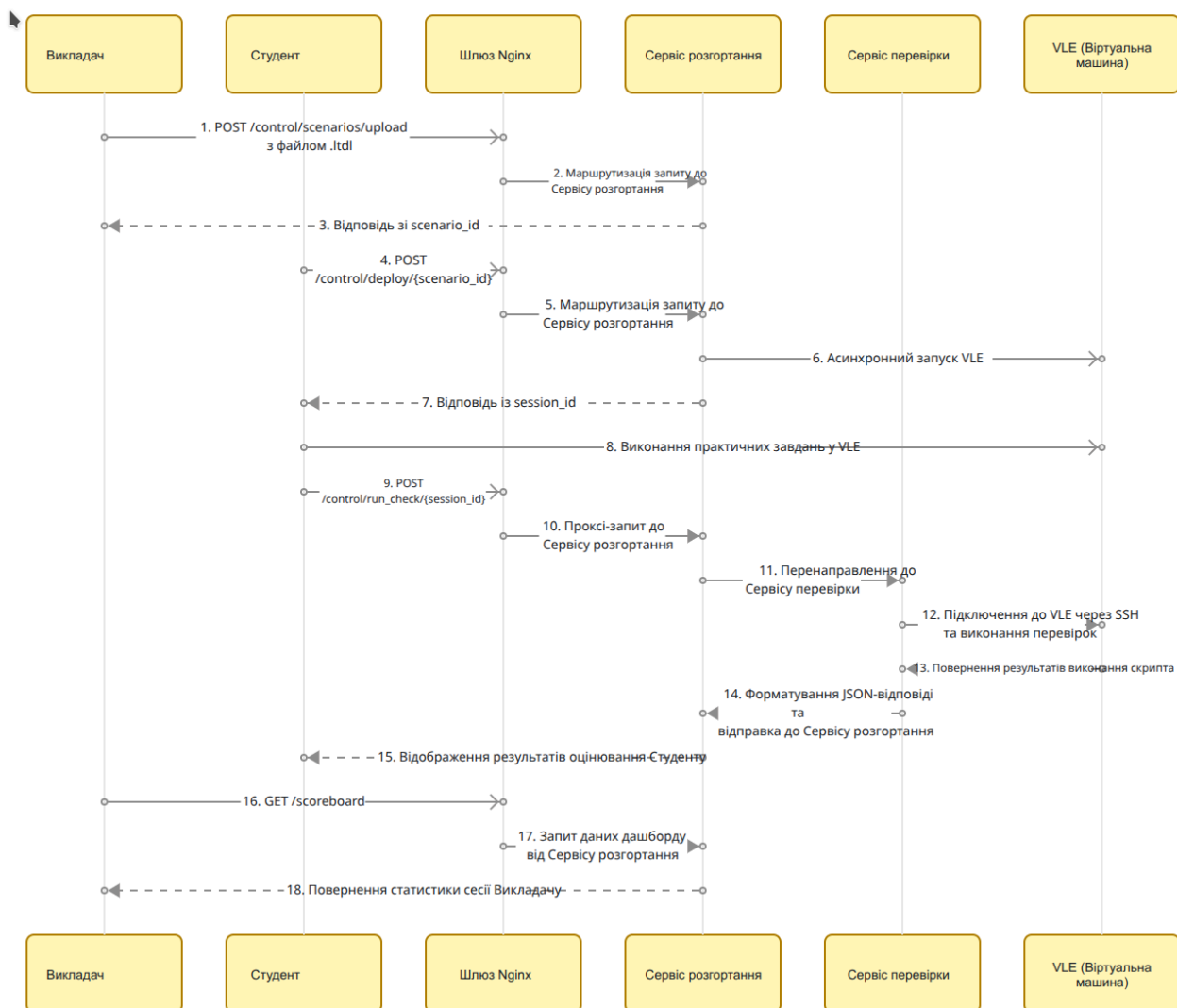


Рис. 4.14. Діаграма послідовності взаємодії між користувачами та компонентами програмного комплексу під час життєвого циклу PPET.

Процес розпочинається з ініціалізації навчального контенту, коли викладач завантажує формалізований опис завдання у форматі мови LTDL через кінцеву точку *POST /control/scenarios/upload*. Після збереження сценарію сервісом розгортання, система готова до обслуговування запитів здобувачів освіти. Студент ініціює створення ізолюваного робочого середовища запитом *POST /control/deploy/{scenario_id}*, передаючи власні ідентифікаційні дані. Сервіс розгортання асинхронно запускає віртуальну машину (VLE) та повертає унікальний ідентифікатор сесії (*session_id*)

Після виконання практичних дій у VLE, студент ініціює процес автоматизованого оцінювання (*POST /control/run_check/{session_id}*). Запит маршрутизується до сервісу верифікації, який встановлює SSH-з'єднання з відповідною віртуальною машиною та виконує перевірку кроків завдання, передбачених сценарієм LTDL. Результати перевірки (кількість пройдених та загальних тестів) повертаються студенту у форматі JSON. Одночасно система оновлює загальну статистику успішності, яку викладач може отримати через агрегований запит *GET /scoreboard* для моніторингу прогресу всієї групи.

Інтерфейс користувача панелі адміністратора показано на рисунку 4.15.

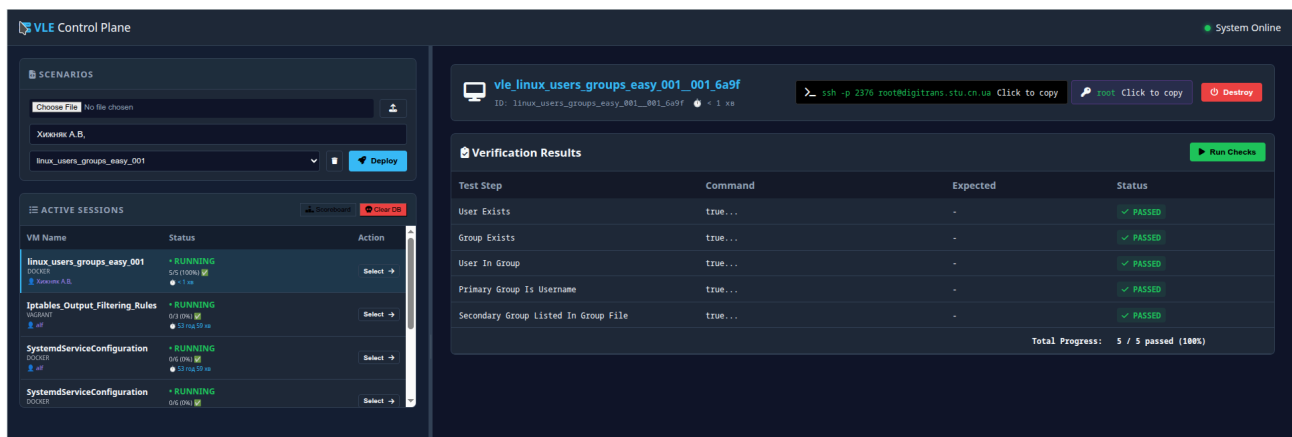


Рис.

4.15. Панель адміністратора для керування VLE.

4.4 Експериментальне дослідження інформаційної технології

4.4.1 Проведення експериментів

Ефективність запропонованих моделей, методів та інформаційної технології в процесах персоналізованого навчання студентів інженерних спеціальностей оцінювалася за допомогою серії експериментів. В експерименті взяли участь здобувачі інженерних спеціальностей та курсанти неформальної освіти. Також під час проведення експериментів визначено рівень теоретичної підготовки студентів (метод діагностики: теоретичний тест по модулю) та стан виконання практичних (лабораторних) робіт студентами. Визначено випадки академічної недоброчесності (підробка за допомогою графічних редакторів та методами AI). Доведено, що

теоретичні знання (тести) та практичні навички (робота в середовищі) – це різні компетенції. Традиційні теоретичні тести не можуть надійно оцінити практичну підготовку, що посилює актуальність запропонованої інформаційної технології. Проведено дослідження якості генерації PPET на основі LTDL; порівняльну оцінку ефективності різних моделей GenAI для автоматичного створення PPET у форматі LTDL; апробовано механізми розгортання VLE, виконання та автоматичної перевірки PPET; проведено вимірювання часу, який потрібен системі для розгортання та вимірювання пропускної здатності системи; проведено оцінку сформованості практичних навичок; пораховано індекс персоналізації.

4.4.2 Дослідження ефективності генерації PPET на основі LTDL

Мета та гіпотеза експерименту. Експеримент проводиться з метою кількісно оцінити вплив використання розробленої домен-орієнтованої мови LTDL та модуля синтаксичної валідації на якість, безпомилковість та здатність до автоматичного розгортання згенерованих навчальних завдань порівняно з базовою генерацією за допомогою LLM. Очікується, що застосування формальної граматики LTDL як проміжного шару між LLM та VLE дозволить суттєво знизити рівень структурних та логічних помилок ("галюцинацій" моделі) та підвищити відсоток завдань, що успішно розгортаються без ручного втручання викладача. Додаткова гіпотеза полягає в тому, що використання мультиагентного підходу (перевірка генерації за допомогою другої LLM) не здатне повністю усунути ці помилки через ймовірнісну природу LLM, що додатково обґрунтовує необхідність застосування формальної мови LTDL.

Умови експерименту. Для проведення експерименту використано наступну модель LLM gemini-2.5-flash та тестову вибірку із 50 унікальних запитів (промтів) на створення практичних завдань з 10 інженерних дисциплін (по 5 на кожену дисципліну).

Для порівняльного аналізу досліджувалися 3 різних підходи, що схематично зображено на рисунку 4.16.

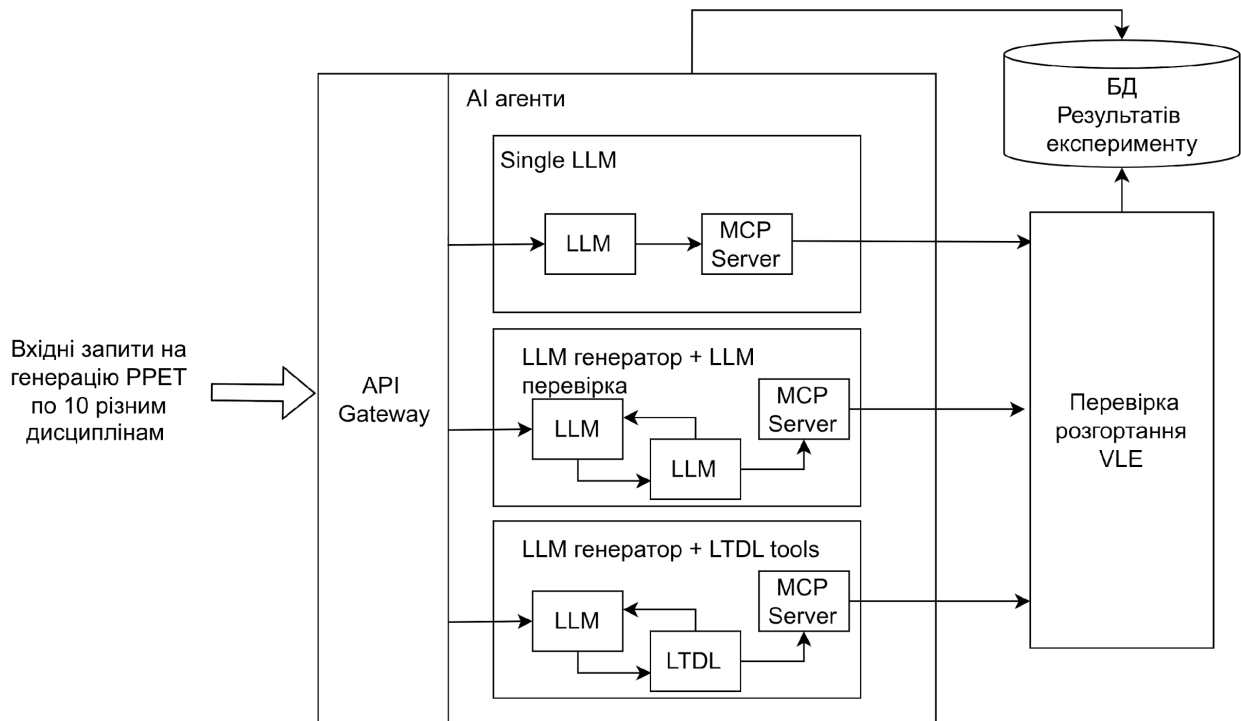


Рис. 4.16. Схематичне зображення умов експерименту.

Підхід А (LLM генерація). Одиночна LLM отримує детальну інструкцію, але не отримує жодних прикладів або формальних схем, і покладається лише на свої внутрішні ваги. LLM генерує конфігураційні файли та опис завдання безпосередньо у вигляді звичайного тексту.

Підхід Б (LLM-генератор + LLM-перевірка): Мультиагентний підхід, який відрізняється від попереднього наявністю другої моделі LLM, що виступає в ролі верифікатора і перевіряє результат генерації першої моделі, намагаючись виправити помилки.

Підхід В (LLM+LTDL), де LLM генерує завдання у суворому форматі розробленої мови LTDL. Отриманий код проходить етап автоматичного парсингу та перевірки валідатором на відповідність формальній граматиці, і тільки в разі успішної валідації відбувалася спроба розгортання.

Для забезпечення об'єктивності порівняння і для уникнення штучного

заниження результатів в Підході А і Б використовувався розширений системний промт (наведено нижче), який явно вимагав від моделі формування всіх необхідних компонентів завдання (інструкцій, конфігурацій, тестів та змінних) у структурованому форматі Markdown. Застосований системний промт для експерименту наведено в додатку 3.

Хід експерименту. Експеримент проводився в автоматизованому режимі із застосування спеціально написаного Python-скрипту, який для кожного із 50 тестових промтів відправляв запит до API обраної LLM за Підходом А, Підходом Б, Підходом В. Система логувала кожен етап: код відповіді LLM, час генерації, наявність помилок парсингу, фіксувався кінцевий статус розгортання (успіх/помилка) для розрахунку DSR.

Результати Підходів А і Б парсилися, виконувався пошук блоків коду в Markdown та передавалися для спроби розгортання VLE на основі згенерованих файлів. Оцінки виставлялися на основі логів розгортання, виконання скриптів перевірки та загального ревью згенерованого тексту завдання. Результати Підходу В пропускалися через парсер LTDL. Оцінки виставлялися на основі логів транслятора та розгортання VLE.

Метрики оцінювання. Для уникнення необ'єктивності бінарної оцінки («працює / не працює»), якість генерації кожного j -го завдання оцінювалася за допомогою багатокритеріальної моделі з ваговими коефіцієнтами. Інтегральний показник якості генерації $Q_j \in [0, 1]$ розраховувався за формулою:

$$Q_j = \sum_{i=1}^4 w_i \cdot c_{ij},$$

де w_i – ваговий коефіцієнт важливості i -го структурного елемента завдання для його автоматичного розгортання представлені в Таблиці 4.2;

$c_{ij} \in \{0; 0.5; 1\}$ – дискретна оцінка коректності i -го елемента для j -ї генерації

(де 1 – повна відповідність, 0.5 – наявність некритичних помилок, що

піддаються автоматичному відновленню, 0 – фатальна помилка або відсутність

компонента).

Таблиця 4.2

Вагові коефіцієнти структурних елементів завдання

Назва елементу	Ваговий коефіцієнт	Опис
Текстовий опис та інструкції	$W_1 = 0.15$	Оцінюється повнота та зрозумілість умови завдання для студента.
Інфраструктурний код VLE	$W_2 = 0.45$	Найкритичніший елемент для конфігурації середовища. Оцінюється синтаксична правильність та готовність коду до виконання.
Логіка автоматичної перевірки	$W_3 = 0.30$	Валідність скриптів/предикатів, що підтверджують виконання завдання студентом.
Параметризація та динамічні змінні	$W_4 = 0.10$	Наявність параметрів (наприклад, згенерованих портів чи паролів) для забезпечення параметризації та запобігання академічній недоброчесності.

Головною результуючою метрикою експерименту виступав коефіцієнт успішного розгортання (*Deployment Success Rate, DSR*) – відсоток завдань з вибірки, які були успішно ініціалізовані у VLE і пройшли базовий тест доступності (health-check).

Оцінка результатів. У Таблиці 4.3 наведено порівняльний аналіз середніх значень якості генерації та метрики DSR для всіх підходів, а в Таблиці 4.4 – середнє значення інтегрального показника якості генерації по обраним дисциплінам для всіх підходів. Оцінка якості генерації кожного з 50 завдань для визначення середнього значення інтегрального показника якості генерації та відсотку успішного розгортання наведена в додатку I.

Таблиця 4.3

Порівняльний аналіз підходів А, Б, В.

Метрика оцінювання	Підхід А (Базовий: одиначна LLM)	Підхід Б (Мультиагентний: LLM+LLM)	Підхід В (Запропонований: LLM+LTDL)
Середнє значення Інтегрального показника якості генерації Q	0,39	0,78	0,90
Успішне розгортання (DSR)	38 %	74 %	96 %

Таблиця 4.4

Середнє значення інтегрального показника якості генерації по обраним дисциплінам

Дисципліна	Підхід А (Базовий: одиначна LLM)	Підхід Б (Мультиагентний: LLM+LLM)	Підхід В (Запропонований: LLM+LTDL)
Операційні системи	0,38	0,78	0,88
Організація баз даних	0,54	0,72	0,88
Комп'ютерні мережі	0,55	0,47	0,94
Веб-програмування	0,49	0,91	0,91
Промислові комп'ютерні мережі та інтерфейси	0,52	0,53	0,83
Основи DevOps та хмарних технологій	0,48	0,85	0,87
Безпека ІС	0,34	0,93	0,9
Системне програмування	0,23	0,94	0,96
Розподілені системи та паралельні обчислення	0,11	0,92	1
Розгортання систем машинного навчання	0,27	0,71	0,85

Лінійна діаграма значень інтегрального показника якості генерації представлена на рисунку 4.17.

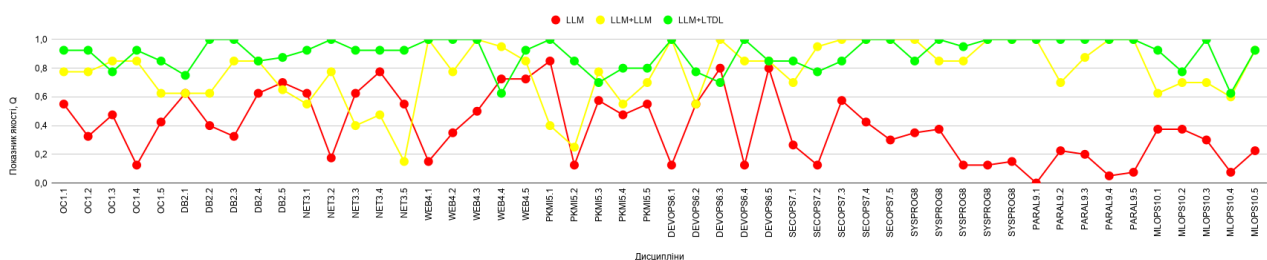


Рис.4.17. Лінійна діаграма значень інтегрального показника якості генерації

Аналіз отриманих даних підтверджує висунуту гіпотезу. Використання чистого генеративного підходу (Підхід А) призводить до низького показника інтегральної якості. Це зумовлено схильністю будь-якої LLM до синтаксичних помилок у конфігураційних файлах та втрати логічних зв'язків між змінними. Додаткова перевірка з боку іншої LLM (Підхід Б) суттєво покращує показник, хоча існують

помилки, які друга LLM часто не здатна детерміновано виправити. З урахуванням використання чіткого детального системного промту в Підходах А і Б помилки, допущені моделлю за таких умов, свідчать не про низьку якість промту, а про фундаментальні обмеження чистого генеративного підходу при синтезі складних інфраструктурних зв'язків. Це робить неможливим надійне автоматичне розгортання. Впровадження домен-орієнтованої мови LTDL (Підхід В) успішно мінімізує ці ризики. Делегування контролю за синтаксисом дозволяє суттєво підвищити якість генерації та досягти показника успішного розгортання на рівні 96%, що доводить високу надійність розробленої ІТ для застосування в реальному процесі персоналізованого навчання студентів інженерних спеціальностей.

4.4.3 Порівняння AI моделей для генерації

Мета та гіпотеза експерименту. Метою експерименту є порівняльна оцінка ефективності різних моделей GenAI для автоматичного створення PPET у форматі LTDL, а також дослідження впливу складності вхідного контексту (цілі лабораторної роботи або повний текст завдання) на якість, стабільність та ресурсоемність процесу генерації. Додатковою метою є перевірка можливості використання LLM-моделей у повністю автоматизованому режимі генерації практичних завдань у межах запропонованої ІТ технології. Гіпотезою експерименту є припущення, що сучасні GenAI-моделі здатні автоматично генерувати синтаксично коректні описи практичних завдань у форматі LTDL, однак ефективність генерації залежить від архітектури моделі, складності вхідного контексту та режиму виконання (людино-керований або повністю автоматизований). Передбачається, що використання новіших LLM-моделей забезпечує зменшення кількості ітерацій, необхідних для отримання синтаксично коректного результату, але може супроводжуватися збільшенням витрат обчислювальних ресурсів, часу виконання та споживання токенів.

Умови експерименту. Для емпіричного вивчення потенціалу GenAI в області генерування завдань проведено два експерименти з використанням різних LLM-моделей, завдань і режимів виконання. В перший експеримент була залучена

людина, в другому функції агентів-генераторів завдань та агентів-перевірятьників синтаксису виконував GenAI.

Завданням, поставленим перед різними моделями GenAI, було створення практичного завдання з налаштування сервера IP-телефонії для навчальної дисципліни «Сучасні телекомунікаційні системи та IP-телефонія» для студентів магістратури спеціальності 123 «Комп'ютерна інженерія». Для цього обрано перше лабораторне завдання з методичних вказівок курсу [170]. Кожен із двох експериментів проводився у 2 етапи (фази): на першому етапі РРЕТ створювалися на основі цілей лабораторної роботи; на другому - на основі повного тексту лабораторної роботи. Оцінка результатів надається як по ходу проведення експерименту, бо від результатів першої фази залежить проведення другої фази, так і підсумковим висновком в кінці опису.

Хід експерименту 1. Автоматичне генерування з участю людини. У цьому експерименті завдання (запит користувача) і контекст-промпт (запит системи, граматичні файли та приклади) були передані моделям зовнішнім актором через веб-інтерфейс. Результати роботи моделі зберігалися у файлі, після чого інструмент ANTLR запускався вручну для перевірки синтаксису. У випадках виявлення синтаксичних помилок результати аналізу поверталися до моделі для повторного генерування. Для цілей цього дослідження були обрані загальнодоступні моделі, розроблені провідними постачальниками AI, а саме: ChatGPT від OpenAI, Gemini від Google, Claude від Anthropic та Grok від xAI.

Результати фази 1 представлені в Таблиці 4.5.

Таблиця 4.5

Результати експерименту 1, фаза 1.

Model	Stages	Preconditions	Checks	Iterations
Gemini 2.5	7	6	16	4
Grock 3 Fast	6	5	12	2
Grock 4	6	15	19	2
Claude Sonnet 4	10	3	36	3
ChatGPT 5	6	0	10	2

Моделі продемонстрували помітні відмінності в їх здатності генерувати РРЕТ, базуючись виключно на цілях лабораторної роботи. Хоча деякі моделі (наприклад, Claude Sonnet 4) потребували більшої кількості ітерацій для досягнення синтаксично правильних результатів, інші (наприклад, Grok 3 Fast і ChatGPT 5) досягли задовільних результатів з меншою кількістю спроб. Ці відмінності підкреслюють вплив архітектури моделі та даних для навчання на ефективність виконання завдань.

На основі цих попередніх результатів у другій фазі експерименту запропоновано більш складний сценарій, в якому моделям поставлено завдання створити РРЕТ на основі повного тексту лабораторної роботи. Результати фази 2 наведено в Таблиці 4.6.

Таблиця 4.6

Результати експерименту 1, фаза 2.

Model	Stages	Preconditions	Checks	Iterations
Gemini 2.5	11	0	9	6
Grock 3 Fast	9	9	30	5
Grock 4	5	5	15	3
Claude Sonnet 4	10	10	35	3
ChatGPT 5	11	2	29	10
Основний файл	4	8	30	

Результати фази 2 (Таблиця 4.6) показують, що складність завдання, яке вимагало створення повного практичного завдання, призвела до значного збільшення кількості ітерацій та перевірок валідації в більшості моделей порівняно з першою фазою. Примітно, що Claude Sonnet 4 та ChatGPT 5 продемонстрували високу стійкість у повторному генеруванні, хоча й за рахунок більшої кількості ітерацій, тоді як моделі Grok досягли більш лаконічних результатів з меншою кількістю повторних спроб. Gemini 2.5 продемонструвала стабільну продуктивність на обох фазах, хоча і з помірними вимогами до ітерацій. У цілому, результати свідчать про те, що хоча всі оцінені моделі здатні генерувати синтаксично правильні

та педагогічно релевантні результати, їх ефективність та послідовність значно варіюються залежно від рівня складності завдання.

Хід експерименту 2. Автоматичне генерування за допомогою AI API. Щоб оцінити можливості GenAI в повністю автоматизованому режимі, запропонований робочий процес AI був реалізований за допомогою платформи з відкритим кодом Flowise. Flowise – це середовище з низьким рівнем кодування для проєктування конвеєрів AI-агентів та їх інтеграції із зовнішніми інструментами. Його веб-інтерфейс, побудований на бібліотеці ReactFlow, дозволяє інтуїтивно та інтерактивно створювати AI-потoki. Для експериментів були обрані моделі Gemini AI, які виконували функції агентів-генераторів завдань та агентів-перевірять синтаксису. Зокрема, протестовано три версії: gemini-1.5-flash, gemini-2.0-flash та найновіша безкоштовна модель gemini-2.5-flash-preview-05-20. Сервіс AI від Google був обраний через наявність безкоштовних ключів API. Сервер SSH MCP був використаний як допоміжний інструмент для генерації файлів LTDL і виклику парсера ANTLR4 для перевірки синтаксису. Реалізація потоку AI показана на рисунку 4.18.

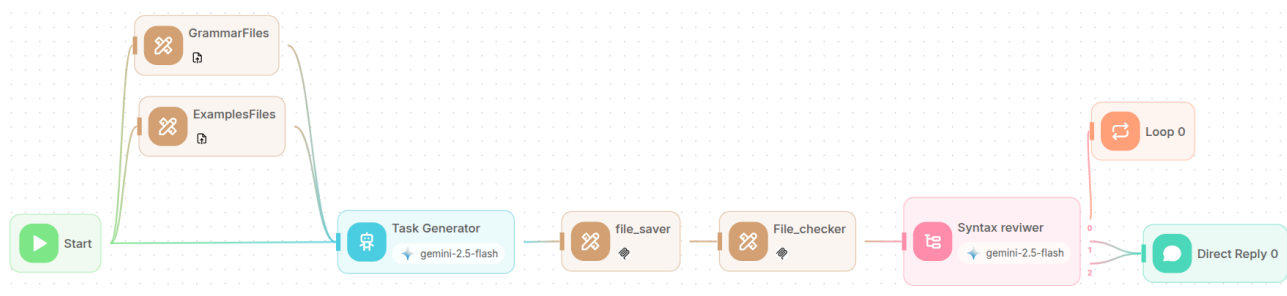


Рис. 4.18. Реалізація потоку AI у Flowise

Результати фази 1 підсумовані в Таблиці 4.7. Оцінка враховувала такі параметри: версія моделі, кількість ітерацій, рівень успішності у створенні документів LTDL без синтаксичних помилок, середнє споживання токенів на ітерацію AVG(T) та середній час AVG(t), витрачений на ітерацію (у секундах).

Таблиця 4.7

Результати експерименту 2, фаза 1.

Model	Iterations	Success	AVG(T)	AVG(t)
gemini-1.5-flash	>15	-	15501	9
gemini-2.0-flash	>15	-	20043	9
gemini-2.5-flash-preview-05-20	5	+	7803	8

Як показано в Таблиці 4.7, тільки модель gemini-2.5-flash-preview-05-20 успішно генерувала синтаксично правильні результати в межах ітераційного обмеження, вимагаючи менше спроб у порівнянні з попередніми версіями Gemini. Однак вона також потребувала більшого використання токенів і дещо довшого часу обробки. Кількість API-запитів на цьому етапі показано на рисунку 4.19.

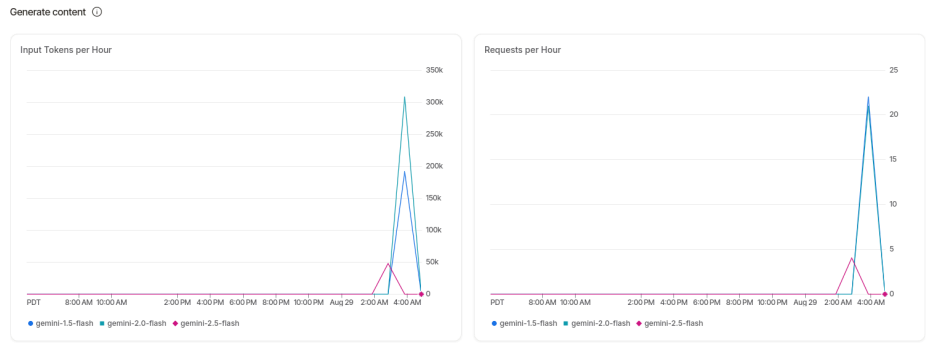


Рис. 4.19. Gemini API використання у фазі 1.

Для фази 2 експерименту 2 потік AI розширено додатковим компонентом завантаження файлів, щоб можна було завантажити повний текст лабораторного завдання. Результати наведено в Таблиці 4.8.

Таблиця 4.8

Результати експерименту 2, фаза 2.

Model	Iterations	Success	AVG(T)	AVG(t)
gemini-1.5-flash	>15	-	12603	7
gemini-2.0-flash	>15	-	18088	9
gemini-2.5-flash-preview-05-20	4	+	79702	33

На цьому етапі модель `gemini-2.5-flash-preview-05-20` знову перевершила попередні моделі, успішно генеруючи дійсні результати зі значно меншою кількістю ітерацій. Однак це відбулося за рахунок збільшення споживання токенів і часу обробки порівняно з етапом 1. Використання API на цьому етапі показано на рисунку 4.20.

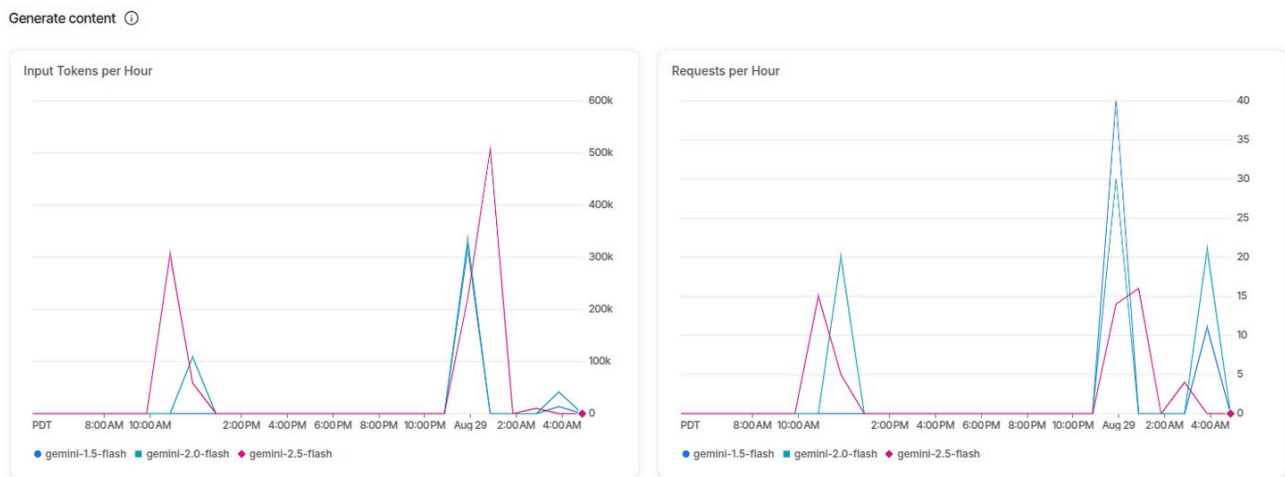


Рис. 4.20. Gemini API використання у фазі 2.

Висновки. Отримані результати експериментів підтвердили можливість використання GenAI для автоматичного створення PPET у форматі LTDL, що є важливим для реалізації запропонованої ІТ. Проведене порівняння показало, що ефективність генерації суттєво залежить від архітектури моделі та складності вхідного контексту. Більш сучасні моделі продемонстрували вищу стабільність у досягненні синтаксично коректних результатів та потребували меншої кількості ітерацій повторної генерації, хоча інколи це супроводжувалося більшим споживанням обчислювальних ресурсів. Також встановлено, що використання повністю автоматизованого режиму генерації з залученням AI-агентів і зовнішніх інструментів валідації дозволяє реалізувати замкнений цикл створення та перевірки завдань без участі людини. Отримані результати підтверджують доцільність інтеграції GenAI з формальними механізмами валідації у межах запропонованої ІТ для автоматизованої генерації PPET.

4.4.4 Натурний експеримент з апробації механізмів розгортання VLE, виконання та автоматичної перевірки PPET

Мета та гіпотеза експерименту. Метою експерименту є дослідження роботи системи в режимі автоматичної перевірки персоналізованих завдань, з попереднім розгортання відповідних VLE для такого виконання, статистичний аналіз отриманих результатів та перевірка кореляції результатів практичного тесту з результатами теоретичних тестів по обраній темі. Об'єктом експерименту є процес виконання PPET. Перевіряється, як використання запропонованих механізмів, які поєднують автоматичне розгортання VLE та автоматичну перевірку PPET, дозволить забезпечити об'єктивне та масштабоване оцінювання практичних навичок студентів, та оцінити використання мови LTDL для опису конфігурацій середовища. Для моніторингу ходу експерименту використано методи спостереження за реальним навчальним процесом, аналіз результатів роботи системи методами статистичного аналізу та опитування учасників.

Умови експерименту. В експеримент загалом було залучено 252 студента. Студенти поділені на 8 груп в середньому по 30 студентів (поділ груп зумовлений обмеженими можливостями навчального серверу).

Для експериментальної перевірки сформовано текстове PPET з вивчення UNIX-like операційної системи. Завдання покриває запуск команд в командному рядку; налаштування середовища; роботу з файлами, директоріями та архівами; роботу з файловими системами та потоками виводу. Також створено інструкції розгортання та налаштування VLE та інструкції системі для перевірки процесу виконання завдання. У якості VLE використовувалася VM на базі Oracle Enterprise Linux 9.1, у якості віртуалізатора використовувався VirtualBox 7.0.1, для розгортання конкретного екземпляра VLE - vagrant 2.2.4. Параметри навчального серверу: 64 Gb RAM, CPU 16 cores Intel(R) Xeon(R) E-2288G, 3.70GHz. Сценарій виконання завдання показано на рисунку 4.21, розроблений сценарій має 4 незалежні треки. Етапи, результати яких винесені в інструкції для перевірки прогресу, виділені сірим кольором.



Рис. 4.21. Сценарій виконання PPET з основ CLI в ОС UNIX.

Інструкції для модуля автоматичної перевірки складаються з 9 етапів і представлені в Таблиці 4.9.

Таблиця 4.9

Опис інструкцій для етапів автоматичної перевірки.

#	Текстовий код	Очікуваний результат	Тип перевірки
1	login	Login to VLE with credentials	True-False
2	passwd	Password changed	Hash
3	resdir	Directory is created and all necessary files are in it	True-False
4	resarch	Archive with necessary files is created	True-False, Hash
5	diskmnt	Correct disk mounted to target directory	True-False
6	killpic	Process is deactivated	True-False
7	unalias	Alias is removed and there is a correct command in history	True-False
8	sizeelog	File contains output	True-False
9	sizeerr	File contains errors	True-False

Хід експерименту. Перевірка етапів виконувалася кожні 5 хв., послідовно для кожного VLE з урахуванням залежності етапів один від одного для оптимізації процесу автоматичної перевірки. Успішний результат виконання конкретного етапу фіксувався у вигляді відмітки часу, а загальний відсоток виконання всього завдання збільшувався на відповідний відсоток. Для спрощення кожен етап має однакову вагу в 11% від загального, як результат ділення 100% на кількість етапів. В результаті для кожного студента отримано набір часових міток та загальний прогрес у відсотках.

Поточний прогрес у відсотках відображався в командному рядку кожного VLE, щоб студенти мали можливість відслідковувати свій персональний прогрес.

Спостереження за ходом експерименту та процесом виконання завдання викладачами відбувалось за допомогою веб-сторінки сформованої модулем обробки результатів для відслідковування прогресу; zoom-конференції, в рамках якої в окремій кімнаті студент транслює екран та відео з камери; утиліти nmon та nmonchart для фіксації навантаження на навчальний сервер. Прогрес виконання відображався у вигляді таблиці, рядки – це персональне VLE, а стовпчики - час, коли було завершено проміжний етап завдання та загальний прогрес. Приклад, такої таблиці показано на рисунку 4.22.

Host	login	passwd	resdir	resarch	diskmnt	killpic	unalias	sizeog	sizeerr	Conditional checks	Time delta	Progress
client2	10:15	10:20	11:05	11:05	11:20	12:30	11:25	11:35	11:35	–	2:15	100%
client3	10:15	10:20	11:45	11:45	11:50		11:55	12:00	12:00	–	1:45	88%
client4	10:15	10:15	11:15	11:20	11:45	12:15	11:45	12:05	12:05	–	2:00	100%
client5	10:15	10:15	11:10	11:15	11:30	12:30	11:35	11:45	11:45	–	2:15	100%
client6	10:15	10:15	11:05	11:10	11:30		11:35	11:35	12:15	–	2:00	88%
client7	10:15	10:20	10:55	11:00	11:35	11:45	11:20	11:25	11:35	–	1:30	100%

Рис. 4.22. Веб-сторінка відслідковування прогресу з часовими мітками.

Оцінка результатів. В ході експерименту були отримані такі дані: повністю всі етапи виконали 71 студент, що є 28.2% від загальної кількості студентів, мінімальний час, який знадобився для виконання всіх етапів дорівнює 35 хвилинам, максимальний 2 години 21 хвилина, середній значення 1:46, а медіана 1:50. Також для подальшого аналізу результатів виділено логічні групи за результатом проведеного теоретичного тексту по обраним темам: нижче 60 відсотків, 60-89 відсотків та 90+ відсотків. Результати представлені на рисунку 4.23.

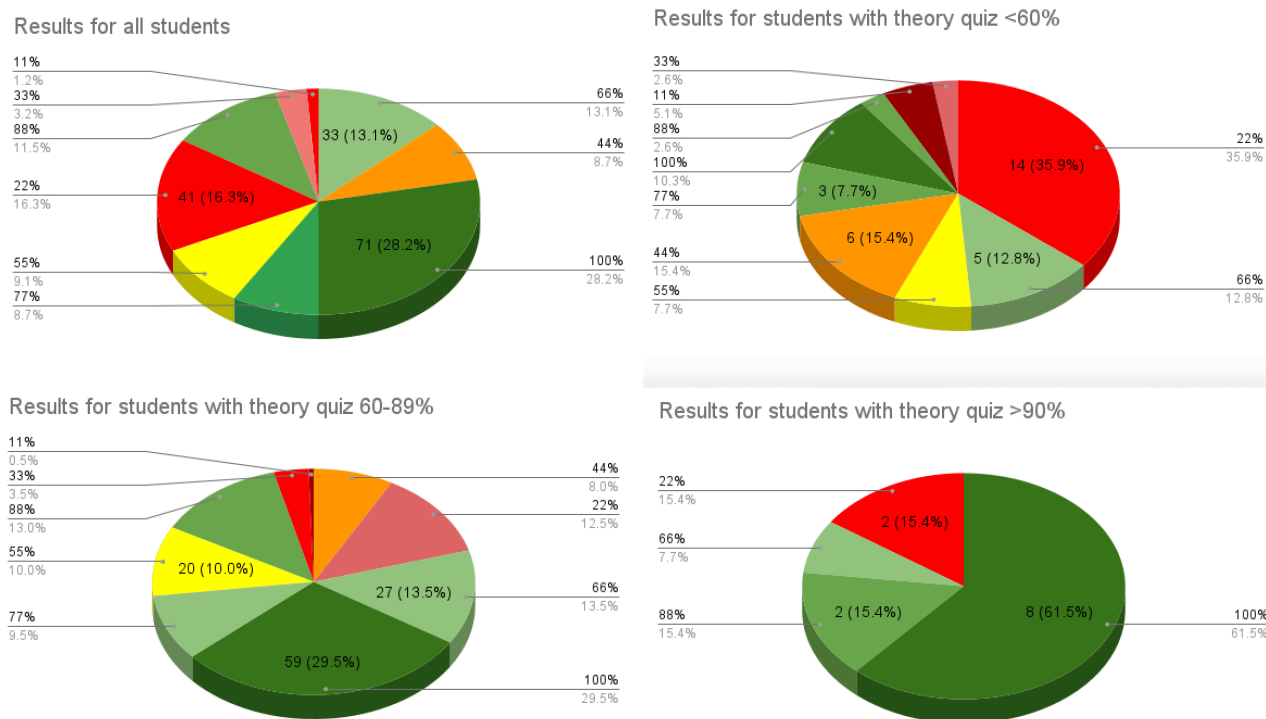


Рис. 4.23. Результати виконання студентами всіх етапів PPEP.

На основі отриманих даних проведено дослідження кореляції результатів теоретичних тестів та результатів виконання практичного завдання по обраним темам [171]. Визначено, що теоретичні знання та практичні навички (робота в середовищі) – це різні компетенції. Традиційні теоретичні тести не можуть надійно оцінити практичну підготовку, що посилює актуальність запропонованої інформаційної технології.

Для оцінки ефективності розроблених механізмів розгортання проведено вимірювання часу, який потрібен системі для підготовки середовища для студента, та вимірювання пропускну здатності системи, та введено наступні метрики продуктивності: час розгортання та пропускну здатність системи [172].

Загальний час розгортання VLE використовується для оцінки масштабованості системи при різній кількості студентів і оцінюється за формулою (3.1).

Статистичні дані щодо часу розгортання окремих екземплярів VLE ($n = 30$) наведено в таблиці 4.10.

Таблиця 4.10

Розгортання екземплярів VLE.

Metric	Min (s)	Max (s)	Mean (s)	Std. Dev. (s)
T_{parse}	2	3	2,5	0,71
$T_{translate}$	4	6	5	1,41
$T_{provision}$	210	420	315	148,49
$T_{validation}$	12	28	20	11,31
T_{total}	228	457	337,5	160,51

Аналіз часових характеристик процесу розгортання екземплярів VLE показав, що домінуючий внесок у загальний час виконання припадає на етап розгортання (*provisioning*), тоді як етапи парсингу, трансляції та валідації, пов'язані із використанням розробленого методу розгортання і мови LTDL, формують незначне обчислювальне навантаження. Процеси розгортання середовища залежать від інфраструктурних чинників, а не від логічних компонентів методу.

Пропускна здатність системи (Θ): $\Theta = \frac{N_{VLEs}}{\Delta t}$,

де N_{VLEs} – кількість VLE, які система може успішно розгорнути (або вже розгорнула) за проміжок часу Δt , характеризує масштабованість рішення.

Після завершення експерименту проведено опитування, в якому прийняло участь 186 респондентів з 252. Були поставлені наступні питання.

А. Оцініть зручність того, що VLE розгортається автоматично і не треба робити додаткових дій по розгортанню та налаштуванню середовища? В разі наявності - залиште коментар щодо цієї можливості.

а) так, це дуже зручно - 89% б) ні, не зручно - 11%

Серед коментарів студентів можна відмітити наступний: *"так, це дуже зручна опція, що практика запускається на окремому сервері, до якого треба приєднатись по ssh. Раніше я витрачав багато часу на завантаження, розгортання віртуальної машини. Часто виникали помилки, пов'язані з версіями образу та моєю архітектурою"*. Разом з тим, студенти, які не були задоволені автоматизованою системою відмічали відсутність повного контролю за VLE і наявністю заважаючих елементів.

В. Чи отримали ви користь з того, що бачили прогрес виконання завдання в реальному часі? В разі наявності - залиште коментар щодо цієї можливості.

а) так, це зручно, допомагає триматись ритму – 74% б) так, це корисно, але стресово – 26%

Серед коментарів студентів, які обрали варіант Б, можна відмітити наступний: *"Ліміт часу заважав зосередитись на самому завданні, і я думав тільки про час, хоча врешті реїт - це гарна можливість покращити свою стресостійкість"*.

С. Чи сподобалось вам, що результат виконання можна побачити одразу і не треба чекати перевірку завдання викладачем?

а) Так, це дуже корисно – 100% б) ні, мені це не стало в нагоді – 0%.

В питанні автоматичного і миттєвого отримання результату були одностайні, але разом з тим відмічали недостатню кількість часу для виконання завдання в ході експерименту.

Моніторинг навантаження на навчальний сервер показано на рисунку 4.24.

Високе навантаження відмічалось під час запуску VM та піками навантаження, пов'язаними з виконанням завдань, які потребують багато процесорної потужності. Основне навантаження припадає на період часу від +20 хвилин до +90 хвилин від початку виконання завдання.

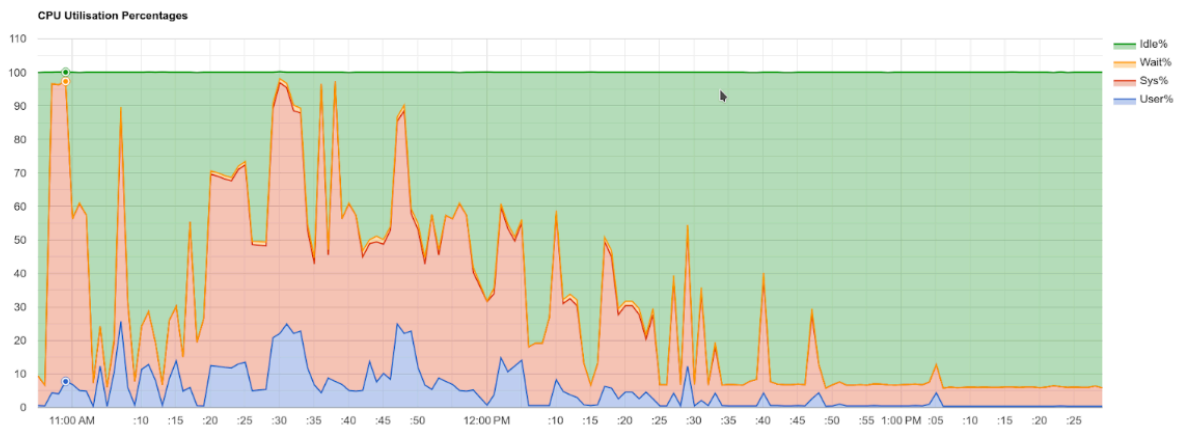


Рис. 4.24. Навантаження на навчальний сервер під час виконання PPET.

4.4.5 Оцінка сформованості практичних навичок

Мета і гіпотеза експерименту. Метою є оцінювання можливості формалізованого визначення сталості практичних навичок студентів у процесі виконання PPET, перевірка доцільності використання запропонованої метрики Sustainability Index (SI) та оцінювання ефективності запропонованого підходу до формування сталих практичних навичок студентів шляхом використання параметризованих варіацій PPET. В основу експерименту покладено гіпотезу про те, що використання множини параметризованих варіацій практичного завдання дозволяє а) визначити показник SI та б) отримати підвищення ефективності виконання контрольного PPET. **Умови експерименту.** Експеримент проводиться на ізолюваних VLE, що автоматично розгортаються на основі опису PPET мовою LTDL. Учасниками експерименту є слухачі публічних курсів з адміністрування операційних систем на базі Linux, розбиті на 2 групи по 30 студентів: контрольна група виконує одиничне контрольне завдання, експериментальна – виконує множину параметризованих варіацій завдання перед виконанням контрольного PPET. Оцінювання результатів здійснюється автоматично. Наявність теоретичних знань і готовності до проходження практичного тесту перевірялось шляхом проведення тестування та інтерв'ю.

Хід експерименту. На першому етапі для кожного студента формується

практичне завдання. У контрольному підході студенту генерується одна задача, тоді як в експериментальному підході генерується множина її параметризованих варіацій із використанням мови LTDL. На другому етапі студенти виконують отримані завдання у середовищі VLE. Для кожної варіації фіксується результат виконання (r_i) та коефіцієнт автономності (a_i). Рівень складності (w_i) прийнято рівним 1, оскільки всі параметризовані варіації завдання мають однакову структурну складність, що дозволяє виключити вплив цього фактору на обчислення показника SI.

На третьому етапі для студентів експериментальної групи обчислюється інтегральний показник сталості практичної навички – Sustainability Index (SI) на основі виконання 5 варіацій завдання. Формулу розрахунку наведено в розділі 3.5. Результати оцінювання виконання практичних завдань, розрахунку індексу, середні та стандартні відхилення наведені в Таблиці 4.11.

Таблиця 4.11

Результати оцінювання виконання PPET і розрахунку індексу

Група	Кількість варіацій завдань	Середній результат виконання (r_i)	Std (r_i)	Середній SI	Std (SI)
Контрольна	1	0,68	0,21	-	-
Експериментальна	5	0,81	0,12	0,78	0,09

Примітка. Для контрольної групи показник SI не обчислюється, оскільки оцінювання сталості навичок потребує виконання множини параметризованих варіацій завдання.

Результати експерименту свідчать, що запропонований підхід дозволяє отримати кількісну оцінку сталості практичних навичок. Для контрольної групи показник SI не обчислюється, оскільки метрика передбачає аналіз множини варіацій і не може бути застосовано до одиничного результату. Слід зазначити, що на даному етапі дослідження відсутні безпосередні альтернативні метрики, з якими можна здійснити коректне порівняння показника SI. Одним із сучасних підходів до оцінювання професійних навичок інженерів є EPSA 2.0 [174], який базується на аналізі виконання студентами комплексних сценаріїв та експертному оцінюванні результатів за допомогою рубрик. Даний підхід орієнтований на оцінювання

професійних (*soft*) компетентностей, зокрема критичного мислення, комунікації та прийняття рішень у складних умовах. Разом з тим, EPSA 2.0 має ряд обмежень, зокрема залежність від суб'єктивного експертного оцінювання, відсутність автоматизації та орієнтацію на аналіз одиничного виконання завдання. Методика Competence Retention Analysis Technique [175] орієнтована на прогнозування того, які саме компоненти навичок втрачаються швидше за все. Освітні онлайн-платформи, такі як Coursera [176], використовують різні аналітичні метрики, зокрема завершуваність (*completion rate*), оцінки за завдання (*score*), кількість спроб виконання (*attempts*) та час виконання (*time-on-task*), що дозволяє оцінювати активність та результати навчання, однак вони орієнтовані переважно на аналіз окремих спроб виконання або агрегованих результатів. Загалом, існуючі підходи не забезпечують формалізованого оцінювання сталості практичних навичок як здатності студента стабільно відтворювати результат у змінних умовах. Обчислення SI в межах даного дослідження формує базу для подальшого порівняння та аналізу динаміки сформованості практичних навичок у майбутніх роботах.

Висновки. Використання множини параметризованих варіацій завдань забезпечує вищі результати виконання та можливість формалізованого оцінювання сталості практичних навичок. Метрика SI є придатним інструментом для такого оцінювання. Традиційний підхід, що базується на виконанні одиничного завдання по темі, не дозволяє оцінити сталість сформованих навичок. Запропонований підхід розширює можливості оцінювання результатів, забезпечуючи перехід від аналізу окремого виконання завдання по темі до оцінювання сформованості практичної навички як стійкої здатності до розв'язання схожих задач у змінних умовах. Крім того, значення показника успішності виконання PPET (r_i) опосередковано відображає рівень сформованості практичних навичок.

4.4.6 Визначення індексу персоналізації запропонованої інформаційної технології

Для кількісної оцінки рівня персоналізації використано узагальнену метрику –

індекс персоналізації (Personalization Index, PI), який враховує рівні персоналізації [38], реалізовані в даній системі/платформі/підході. Математично індекс персоналізації практичного завдання представлений як:

$$PI = \sum_{i=1}^N W_i \cdot L_i$$

де N – кількість рівнів персоналізації [38];

L_i – індикатор реалізації рівня i в системі (1 - реалізовано, 0 - ні);

W_i – вага рівня, пропорційна його складності, згідно таблиці 4.10.

PI розраховується як сума вагових коефіцієнтів реалізованих в даному завданні рівнів персоналізації і дозволяє кількісно порівнювати PPET між собою за глибиною персоналізації. Розрахунок PI наведено в Таблиці 4.12.

Таблиця 4.12

Розрахунок PI для оцінки рівня персоналізації PPET

	L1	L2	L3	L4	L5	L6	L7	L8	L9	PI
Ваговий коефіцієнт, $W(i)$	0,03	0,04	0,06	0,10	0,12	0,15	0,16	0,16	0,18	
SERA [44]	1	0	1	0	1	0	0	0	0	0,21
CodeCombat [56]	1	1	1	0	1	0	1	0	0	0,40
Labster [52]	1	1	0	1	1	0	1	1	0	0,60
JupyterHub [41]	1	1	1	1	1	0	0	0	0	0,34
Codio [51]	1	1	1	1	1	1	0	0	0	0,49
CPPE [68]	1	1	0	0	0	0	1	1	0	0,39
Розробка автора	1	1	1	1	1	1	1	1	0	0,81

4.5 Висновки до розділу IV

У розділі описане вирішення актуального наукового завдання щодо створення інформаційної технології, яка комплексно автоматизує процеси генерації унікальних (параметризованих) і персоналізованих практичних завдань, автоматичного розгортання індивідуальних VLE для виконання цих завдань та їх подальшої автоматичної перевірки. Також у рамках роботи безпосередньо розроблена

підсистема розгортання VLE та оцінювання PPET та її архітектура. Розроблена підсистема – реалістична, інноваційна, добре структурована, з високим потенціалом для трансформації освітнього процесу на кафедрі. Її впровадження вимагає поступової реалізації у 4 фазах: підготовка інфраструктури, навчання викладачів та студентів, оцінка і аналіз результатів впровадження, масштабування на інші дисципліни/кафедри.

Ключовими ознаками, що дозволяють вважати результати дослідження інформаційною технологією персоналізації практичних завдань у підготовці інженерних спеціалістів, є:

1. Функціональна модель, яка описує процеси створення, розгортання, виконання та перевірки PPET і формалізує їх повний життєвий цикл як єдиний безперервний конвеєр із замкненим адаптивним циклом зворотного зв'язку, що формує уніфікований підхід до організації практичної підготовки з інженерних спеціальностей.

2. Формальна модель, яка описує структуру практичного завдання як об'єкт, що включає набір взаємопов'язаних компонентів, які визначають цілі завдання, вхідні дані, методи виконання, інструменти реалізації, середовище виконання та критерії оцінювання. Такий підхід дозволяє перейти від неформального текстового опису практичних завдань до їх формальної специфікації, що створює основу для автоматизованої генерації, виконання та перевірки завдань у DLE.

3. На основі моделей розроблено домен-специфічну мову LTDL, яка забезпечує машинно-читаний опис PPET та використовується для автоматизації процесів генерації завдань, розгортання VLE та перевірки результатів. LTDL є операційною реалізацією частини формальної моделі PPET і дозволяє формалізувати PPET, конфігурацію VLE та критерії оцінювання результатів.

4. Реалізація персонального AI-асистента в межах запропонованої інформаційної технології здійснена із використанням середовища Flowise. Асистент побудований за принципом RAG і функціонує як окремий агент. Для нього на основі наданої тематики та методичних вказівок до практичних завдань сформовано

векторне представлення (embeddings), які збережені у векторній базі даних Qdrant, що дозволяє здійснювати семантичний пошук релевантних фрагментів навчального матеріалу відповідно до запиту студента та передавати їх у контекст генерації.

5. Інформаційна технологія включає метод автоматизованої генерації та параметризації, що поєднують можливості GenAI з формальними обмеженнями мови LTDL. Такий підхід забезпечує контрольовану генерацію великої кількості унікальних варіантів завдань при збереженні їх однакової дидактичної складності та структурної коректності.

6. У межах інформаційної технології реалізовано метод автоматичного розгортання VLE, який дозволяє динамічно створювати ізольовані VLE виконання для кожного студента, що забезпечує відтворюваність умов виконання завдань, контроль за використанням інструментів та масштабованість системи при роботі з великою кількістю студентів.

7. Інформаційна технологія включає метод автоматичної перевірки виконання PPET, що базується на формалізації стану VLE та дозволяє здійснювати багаторівневий контроль в реальному часі. Перевірка здійснюється як для кінцевого результату, так і для окремих етапів його виконання, що дозволяє більш точно оцінювати сформованість практичних навичок студентів.

Мультимодальність запропонованої інформаційної технології полягає у поєднанні декількох підходів до автоматизації навчального процесу: формального моделювання PPET, використання DSL для їх специфікації, застосування GenAI для створення варіантів завдань, а також використання VLE для виконання та перевірки практичних робіт.

Сильні сторони запропонованої інформаційної технології:

- формалізація структури практичних завдань за допомогою моделі PPET;
- використання домен-орієнтованої мови LTDL для автоматизації процесів створення, виконання та перевірки завдань;
- інтеграція GenAI з формальними методами валідації;
- можливість автоматичного розгортання ізольованих VLE;

- автоматична перевірка РРЕТ у реальному середовищі виконання;
- масштабованість системи при роботі з великою кількістю студентів.

Запропонована інформаційна технологія - це сучасне рішення, яке вдало вписується в концепцію цифрових навчальних середовищ наступного покоління Next Generation Digital Learning Environment [173] – гнучких, інтегрованих екосистем для персоналізованого навчання, з акцентом на інтероперабельність, співпрацю, доступність та аналітику. Такий комплексний підхід забезпечує можливість автоматизованої генерації, виконання та перевірки практичних завдань у масштабованих DLE.

Також проведено серію експериментів, які засвідчили працездатність всіх механізмів та системи в цілому, був апробований механізм генерації РРЕТ і порівняння AI-моделей, механізми автогенерації унікального завдання до виконання, механізми розгортання VLE та перевірки виконання РРЕТ в автоматичному режимі. Структура та функції працюють як передбачалося.

Результати досліджень, приведених в розділі, опубліковані в роботах [150, 168, 171].

ВИСНОВКИ

У дисертаційній роботі вирішено актуальне наукове завдання щодо комплексного забезпечення процесів автоматизованої генерації персоналізованих практичних інженерних завдань, автоматичного розгортання віртуальних навчальних середовищ та автоматичної перевірки результатів їх виконання. Це спрямовано на підвищення рівня персоналізованого навчання та формування у студентів сталих практичних навичок.

У результаті проведеного дослідження **усі поставлені задачі вирішено**, що підтверджується отриманими **науковими та практичними результатами**:

1. Проведено аналіз існуючих методів персоналізації практичних завдань у навчанні студентів інженерних спеціальностей, а також сучасних підходів до автоматизованої генерації, параметризації та перевірки завдань у DLE. Виявлено основні обмеження існуючих підходів: відсутність єдиних критеріїв порівняння методів персоналізації, недостатню інтеграцію процесів генерації, виконання і перевірки завдань, а також відсутність єдиного підходу до формалізації структури практичних завдань та їх життєвого циклу, що ускладнює масштабовану автоматизацію процесів їх генерації та перевірки. Обґрунтовано необхідність створення функціональної та формальної моделей PPET, придатних для автоматизованої обробки. Запропоновано систему кількісних показників для оцінки та порівняння рівня персоналізації практичних завдань та середовищ, що забезпечують їх реалізацію.

2. Розроблено функціональну модель PPET. Сутність моделі полягає в тому, що вона описує процеси створення, розгортання, виконання та перевірки PPET і формалізує їх повний життєвий цикл як єдиний безперервний конвеєр. Відмінність розробленої моделі від існуючих, що визначає її новизну, полягає в тому, що вона визначає повну послідовність етапів його життєвого циклу від створення до оцінювання результатів з урахуванням контексту та необхідних ресурсів, що формує уніфікований підхід до організації практичної підготовки з інженерних

спеціальностей. Впровадження даної моделі в інформаційну технологію дозволяє залучати здобувача як елемента механізму на етапі практичної взаємодії, а усі рутинні процеси повністю делегувати програмним агентам.

3. Розроблено формальну модель PRET. Сутність моделі полягає в тому, що вона формалізує структуру завдання, основні компоненти та взаємозв'язки між ними, що дозволяє розглядати персоналізоване практичне інженерне завдання як цілісний об'єкт, придатний до автоматизованої обробки. Розроблена формальна модель відрізняється від існуючих представлень освітніх завдань, які базуються переважно на неформальних описах або тестуванні вхідних-вихідних даних, тим що забезпечує формалізоване представлення внутрішньої структури завдання і дозволяє забезпечити однозначність інтерпретації завдання інформаційною системою; створити основу для параметризації та унікалізації варіантів; реалізувати автоматичну перевірку коректності структури завдання та забезпечити інтеграцію із програмними компонентами системи персоналізованого навчання.

4. Вперше розроблено домен-специфічну мову опису практичних завдань LTDL, граматику якої, на відміну від існуючих, охоплює повний життєвий цикл практичного завдання, об'єднуючи його педагогічні та технічні складові, що забезпечує підтримку процесу персоналізованого навчання в автоматичному режимі. Розроблена LTDL відрізняється від існуючих засобів формалізації практичних завдань тим, що функціонує в домені інженерної освіти, призначена для формалізації практичних інженерних завдань як параметризованих і виконуваних сутностей та адаптована до сучасних методів автоматизації на базі AI. Розроблено формальну граматику мови, транслятор та візуальний редактор, а також проведено її тестування.

Застосування даної мови дозволяє об'єднати процеси генерації завдання, автоматичного розгортання навчального середовища та автоматичної перевірки в єдиному формальному визначенні і запропонувати *комплексне* інформаційне забезпечення вказаних процесів, яке наразі відсутнє в існуючих рішеннях. Розроблена мова LTDL виступає формальним базисом для реалізації методів

автоматизованої генерації, розгортання та перевірки PPET.

5. Запропоновано архітектуру інтелектуального асистента, яка на відміну від існуючих, базується на мультиагентній системі, що реалізує BDI-парадигму в інтерпретації персоналізованого навчання і дозволяє формалізувати процес прийняття рішень AI-асистентом, зокрема під час генерації PPET, ітераційної адаптації завдань відповідно до рівня підготовки студента та формування рекомендацій. У розробленій архітектурі запропоновано дворівневу доменно-орієнтовану інтерпретацію BDI для задач персоналізованого навчання, яка відрізняється від класичних підходів структурною декомпозицією когнітивних станів та інтеграцією з процесом генерації практичних завдань. Таким чином інтелектуальний асистент реалізує персоналізовану стратегію навчання в межах DLE та замкнений цикл адаптації, у якому результати виконання завдань у середовищі VLE використовуються для оновлення переконань агента через функцію перегляду переконань, що забезпечує контекстно-залежну персоналізацію та безперервне уточнення навчальної траєкторії студента. Впровадження архітектури дозволяє (а) забезпечити контекстно-залежну генерацію завдань; (б) враховувати історію навчання; (в) формалізувати процес адаптації; (г) інтегрувати генерацію, виконання та оцінювання в рамках єдиного життєвого циклу завдання.

6. Удосконалено методи автоматизації персоналізованих практичних завдань, зокрема: метод автоматизованої генерації PPET; метод параметризації завдань; метод автоматичного розгортання VLE; метод автоматичної перевірки результатів виконання.

Метод генерації PPET удосконалено за рахунок поєднання генеративних можливостей LLM та формальної граматики LTDL, що забезпечує зниження кількості структурних та логічних помилок. Середнє значення інтегрального показника якості генерації в підході “LTDL+LLM” становить 0,90 (що вище, ніж у підходах “LLM+LLM” та “одиночна LLM” на 0,12 та 0,51 відповідно); а відсоток успішного розгортання відповідних середовищ без ручного втручання викладача при такій генерації підвищується до 96%.

Дослідження генерації PPET із різними AI-моделями підтвердили можливість використання GenAI для автоматичного створення PPET у форматі LTDL, що є важливим для реалізації запропонованої IT. Проведене порівняння показало, що LLM новіших поколінь демонструють вищу стабільність у досягненні синтаксично коректних результатів та потребують меншої кількості ітерацій повторної генерації, що може супроводжуватися збільшенням споживання обчислювальних ресурсів. Встановлено, що використання повністю автоматизованого режиму генерації з залученням AI-агентів і зовнішніх інструментів валідації дозволяє реалізувати замкнений цикл створення та перевірки завдань без участі людини.

Метод параметризації PPET удосконалено за рахунок поєднання стохастичної генерації параметрів із формальними обмеженнями мови LTDL, що дозволяє, на відміну від існуючих, формувати великий простір унікальних варіантів завдань при збереженні їх структурної коректності. Впровадження методу забезпечує контрольовану стохастичність параметризації: створюється велика кількість унікальних варіантів завдань, які відповідають навчальним цілям. Експериментально в навчальному процесі створено 252 унікальні варіації інженерного завдання для однієї навчальної теми без зниження рівня складності та структурної цілісності.

Метод автоматичного розгортання VLE удосконалено на основі LTDL-опису завдання, який, на відміну від існуючих підходів до організації VLE, забезпечує автоматичну трансляцію формального опису завдання у конфігурацію середовища виконання. Впровадження методу автоматичного розгортання дозволяє: (а) забезпечити уніфікацію конфігурації VLE для всіх студентів, усунути локальні проблеми програмного забезпечення і, таким чином, мінімізувати вплив технічних відмінностей на результати виконання практичних завдань; (б) забезпечити відтворюваність результатів виконання PPET у межах заданих параметрів генерації завдання та конфігурацій середовища і контрольованого процесу його ініціалізації; (в) знизити ризики реверс-аналізу з боку студентів з метою отримання відповідей без виконання завдання; (г) скоротити час підготовки до виконання практичних завдань за рахунок автоматизації процесу налаштування середовища. Аналіз часових

характеристик процесу розгортання екземплярів VLE показав, що процеси розгортання середовища залежать від інфраструктурних чинників, а не від логічних компонентів методу. Домінуючий внесок у загальний час виконання припадає на етап розгортання, тоді як етапи парсингу, трансляції та валідації, пов'язані із запровадженням мови LTDL, формують незначне навантаження (в середньому 27,5 с із 337,5 с загального часу розгортання, що приблизно становить 8% часу). Впровадження запропонованого методу в освітній процес також супроводжується високим рівнем позитивного сприйняття з боку студентів: 89% респондентів відзначили зручність автоматичного розгортання VLE, що зменшує потребу в його самостійному налаштуванні. Автоматичне розгортання середовища створює основу для подальшої формалізованої перевірки результатів, оскільки забезпечує відповідність початкового стану середовища заданій моделі завдання.

Метод автоматичної перевірки результатів виконання PPET удосконалено шляхом формалізації стану VLE у вигляді впорядкованого вектора артефактів, що дозволяє визначати відповідність поточного стану VLE цільовому стану виконання етапів завдання. Запропонований метод реалізує повністю автоматизований режим оцінювання та забезпечує його об'єктивність за рахунок виключення суб'єктивного впливу викладача. На відміну від традиційних підходів, метод дозволяє оцінювати не лише кінцевий результат, а й коректність виконання окремих етапів завдання шляхом аналізу проміжних станів середовища, і знижує ризики несанкціонованого отримання результатів без фактичного виконання завдання за рахунок перевірки стану середовища, а не лише кінцевого артефакту.

7. На основі комплексного застосування запропонованих моделей, методів та розроблених програмних засобів їх реалізації (парсер та транслятор мови, програмні модулі генерації, розгортання та перевірки, PAIA-moodle-plugin, LTDL-moodle-plugin, мультиагентна система оркестрації AI-агентів) створено нову інформаційну технологію персоналізації навчання студентів інженерних спеціальностей, яка на відміну від існуючих рішень, забезпечує масштабовану автоматизовану генерацію PPET, автоматичне розгортання VLE та автоматичну

перевірку результатів виконання цих завдань. Експериментальне дослідження із залученням реальних студентів підтверджує, що впровадження розробленої інформаційної технології в навчальний процес дозволяє підвищити рівень персоналізації (індекс $PI = 0,81$) порівняно з існуючими аналогами, також дозволяє усунути дефіцит часу викладача, який при ручній обробці робіт у 6 разів перевищує ліміт навантаження. Аналіз поведінкових показників засвідчив суттєве зростання інтенсивності взаємодії користувачів із системою. Зокрема, середній час активної роботи збільшився з 590–620 хвилин до 780–810 хвилин, а частка взаємодії з рекомендованим контентом зросла з 22–24% до 64–68%. Отримані результати свідчать про підвищення ефективності формування практичних навичок, що підтверджується покращенням показників виконання персоналізованих завдань (0,81 замість 0,68) і індексом сталості навичок на рівні 0,78. Запропонована інформаційна технологія також сприяє підвищенню рівня академічної доброчесності за рахунок унікалізації завдань та контролю середовища виконання. Передбачається, що використання розробленої інформаційної технології сприятиме формуванню більш сталих і міцних практичних навичок.

8. Практичну значущість результатів дослідження підтверджено їх впровадженням в рамках реалізації міжнародного наукового проекту “Цифрова трансформація освітнього процесу ЗВО в Україні та Молдові для сталого співробітництва з підприємствами” в рамках програми ERASMUS + “Розвиток потенціалу вищої освіти”; у навчальному процесі НУ «Чернігівська політехніка» при проведенні лекцій та лабораторних робіт відповідно до плану науково–дослідної роботи НУ «Чернігівська політехніка» (НДР “Цифрове навчальне середовище із віддаленим доступом”, державний реєстраційний номер 0125U000505); в Навчальному центрі PortaOne при викладанні публічних курсів з адміністрування ОС Linux та комп'ютерних мереж; в освітній діяльності компанії SendPulse Inc.

Отримані результати можуть бути використані для подальшого розвитку DLE та створення інтелектуальних систем підтримки персоналізованого навчання у підготовці фахівців інженерних спеціальностей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Maghsudi, S., Lan, A., Xu, J., & Schaar, M. van der. (2021). Personalized education in the artificial intelligence era: What to expect next. *IEEE Signal Processing Magazine*, 38(3), 37–50. 10.1109/MSP.2021.3055032.
2. Shemshack, A., & Spector, J. M. (2020). A systematic literature review of personalized learning terms. *Smart Learning Environments*, 7, 33. <https://doi.org/10.1186/s40561-020-00140-9>
3. Clément, B., Sauzéon, H., Roy, D., & Oudeyer, P.-Y. (2025). Improved performances and motivation in intelligent tutoring systems: Combining machine learning and learner choice [Preprint]. *arXiv*. <https://arxiv.org/abs/2402.01669>
4. Міністерство освіти і науки України. (2020). *Рекомендації щодо впровадження змішаного навчання у закладах фахової передвищої та вищої освіти*. <https://mon.gov.ua/static-objects/mon/sites/1/vishcha-osvita/2020/zmyshene%20navchanny/zmishanenavchannia-bookletsreads-2.pdf>
5. Peng, H., Ma, S., & Spector, J. M. (2019). Personalized adaptive learning: An emerging pedagogical approach enabled by a smart learning environment. In *Foundations and trends in smart learning: Proceedings of 2019 International Conference on Smart Learning Environments (Lecture Notes in Educational Technology)*. Springer. https://doi.org/10.1007/978-981-13-6908-7_24
6. du Plooy, E., Casteleijn, D., & Franzsen, D. (2024). Personalized adaptive learning in higher education: A scoping review of key characteristics and impact on academic performance and engagement. *Heliyon*, 10, e39630. <https://doi.org/10.1016/j.heliyon.2024.e39630>
7. Song, C., Shin, S.-Y., & Shin, K.-S. (2024). Implementing the dynamic feedback-driven learning optimization framework: A machine learning approach to personalize educational pathways. *Applied Sciences*, 14(2), 916. <https://doi.org/10.3390/app14020916>

8. Manoharan, S. (2017). Personalized assessment as a means to mitigate plagiarism. *IEEE Transactions on Education*, 60(2), 112–119. <https://doi.org/10.1109/TE.2016.2604210>.
9. Wokwi. (без дати). Wokwi official website. <https://wokwi.com/>
10. Heldal, R., Nguyen, T., Moreira, A., Lago, P., Duboc, L., Betz, S., Coroama, V., Penzenstadler, B., Porras, J., Capilla, R., Brooks, I., Oyedeki, S., & Venters, C. (2023). Sustainability competencies and skills in software engineering: An industry perspective [Preprint]. *SSRN*. <https://doi.org/10.2139/ssrn.4493646>.
11. Sanchez, P., Álvarez Torres, B., & Iborra, A. (2014). Improving transferable skills in engineering education through a pre-incubation semester. *International Journal of Engineering Education*, 30(4), 862–875..
12. Mangalore, P., Gamit, J. S., & Kanchan, M. (2025). Developing a framework for sustainable engineering education through transformative learning principles. *Discover Sustainability*, 6, 1473. <https://doi.org/10.1007/s43621-025-02146-0>
13. Наказ МОН України №371 “Про затвердження критеріїв оцінювання навчальних досягнень учнів у системі загальної середньої освіти”. Інформаційний збірник Міністерства освіти і науки України від 05.2008 — 2008 р., № 13, стор. 20.
14. Crawley, E. F., Malmqvist, J., Östlund, S., Brodeur, D. R., & Edström, K. (2014). The CDIO approach. In *Rethinking engineering education* (pp. 11–45). Springer. https://doi.org/10.1007/978-3-319-05561-9_2
15. Association for Computing Machinery, & IEEE Computer Society. (2020). *Computing curricula 2020: Paradigms for global computing education*. ACM. <https://doi.org/10.1145/3467967>
16. ABET. (без дати). *Criteria for accrediting engineering programs*. <https://www.abet.org/accreditation/accreditation-criteria/>
17. Holdgraf, C., Culich, A., Rokem, A., Deniz, F., Alegro, M., & Ushizima, D. (2017). Portable learning environments for hands-on computational instruction: Using container-and cloud-based technology to teach data science. In *Practice and Experience in*

Advanced Research Computing 2017: Sustainability, Success and Impact (pp. 1-9).
<https://doi.org/10.48550/arXiv.1703.04900>.

18. Peters, A. K., Capilla, R., Coroamă, V. C., Høldal, R., Lago, P., Leifler, O., ... & Venters, C. C. (2024). Sustainability in computing education: A systematic literature review. *ACM transactions on computing education*, 24(1), 1-53.
<https://doi.org/10.48550/arXiv.2305.10369>.

19. Luxton-Reilly, A., Alblawi, I., Becker, B., Giannakos, M., Kumar, A., Ott, L. M., Paterson, J., Scott, M., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. In *ITiCSE 2018 companion: Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education* (pp. 55–106). Association for Computing Machinery.
<https://doi.org/10.1145/3293881.3295779>.

20. Закон України «Про освіту» від 05.09.2017 № 2145-VIII, опублікований у Відомостях Верховної Ради (ВВР), 2017, № 38-39, ст. 380

21. Міністерство цифрової трансформації України. (2020). Звіт щодо результатів експрес аналізу поточного стану ІТ освіти.
https://thedigital.gov.ua/storage/uploads/files/page/community/docs/Звіт_щодо_результатів_експрес_аналізу_поточного_стану_ІТ_освіти.pdf.

22. Nahorniuk, O., & Levytska, L. (2024). Integrating professional and educational standards for developing IT curriculum. *Universum*, (4), 160–168.
<https://archive.liga.science/index.php/universum/article/view/699>

23. Malik, A. A., Hassan, M., Rizwan, M., Mushtaque, I., Lak, T. A., & Hussain, M. (2023). Impact of academic cheating and perceived online learning effectiveness on academic performance during the COVID-19 pandemic among Pakistani students. *Frontiers in Psychology*, 14, 1124095. <https://doi.org/10.3389/fpsyg.2023.1124095>

24. Valizadeh, M. (2022). Cheating in Online Learning Programs: Learners' Perceptions and Solutions. *Turkish Online Journal of Distance Education*, 23(1), 195–209.
<https://doi.org/10.17718/tojde.1050394>.

25. Naidu, K., & Sevnarayan, K. (2023). ChatGPT: An ever-increasing encroachment of artificial intelligence in online assessment in distance education. *Online Journal of Communication and Media Technologies*, 13(1), e202336. <https://doi.org/10.30935/ojcmt/13291>
26. Stevens, R., Silver, L., Richards, R., & Campbell, K. (2022). A comparison of faculty and student perspectives of academic integrity in an online environment: A pilot study. *Journal of Business Administration Online*, 16(2), 1–13. <https://www.atu.edu/business/jbao/fall2022/202202%2002%20Stevens%20Silver%20Richards%20Campbell.pdf>
27. Souza, D., Felizardo, K., & Barbosa, E. (2016). A systematic literature review of assessment tools for programming assignments. In *2016 IEEE 29th international conference on software engineering education and training (CSEET)* (pp. 147–156). IEEE. <https://doi.org/10.1109/CSEET.2016.48>.
28. Horváth, M., Kormaník, T., & Porubán, J. (2024). Adaptation of automated assessment system for large programming courses. In *5th international computer programming education conference (ICPEC 2024) (Open Access Series in Informatics (OASIs))*, Vol. 122, pp. 4:1–4:11). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/OASIs.ICPEC.2024.4>.
29. Walkington, C., & Bernacki, M. L. (2020). Appraising research on personalized learning: Definitions, theoretical alignment, advancements, and future directions. *Journal of Research on Technology in Education*, 52(3), 235–252. <https://doi.org/10.1080/15391523.2020.1747757>
30. Bloom, B. S. (1968). *Learning for mastery: Instruction and curriculum*. Regional Educational Laboratory for the Carolinas and Virginia. [http://www.researchforteachers.org.uk/sites/default/files/Docs/Bloom%20\(1968\)%20Learning%20for%20Mastery_0.pdf](http://www.researchforteachers.org.uk/sites/default/files/Docs/Bloom%20(1968)%20Learning%20for%20Mastery_0.pdf).
31. Vasilev, I. (2024). Bloom's mastery learning model: A case study example of application in the TVET. *Knowledge International Journal*, 66(2), 211–216. <https://www.researchgate.net/publication/384761613>.

32. Deci, E. L., & Ryan, R. M. (2008). Self-determination theory: A macrotheory of human motivation, development, and health. *Canadian Psychology / Psychologie canadienne*, 49(3), 182–185. <https://doi.org/10.1037/a0012801>.
33. Agustianto, K., Permanasari, A. E., Kusumawardani, S. S., & Hidayah, I. (2016). Design adaptive learning system using metacognitive strategy path for learning in classroom and intelligent tutoring systems. In *AIP Conference Proceedings*, 1755(1), 070012. <https://doi.org/10.1063/1.4958507>
34. Gibbons, A. S. (2008). Model-centered instruction, the design and the designer. In D. Ifenthaler, P. Pirnay-Dummer, & J. M. Spector (Eds.), *Understanding models for learning and instruction* (pp. 161–173). Springer. https://doi.org/10.1007/978-0-387-76898-4_8.
35. Sannino, A., & Engeström, Y. (2018). Cultural-historical activity theory: Founding insights and new challenges. *Cultural-Historical Psychology*, 14, 43–56. <https://www.researchgate.net/publication/328398841>.
36. Chen, K. Z., Tseng, J. Y., & Oakley, B. (2025). Transforming online teacher training through expansive learning: A case study applying cultural-historical activity theory and the change laboratory method. *The Asia-Pacific Education Researcher*, 34, 1413–1430. <https://doi.org/10.1007/s40299-024-00954-y>.
37. Guadagnoli, M. A., & Lee, T. D. (2004). Challenge point: A framework for conceptualizing the effects of various practice conditions in motor learning. *Journal of Motor Behavior*, 36(2), 212–224. <https://doi.org/10.3200/JMBR.36.2.212-224>.
38. Khyzhniak, A., & Kazymyr, V.. (2025). Uzahalнена klasyfikatsiia rivniv personalizatsii praktychnykh zavdan v IT-osviti [Generalized classification of personalization levels in practical assignments in IT education]. *Nauka i tekhnika sohodni*, 7(48), 1932–1949. [https://doi.org/10.52058/2786-6025-2025-7\(48\)-1932-1949](https://doi.org/10.52058/2786-6025-2025-7(48)-1932-1949)
39. Moodle LMS. (без дати). <https://moodle.org/>
40. Canvas LMS. (без дати). <https://www.instructure.com/canvas>
41. JupyterHub. (без дати). <https://jupyter.org/hub>
42. OpenOlat LMS. (без дати). <https://www.openolat.com/>

43. Cisco Net Academy. (без дати). <https://www.netacad.com/>
44. Agudo, I., Rios, R., & Nieto, A. (2019). Personalized computer security tasks with automatic evaluation and feedback. In *Proceedings of the 2019 AIS SIGED international conference on information systems education and research*. <https://aisel.aisnet.org/siged2019/1>.
45. Vykopal, J., Švábenský, V., Šeda, P., & Čeleda, P. (2022). Preventing cheating in hands-on lab assignments. In *Proceedings of the 53rd ACM technical symposium on computer science education* (Vol. 1, pp. 78–84). Association for Computing Machinery. <https://doi.org/10.1145/3478431.3499420>
46. Chan, J., & Teng, S. (2024). The Design and Implementation of PAGE: Personalised Assessment Generative Engine. *Pacific Journal of Technology Enhanced Learning*, 6(2), 33-46. <https://doi.org/10.24135/pjtel.v6i2.203>.
47. Zampirolli, F., Pisani, P., Josko, J., Kobayashi, G., Fraga, F., Goya, D., & Savegnago, H. (2020). Parameterized and automated assessment on an introductory programming course. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, (pp. 1573-1582). Porto Alegre: SBC. doi:10.5753/cbie.sbie.2020.1573.
48. Srinivasan, B., Nehru, M., Parthasarathi, R., Mukherjee, S., & Thankachan, J. A. (2024). Automated computer program evaluation and projects - Our experiences [Preprint]. *arXiv*. <https://arxiv.org/abs/2404.04521>.
49. Segal, A., Gal, K., Shani, G., & Shapira, B. (2019). A difficulty ranking approach to personalization in e-learning. *International Journal of Human-Computer Studies*, 130, 261–272. <https://doi.org/10.1016/j.ijhcs.2019.07.002>.
50. Khan Academy. (без дати). <https://www.khanacademy.org/>
51. Codio. (без дати). <https://www.codio.com/>
52. Labster. (без дати). <https://www.labster.com/>
53. Jacobs, S., Peters, H., Jaschke, S., & Kiesler, N. (2025). Unlimited practice opportunities: Automated generation of comprehensive, personalized programming tasks. In *ITiCSE 2025: Proceedings of the 30th ACM conference on innovation and technology*

in computer science education (Vol. 1, pp. 319–325). Association for Computing Machinery. <https://doi.org/10.1145/3724363.3729089>.

54. Tang, X., Chen, Y., Li, X., Liu, J., & Ying, Z. (2019). A reinforcement learning approach to personalized learning recommendation systems. *British Journal of Mathematical and Statistical Psychology*, 72(1), 108–135. <https://doi.org/10.1111/bmsp.12144>.

55. Das Adhikary, P., & Metsämuuronen, J. (2025). Knowledge tracing models in digital learning: Historical evolution, categorization, and empirical evaluation [Preprint]. *ResearchGate*. <https://www.researchgate.net/publication/392398351>.

56. CodeCombat. (без даты). <https://codecombat.com/>

57. CodeGrade. (без даты). <https://www.codegrade.com/>

58. SAP Learning Hub. (без даты). <https://learninghub.sap.com/>

59. Song, X., Chen, K., Bi, Z., Niu, Q., Song, J., Liu, J., Peng, B., Zhang, S., Liu, M., Li, M., Yuan, Z., Zhang, L., Zhong, Y., Pan, X., Xu, J., Zhang, Y., Wang, J., & Feng, P. (2024). Mastering reinforcement learning: Foundations, algorithms, and real-world applications [Preprint]. *SSRN*. <https://doi.org/10.2139/ssrn.5208695>.

60. Ersozlu, Z., Taheri, S., & Koch, I. (2024). A review of machine learning methods used for educational data. *Education and Information Technologies*, 29, 22125–22145. <https://doi.org/10.1007/s10639-024-12704-0>.

61. Zhang, J., & Fan, Y. (2025). Machine learning based big data analytics for education in curriculum reforms. *Applied Mathematics and Nonlinear Sciences*, 10(1), 1–13. <https://doi.org/10.2478/amns-2025-0135>.

62. Lou, H., Yue, P., & Chen, J. (2025). Research on adaptive education path dynamic programming algorithm based on reinforcement learning and cognitive graphs. *IEEE Access*, 13, 105446–105462. <https://doi.org/10.1109/ACCESS.2025.3578389>.

63. Bucchiarone, A., Martorella, T., & Colombo, D. (2022). Polyglot: A personalized and gamified e-tutoring system [Preprint]. *arXiv*. <https://arxiv.org/abs/2210.15256>.

64. CodinGame. (без даты). <https://www.codingame.com/start/>

65. Bassner, P., Frankford, E., & Krusche, S. (2024). Iris: An AI-driven virtual tutor for computer science education. In *ITiCSE 2024: Proceedings of the 2024 ACM conference on innovation and technology in computer science education* (Vol. 1, pp. 394–400). Association for Computing Machinery. <https://doi.org/10.1145/3649217.3653543>.
66. Kira Learning. (без дати). <https://www.kira-learning.com/>
67. Vykopal, J., Šeda, P., Švábenský, V., & Čeleda, P. (2023). Smart environment for adaptive learning of cybersecurity skills. *IEEE Transactions on Learning Technologies*, 16(3), 443–456. <https://doi.org/10.1109/TLT.2022.3216345>.
68. Logacheva, E., Hellas, A., Prather, J., Sarsa, S., & Leinonen, J. (2024). Evaluating contextually personalized programming exercises created with generative AI. In *Proceedings of the 2024 ACM conference on international computing education research (ICER '24)* (pp. 95–113). Association for Computing Machinery. <https://doi.org/10.1145/3632620.3671103>
69. Govea, J., Navarro, A., Sánchez-Viteri, S., & Villegas, W. (2024). Implementation of deep reinforcement learning models for emotion detection and personalization of learning in hybrid educational environments. *Frontiers in Artificial Intelligence*, 7, 1458230. <https://doi.org/10.3389/frai.2024.1458230>
70. Khyzhniak, A. V., & Kazymyr, V. V. (2025). Analysis of methods for supporting personalization in IT education. *Herald of Advanced Information Technology*, 8(3), 366–381. <https://doi.org/10.15276/hait.08.2025.24>
71. Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. (2022). Automatic generation of programming exercises and code explanations using large language models. In *ICER '22: Proceedings of the 2022 ACM conference on international computing education research* (Vol. 1, pp. 27–43). Association for Computing Machinery. <https://doi.org/10.1145/3501385.35439>.
72. Jacobs, S., Haas, J., & Kiesler, N. (2025, November). Student Engagement with GenAI's Tutoring Feedback: A Mixed Methods Study. In *Proceedings of the 25th*

Koli Calling International Conference on Computing Education Research (pp. 1-12).
<https://doi.org/10.1145/3769994.3770034>

73. Jordan, M. (2024). Need a programming exercise generated in your native language? ChatGPT's got your back: Automatic generation of non-English programming exercises using OpenAI GPT-3.5. In *SIGCSE 2024: Proceedings of the 55th ACM technical symposium on computer science education* (Vol. 1, pp. 618–624). Association for Computing Machinery. <https://doi.org/10.1145/3626252.3630897>

74. Holmes, W., Bialik, M., & Fadel, C. (2019). *Artificial intelligence in education: Promise and implications for teaching and learning*. Center for Curriculum Redesign.

75. Garzón, J., Patiño, E., & Marulanda, C. (2025). Systematic Review of Artificial Intelligence in Education: Trends, Benefits, and Challenges. *Multimodal Technologies and Interaction*, 9(8), 84. <https://doi.org/10.3390/mti9080084>

76. Alkafaween, U., Albluwi, I., & Denny, P. (2025). Automating autograding: Large language models as test suite generators for introductory programming. *Journal of Computer Assisted Learning*, 41(1), e13100. <https://doi.org/10.1111/jcal.13100>

77. Qu, F., Jia, X., & Wu, Y. (2021). Asking questions like educational experts: Automatically generating question–answer pairs on real-world examination data. In *Proceedings of the 2021 conference on empirical methods in natural language processing (EMNLP 2021)* (pp. 2583–2593). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.emnlp-main.202>.

78. Rodriguez-Torrealba, R., Garcia-Lopez, E., & Garcia-Cabot, A. (2022). End-to-end generation of multiple-choice questions using text-to-text transfer transformer models. *Expert Systems with Applications*, 208, 118258. <https://doi.org/10.1016/j.eswa.2022.118258>.

79. Del Carpio Gutierrez, A., Denny, P., & Luxton-Reilly, A. (2024). Evaluating automatically generated contextualised programming exercises. In *SIGCSE 2024: Proceedings of the 55th ACM technical symposium on computer science education* (Vol. 1, pp. 289–295). Association for Computing Machinery.

<https://doi.org/10.1145/3626252.3630863>

80. Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024). Prompt problems: A new programming exercise for the generative AI era. In *SIGCSE 2024: Proceedings of the 55th ACM technical symposium on computer science education* (Vol. 1, pp. 296–302). Association for Computing Machinery. <https://doi.org/10.1145/3626252.3630909>

81. Roe, J., & Perkins, M. (2024). Generative AI in self-directed learning: A scoping review [Preprint]. *arXiv*. <https://arxiv.org/abs/2411.07677>

82. Azahar, M. Z. A., & Ghauth, K. I. (2022). A hybrid automated essay scoring using NLP and random forest regression. In *Proceedings of the international conference on computer, information technology and intelligent computing (CITIC 2022)* (pp. 448–457). Atlantis Press. https://doi.org/10.2991/978-94-6463-094-7_35.

83. Combéfis, S. (2022). Automated code assessment for education: Review, classification and perspectives on techniques and tools. *Software, 1*, 3–30. <https://doi.org/10.3390/software1010002>

84. Xu, L., Huang, D., & Tsai, W.-T. (2014). Cloud-based virtual laboratory for network security education. *IEEE Transactions on Education, 57*(3), 145–150. <https://doi.org/10.1109/TE.2013.2282285>

85. Cardoso, M., Barroso, R., Castro, A. V., & Rocha, A. (2017). Virtual programming labs in the computer programming learning process: Preparing a case study. In *Proceedings of EDULEARN17 conference* (pp. 7146–7155). IATED..

86. Chea, M. S., Han, S., & Lee, H. (2019). Docker-based cloud system for computer programming labs. In *Proceedings of the 14th International Conference on Computer Science and Education (ICCSE 2019)* (pp. 622–626). <https://doi.org/10.1109/ICCSE.2019.8845470>

87. Chen, G. (2020, April). CvLabs: A container based interactive virtual lab for IT education (Technical report). Georgia Institute of Technology. <https://smartech.gatech.edu/handle/1853/64898>

88. Vykopal, J., Ošlejšek, R., Čeleda, P., Vizváry, M., & Tovarňák, D. (2017).

KYPO cyber range: Design and use cases. In *Proceedings of the 12th international conference on software technologies (ICSOFT 2017)* (pp. 310–321). SciTePress. <https://doi.org/10.5220/0006428203100321>.

89. Potkonjak, V., Gardner, M., Callaghan, V., Mattila, P., Guetl, C., Petrović, V. M., & Jovanović, K. (2016). Virtual laboratories for education in science, technology, and engineering: A review. *Computers & Education*, 95, 309–327. <https://doi.org/10.1016/j.compedu.2016.02.002>.

90. Li, J., & Liang, W. (2024). Effectiveness of virtual laboratory in engineering education: A meta-analysis. *PLoS ONE*, 19(12), e0316269. <https://doi.org/10.1371/journal.pone.0316269>

91. Loksa, D., Margulieux, L., Becker, B. A., Craig, M., Denny, P., Pettit, R., & Prather, J. (2022). Metacognition and self-regulation in programming education: Theories and exemplars of use. *ACM Transactions on Computing Education*, 22(4), Article 39, pp.1-31. <https://doi.org/10.1145/3487050>

92. Somani, G., Gaur, M. S., Sanghi, D., Conti, M., Rajarajan, M., & Buyya, R. (2017). Combating DDoS attacks in the cloud: Requirements, trends, and future directions. *IEEE Cloud Computing*, 4(1), 22–32. <https://doi.org/10.1109/MCC.2017.14>.

93. Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher Education*, 32, 347–364. <https://doi.org/10.1007/BF00138871>

94. Malmqvist, J., Edström, K., & Rosén, A. (2020). CDIO standards 3.0: Updates to the core CDIO standards. In *Proceedings of the 16th international CDIO conference*. pp. 60–76. https://www.cdio.org/sites/default/files/documents/CDIO_Proceedings_2020_Malmqvist_135.pdf

95. Pyakurel, P., & Soupeez, J.-B. R. G. (2025). Embedding authenticity in assessments for engineering education. In *Proceedings of the 14th international conference on new perspectives in science education*. *Pixel*. <https://conference.pixel-online.net/files/npse/ed0014/FP/9726-ENGE7111-FP-NPSE14.pdf>

96. Dela Cruz, J. P., Lejano, M. V., Martin, J., Marquez, S. J. R., Fernandez, A. R. S., & Bautista, R. G. (2025). Virtual laboratories in enhancing experimental skills and scientific understanding among high school learners. *American Journal of Educational Research*, 13(6), 338–343. <https://pubs.sciepub.com/education/13/6/6>
97. Skulmowski, A., & Xu, K. (2022). Understanding cognitive load in digital and online learning: A new perspective on extraneous cognitive load. *Educational Psychology Review*, 34, 171–196. <https://doi.org/10.1007/s10648-021-09624-7>.
98. IMS Global Learning Consortium. (без дати). Question & test interoperability (QTI) specification. <https://www.imsglobal.org/spec/qti/v3p0/oview/>
99. Paiva, J., Leal, J., & Figueira, Á. (2022). Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education*, 22(3), Article 34, 1–40. <https://doi.org/10.1145/3513140>.
100. Humble, N. (2023). A conceptual model of what programming affords secondary school courses in mathematics and technology. *Education and Information Technologies*, 28, 10183–10208. <https://doi.org/10.1007/s10639-023-11577-z>
101. Kattis. (без дати). <https://open.kattis.com>
102. DOMjudge. (без дати). <https://www.domjudge.org/>
103. CodeRunner. (без дати). <https://coderunner.org.nz/>
104. Helic, D. (2007). Formal representations of learning scenarios: A methodology to configure e-learning systems. *Journal of Universal Computer Science*, 13(4), 504–530. <https://doi.org/10.3217/jucs-013-04-0504>
105. Woodcock, J., Larsen, P. G., Bicarregui, J., & Fitzgerald, J. (2009). Formal methods: Practice and experience. *ACM Computing Surveys*, 41(4), Article 19, 1–36. <https://doi.org/10.1145/1592434.1592436>
106. Fernández-Fernández, C., & Simons, A. (2014). An implementation of the task algebra, a formal specification for the task model in the discovery method. *Journal of Applied Research and Technology*, 12(5), 908–918. [https://doi.org/10.1016/S1665-6423\(14\)70597-8](https://doi.org/10.1016/S1665-6423(14)70597-8)
107. World Wide Web Consortium. (без дати). Task meta models.

https://www.w3.org/2005/Incubator/model-based-ui/wiki/Task_Meta_Models.html

108. Stock, S., Dunkelau, J., & Mashkoor, A. (2024). Application of AI to formal methods - An analysis of current trends [Preprint]. *arXiv*. <https://arxiv.org/abs/2411.14870>.
109. Choi, Y., Lee, Y., Cho, J., Baek, J., Shin, D., Yu, H., & Heo, J. (2020). Assessment modeling: Fundamental pre-training tasks for interactive educational systems [Preprint]. *arXiv*. <https://arxiv.org/abs/2002.05505>.
110. Fartuşnic, R., Istrate, O., & Fartuşnic, C. (2025). Beyond automation: A conceptual framework for AI in educational assessment. *Journal of Digital Pedagogy*, 4(1), 83–102. <https://doi.org/10.61071/JDP.2555>
111. Jiang, Y. H., Lu, Y., Dai, L., Wang, J., Li, R., & Jiang, B. (2025). Agentic workflow for education: Concepts and applications [Preprint]. *arXiv*. <https://arxiv.org/abs/2509.01517>.
112. Ichida, A. Y., & Meneguzzi, F. (2023). Modeling a conversational agent using BDI framework. In *SAC '23: Proceedings of the 38th ACM/SIGAPP symposium on applied computing* (pp. 856–863). Association for Computing Machinery. <https://doi.org/10.1145/3555776.3577657>
113. Casali, A., Godo, L., & Sierra, C. (2006). Modeling travel assistant agents: A graded BDI approach. In M. Bramer (Ed.), *Artificial intelligence in theory and practice* (Vol. 217, pp. 415–424). Springer. https://doi.org/10.1007/978-0-387-34747-9_43.
114. Archibald, B., Sevegnani, M., & Xu, M. (2025). Modelling and verifying BDI agents under uncertainty. *Science of Computer Programming*, 242, 103254. <https://doi.org/10.1016/j.scico.2024.103254>.
115. Liang, P., Jordan, M. I., & Klein, D. (2010). Learning programs: A hierarchical Bayesian approach. In *Proceedings of the 27th international conference on machine learning (ICML 2010)* (pp. 639–646). Omnipress. <https://dl.acm.org/doi/10.5555/3104322.3104404>
116. Jiménez, D., Guerrero, A.-E., Prieto-Blazquez, J., & Conesa, J. (2014). A domain-specific language for virtual classrooms. *International Journal of Metadata, Semantics and Ontologies*, 9, 312–323. <https://doi.org/10.1504/IJMSO.2014.065444>.

117. Westphal, Oliver. (2020). A Framework for Generating Diverse Haskell-IO Exercise Tasks. *Pre-proceedings of the 28th International Workshop on Functional and Logic Programming (WFLP 2020)* <https://doi.org/10.48550/arXiv.2008.12751>.
118. Westphal, Oliver & Voigtländer, Janis. (2020). Describing Console I/O Behavior for Testing Student Submissions in Haskell. *Electronic Proceedings in Theoretical Computer Science*, 321, 19-36. <https://doi.org/10.4204/EPTCS.321.2>.
119. González García, C., Núñez Valdez, E., Moreno Ger, P., Gonzalez Crespo, R., Pelayo García-Bustelo, B., & Cueva Lovelle, J. (2019). Agile development of quiz-based multiplatform educational games using a domain-specific language. *Universal Access in the Information Society*, 18, 599–614. <https://doi.org/10.1007/s10209-019-00681-y>
120. Willert, N., & Thiemann, J. (2024). Template-based generator for single-choice questions. *Technology, Knowledge and Learning*, 29, 355–370. <https://doi.org/10.1007/s10758-023-09659-5>.
121. Kosar, T., Bohra, S., & Mernik, M. (2016). Domain-specific languages: A systematic mapping study. *Information and Software Technology*, 71, 77–91. <https://doi.org/10.1016/j.infsof.2015.11.001>.
122. Sebastián, G., Tesoriero, R., & Gallud, J. A. (2021). A domain specific language notation for a language learning activity generation tool. *Multimedia Tools and Applications*, 80, 36275–36304. <https://doi.org/10.1007/s11042-021-11296-y>
123. Laforcade, P., Nodenot, T., Choquet, C., & Caron, P.-A. (2007). Model-driven engineering (MDE) and model-driven architecture (MDA) applied to the modeling and deployment of technology enhanced learning (TEL) systems: Promises, challenges and issues. In *Architecture solutions for e-learning systems* (p. 21). IGI Global. <https://doi.org/10.4018/978-1-59904-633-4.ch007>
124. Zschaler, S., Barnett, W., Boronat, A., et al. (2026). The MDENet education platform: Zero-install directed activities for learning MDE. *Software and Systems Modeling*, 25, 287–313. <https://doi.org/10.1007/s10270-025-01292-3>
125. Jurado, F., Rodríguez, F., Chavarriaga, E., & Rojas, L. (2025). Blending language models and domain-specific languages in computer science education: A case

study on API RESTful. *International Journal of Interactive Multimedia and Artificial Intelligence*, 9, 86–104. <https://doi.org/10.9781/ijimai.2025.09.005>

126. Khyzhniak, A., & Kazymyr, V. (2025). Domenno-orientovana mova opysu personalizovanykh praktychnykh zavdan dlia inzhenernykh spetsialnostei [Domain-Specific Language for Describing Personalized Practical Tasks in Engineering Disciplines]. *Technical sciences and technologies*, (3 (41)), 261–271. [https://doi.org/10.25140/2411-5363-2025-3\(41\)-261-271](https://doi.org/10.25140/2411-5363-2025-3(41)-261-271)

127. Sukacke, V., de Carvalho Guerra, A. O. P., Ellinger, D., Carlos, V., Petroniene, S., Gaiziuniene, L., Blanch, S., Marba-Tallada, A., & Brose, A. (2022). Towards active evidence-based learning in engineering education: A systematic literature review of PBL, PjBL, and CBL. *Sustainability*, 14(21), Article 13955. <https://doi.org/10.3390/su142113955>

128. Bernacki, M. L., Greene, M. J., & Lobczowski, N. G. (2021). A systematic review of research on personalized learning: Personalized by whom, to what, how, and for what purpose(s)? *Educational Psychology Review*, 33(4), 1675–1715. <https://doi.org/10.1007/s10648-021-09615-8>

129. Banihashem, S. K., Noroozi, O., van Ginkel, S., Macfadyen, L. P., & Biemans, H. J. A. (2022). A systematic review of the role of learning analytics in enhancing feedback practices in higher education. *Educational Research Review*, 37, Article 100489. <https://doi.org/10.1016/j.edurev.2022.100489>

130. Deeva, G., Bogdanova, D., Serral, E., Snoeck, M., & De Weerd, J. (2021). A review of automated feedback systems for learners: Classification framework, challenges and opportunities. *Computers & Education*, 162, Article 104094. <https://doi.org/10.1016/j.compedu.2020.104094>

131. Van den Beemt, A., Groothuijsen, S., Ozkan, L., & Hendrix, W. (2023). Remote labs in higher engineering education: Engaging students with active learning pedagogy. *Journal of Computing in Higher Education*, 35, 320–340. <https://doi.org/10.1007/s12528-022-09331-4>

132. Reining, N., & Kauffeld, S. (2022). Empirical findings on learning success

and competence development at learning factories: A scoping review. *Education Sciences*, 12(11), Article 769. <https://doi.org/10.3390/educsci12110769>.

133. Chen, Y., Ma, L., Pirzada, P., & Chai, K. K. (2025). Evaluating a guided personalised learning model in undergraduate engineering education: A data-driven approach to student-centred pedagogy. *Education Sciences*, 15(7), Article 925. <https://doi.org/10.3390/educsci15070925>

134. Pardo, A., Bartimote, K., Buckingham Shum, S., Dawson, S., Gao, J., Gasevic, D., Leichtweis, S., Liu, D., Martinez-Maldonado, R., Mirriahi, N., Moskal, A. C. M., Schulte, J., Siemens, G., & Vigentini, L. (2018). OnTask: Delivering data-informed, personalized learning support actions. *Journal of Learning Analytics*, 5(3), 235–249. <https://doi.org/10.18608/jla.2018.53.15>

135. Lim, L.-A., Dawson, S., Gasevic, D., Joksimovic, S., Pardo, A., Fudge, A., & Gentili, S. (2021). Students' perceptions of, and emotional responses to, personalised learning analytics-based feedback: An exploratory study of four courses. *Assessment & Evaluation in Higher Education*, 46(3), 339–359. <https://doi.org/10.1080/02602938.2020.1782831>

136. Weidlich, J., Fink, A., Frey, A., Jivet, I., Gombert, S., Menzel, L., Giorgashvili, T., Yau, J., & Drachsler, H. (2025). Highly informative feedback using learning analytics: How feedback literacy moderates student perceptions of feedback. *International Journal of Educational Technology in Higher Education*, 22, Article 43. <https://doi.org/10.1186/s41239-025-00539-9>

137. National Institute of Standards and Technology. (1993). *Integration definition for function modeling (IDEF0)* (FIPS Publication 183). <https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub183.pdf>

138. Kimball, J. P. (1973). *The formal theory of grammar*. Prentice-Hall.

139. Becerra-Bonache, L., Bel-Enguix, G., Jiménez-López, M. D., & Martín-Vide, C. (2014). Mathematical foundations: Formal grammars and languages. In R. Mitkov (Ed.), *The Oxford handbook of computational linguistics* (2nd ed.). Oxford University Press. <https://doi.org/10.1093/oxfordhb/9780199573691.013.021>

140. Гавриленко, С. Ю. (2021). *Формальні мови, граматики та автомати*. Навчальний посібник. Харків: НТУ «ХПІ», <https://repository.kpi.kharkov.ua/server/api/core/bitstreams/5847efdd-6ff5-4f7f-9de8-6f6555ad4cc0/content>
141. Спекторський, І. Я., & Статкевич, В. М. (2019). *Формальні мови та автомати / Підручник для студ спец. 124 Системний аналіз*. КПІ ім. Ігоря Сікорського.
142. Wasowski, A., & Berger, T. (2023). *Domain-specific languages: Effective modeling, automation, and reuse*. Springer. <https://doi.org/10.1007/978-3-031-23669-3>
143. Ramos-Díaz, J. G., Navarro, I., Silva, J., & Arroyo, G. (2012). Defining DSL design principles for enhancing the requirements elicitation process. *Acta Universitaria*, 22, 126-133. <https://doi.org/10.15174/au.2012.352>.
144. Strembeck, M., & Zdun, U. (2009). An approach for the systematic development of domain-specific languages. *Software: Practice and Experience*, 39(15), 1253–1292. <https://doi.org/10.1002/spe.936>.
145. Mernik, M., Heering, J., & Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4), 316–344. <https://doi.org/10.1145/1118890.1118892>.
146. Стативка, Ю. І. (2023). *Формальні мови: Основні концепти і представлення*. КПІ ім. Ігоря Сікорського. <https://ela.kpi.ua/items/5f179fe8-05de-4b23-9563-13fd4d24e37e>.
147. International Organization for Standardization. (1996). *Information technology - Syntactic metalanguage - Extended BNF (ISO/IEC 14977:1996)*. <https://www.iso.org>
148. *Terraform Language Documentation*. (без дати). <https://developer.hashicorp.com/terraform/language>
149. Montenegro-Marín, C. E., Cueva Lovelle, J. M., Sanjuán, O., & García Díaz, V. (2012). Domain-specific language for the generation of learning management systems modules. *Journal of Web Engineering*, 11(1), 23–50.

<https://journals.riverpublishers.com/index.php/JWE/article/view/4225>

150. Khyzhniak, A. V., & Kazymyr, V. V. (2026). Modeli personalizatsii navchannia v tsyfrovomu osvithnomu seredovyshchi [Models of personalized learning in a digital educational environment]. *Technical sciences and technologies*, (1(43)), 279–290. [https://doi.org/10.25140/2411-5363-2026-1\(43\)-279-290](https://doi.org/10.25140/2411-5363-2026-1(43)-279-290)

151. Nowakowski, G., Telenyk, S., & Vovk, Y. (2023). Chatbots lifecycle support platform. In *Proceedings of the IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* (pp. 308–319). <https://doi.org/10.1109/IDAACS58523.2023.10348794>.

152. Rao, A. S., & Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, & E. Sandewall (Eds.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)* (pp. 473–484). Morgan Kaufmann. <http://jmvidal.cse.sc.edu/library/rao91a.pdf>.

153. Mikic Fonte, F., Burguillo, J., & Llamas Nistal, M. (2012). An intelligent tutoring module controlled by BDI agents for an e-learning platform. *Expert Systems with Applications*, 39, 7546–7554. <https://doi.org/10.1016/j.eswa.2012.01.161>

154. Hafnar, D., & Demšar, J. (2024). Zero-shot reasoning: Personalized content generation without the cold start problem. Preprint. *arXiv*. <https://doi.org/10.48550/arXiv.2402.10133>

155. Lemoine, B., & Laforcade, P. (2024). Generator of personalised training games activities: A conceptual design approach. In P. Dondio et al. (Eds.), *Games and Learning Alliance (GALA 2023)* (Lecture Notes in Computer Science, Vol. 14475). Springer. https://doi.org/10.1007/978-3-031-49065-1_31

156. Marcos, T., Faria, A. L., Bermúdez i Badia, S., & Quintal, F. (2025). Task generator 2.0: Integrating interactive technology with personalized task generation. *Procedia Computer Science*, 256, 1285–1293. <https://doi.org/10.1016/j.procs.2025.02.240>

157. McCabe, T.J. (1976) A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2, 308-320. <https://doi.org/10.1109/TSE.1976.233837>

158. Jayasekara, C., Kopp, C., Lee, V., & Arora, C. (2025). Enhancing

Engagement and Learning in Computing Education: Automated Moodle-Based Problem-Solving Assessments. Preprint. *arXiv*. <https://arxiv.org/abs/2508.17191>

159. Peeß, P., Brocker, A., Röpke, R., & Schroeder, U. (2023). A grammar and parameterization-based generator for Python programming exercises. In *Proceedings of the Sixth Workshop “Automatische Bewertung von Programmieraufgaben” (ABP 2023)*. Gesellschaft für Informatik. <https://doi.org/10.18420/abp2023-6>.

160. Booth, T. L., & Thompson, R. A. (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers*, 22(5), 442–450. <https://doi.org/10.1109/T-C.1973.223746>

161. Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools (2nd ed.)*. Addison-Wesley.

162. Flowise. (без дати). <https://flowiseai.com/>

163. LangChain. (без дати). <https://www.langchain.com/>

164. Yuvzhenko, D., Chymshyr, V., Shymkovych, V., Znova, K., Nowakowski, G., & Telenyk, S. (2025). A multimodal retrieval-augmented generation system with ReAct agent logic for multi-hop reasoning. *Information, Computing and Intelligent Systems*, (6), 42–57. <https://doi.org/10.20535/2786-8729.6.2025.330777>

165. ANTLR (без дати). <https://www.antlr.org/>

166. Parr, T. (2013). *The definitive ANTLR 4 reference (2nd ed.)*. Pragmatic Bookshelf.

167. Zabasta, A., Kazymyr, V., Drozd, O., Verslype, S., Espeel, L., & Bruzgiene, R. (2024). Development of shared modeling and simulation environment for sustainable e-learning in the STEM field. *Sustainability*, 16(5), Article 2197. <https://doi.org/10.3390/su16052197>

168. Khyzhniak, A. V., & Prila, O. A. (2025). Rozrobka systemy avtomatyzovanoi heneratsii ta perevirky parametryzovanykh praktychnykh zavdan [Designing a system for automated generation and automated assessment of parameterized practical assignments]. *Technical sciences and technologies*, (2(40)), 221–233. [https://doi.org/10.25140/2411-5363-2025-2\(40\)-221-233](https://doi.org/10.25140/2411-5363-2025-2(40)-221-233)

169. MCP Protocol. (без дати). Model Context Protocol documentation. <https://modelcontextprotocol.io/docs/getting-started/intro>
170. Khyzhnyak, A., & Mylytsia, A. (2024). Fundamentals of IP telephony: Methodological guidelines to practical exercises and independent study. Chernihiv Polytechnic National University. <http://ir.stu.cn.ua/123456789/31174>.
171. Khyzhniak, A. V., Kazymyr, V. V., & Mylytsia, A. Yu. (2025). The information technology for automated personalized practical tasks creation and assessment. *Tavriyskyi naukovyi visnyk. Seriya: Tekhnichni nauky*, (6), 174–185. <https://doi.org/10.32782/tnv-tech.2025.6.18>
172. Khyzhniak, A. V., & Kazymyr, V. V. (2025). Integrated task generation, execution, and assessment methods for enhancing personalized learning. *Nauka i tekhnika syohodni*, 13(54), 1650–1664. [https://doi.org/10.52058/2786-6025-2025-13\(54\)-1650-1664](https://doi.org/10.52058/2786-6025-2025-13(54)-1650-1664)
173. Brown, M., Dehoney, J., & Millichap, N. (2015). *The next generation digital learning environment: A report on research*. EDUCAUSE. <https://library.educause.edu/~media/files/library/2015/4/eli3035-pdf.pdf>.
174. Kranov, A. A., Schmeckpeper, E. R., & Beyerlein, S. W. (2025, June 22–25). The engineering professional skills assessment 2.0: Preparing engineering students for global workplace complexities. In *Proceedings of the ASEE Annual Conference & Exposition*. Montreal, Quebec, Canada. <https://doi.org/10.18260/1-2--57219>
175. Cahillane, M., Anderson, T., MacLean, P., & Smy, V. (2025). Competence retention analysis: A technique for predicting and managing retention within organizational training design and delivery. *Journal of Cognitive Engineering and Decision Making*. <https://doi.org/10.1177/15553434251372444>.
176. Coursera. (без дати). <https://www.coursera.org/>

ДОДАТКИ

Додаток А. Список публікацій здобувача за темою дисертації

1. Хижняк, А., & Пріла, О. (2025). Розробка системи автоматизованої генерації та перевірки параметризованих практичних завдань. Технічні науки та технології, (2 (40), 221–233. DOI: [https://doi.org/10.25140/2411-5363-2025-2\(40\)-221-233](https://doi.org/10.25140/2411-5363-2025-2(40)-221-233) (1,52 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології, UML-проектування компонентів підсистеми персоналізованого практичного навчання) (1,06 ум. друк. арк.)
2. Хижняк, А., & Казимир, В. (2025). Доменно-орієнтована мова опису персоналізованих практичних завдань для інженерних спеціальностей. Технічні науки та технології, (3 (41), 261–271. DOI: [https://doi.org/10.25140/2411-5363-2025-3\(41\)-261-271](https://doi.org/10.25140/2411-5363-2025-3(41)-261-271) (1,28 ум. друк. арк.) (Особистий внесок здобувача: розробка формальної граматики домен-специфічної мови, тестування граматики, парсер та транслятор домен-специфічної мови для опису практичних інженерних завдань, проектування графічного редактора мови) (0,64 ум. друк. арк.)
3. Khyzhniak A. V.; Kazymyr V. V. Analysis of Methods for Supporting Personalization in IT Education. Publ. Nauka i Tekhnika. Odesa: Ukraine. Herald of Advanced Information Technology 8 (3), 366–381. DOI: <https://doi.org/10.15276/hait.08.2025.24> (1,87 ум. друк. арк.) (Особистий внесок здобувача: системний аналіз існуючих методів персоналізації практичних завдань, виявити обмеження існуючих підходів) (0,93 ум. друк. арк.)
4. Khyzhniak A. V.; Kazymyr V. V. Integrated task generation, execution, and assessment methods for enhancing personalized learning. Nauka i tehnica syogodni, 13(54), 2025. pp.1650-1664. DOI: [https://doi.org/10.52058/2786-6025-2025-13\(54\)-1650-1664](https://doi.org/10.52058/2786-6025-2025-13(54)-1650-1664) (1,75 ум. друк. арк.)

- (Особистий внесок здобувача: удосконалення методів автоматичної генерації практичних завдань, автоматичного розгортання навчального середовища та автоматичної перевірки таких завдань) (0,88 ум. друк. арк.)
5. Хижняк, А. В., Казимир, В. В., & Милиця, А. Ю. (2025). Інформаційна технологія для автоматизації створення та оцінювання персоналізованих практичних завдань. *Таврійський науковий вісник. Серія: Технічні науки*, (6), 174-185. DOI: <https://doi.org/10.32782/tnv-tech.2025.6.18> (1,40 ум. друк. арк.)
(Особистий внесок здобувача: розробка інформаційної технології персоналізації навчання, яка забезпечує інтеграцію моделей, методів та програмних засобів у єдиний комплекс автоматизованої генерації, виконання та перевірки практичних завдань) (0,47 ум. друк. арк.)
 6. Хижняк А. В., Казимир В. В. Моделі персоналізації навчання в цифровому освітньому середовищі. *Технічні науки та технології*. No 1(43), 279-290. DOI: [https://doi.org/10.25140/2411-5363-2026-1\(43\)-279-290](https://doi.org/10.25140/2411-5363-2026-1(43)-279-290) (1,40 ум. друк. арк.)
(Особистий внесок здобувача: функціональна модель життєвого циклу персоналізованого практичного завдання, формальна модель практичного завдання, архітектура персонального AI-асистента) (0,7 ум. друк. арк.)
 7. Мізюк, В. А., Хижняк, А. В., & Хренова, В. В. (2025). Використання адаптивних навчальних платформ для персоналізації дистанційного навчання. <https://doi.org/10.5281/zenodo.14605125> (2,45 ум. друк. арк.) (Особистий внесок здобувача: аналіз навчальних платформ адаптивного навчання) (0,82 ум. друк. арк.)
 8. Khyzhniak A., Mylytsia A. Opportunities for using blockchain-technology and NFTs in building a digital learning environment. *Nauka i tehnica syogodni*, 6(47), 2025. pp.864-874. DOI: [https://doi.org/10.52058/2786-6025-2025-6\(47\)-864-874](https://doi.org/10.52058/2786-6025-2025-6(47)-864-874) (1,28 ум. друк. арк.) (Особистий внесок здобувача: побудова цифрового навчального середовища) (0,64 ум. друк. арк.)
 9. Khyzhniak A.V., Kazymyr V.V. A generalized classification of personalization levels in practical assignments for IT-education. *Nauka i tehnica syogodni*, 7(48),

2025. pp.1932-1949. DOI:
[https://doi.org/10.52058/2786-6025-2025-7\(48\)-1932-1949](https://doi.org/10.52058/2786-6025-2025-7(48)-1932-1949) (2,10 ум. друк. арк.)
(Особистий внесок здобувача: система кількісних показників оцінки персоналізації для наукового аналізу, порівняння, та подальшого вдосконалення персоналізованого навчання) (1,05 ум. друк. арк.)
10. A.Khyzhniak, O.Prila, P.Byvoino, S.Lytvyn. The necessity, preconditions and consequences of using gamification in the educational process. Новітні технології у науковій діяльності і навчальному процесі : збірник тез доповідей Всеукр. наук.-практ. конф. студентів, аспірантів та молодих учених (м. Чернігів, 19- 20 квітня 2023 р.) . - Чернігів : НУ «Чернігівська політехніка» 2023. с 56-58. Режим доступу: <http://ir.stu.cn.ua/handle/123456789/27778> (0,35 ум. друк. арк.)
(Особистий внесок здобувача: використання гейміфікації в навчальному процесі для підвищення персоналізації) (0,09 ум. друк. арк.)
11. Хижняк А. В, Киселиця С. В. Переваги та ризики гейміфікації в соціально-освітньому контексті. Юність науки – 2023: соціально-економічні та гуманітарні аспекти розвитку суспільства : збірник тез доповідей XIII Міжнародної науково-практичної конференції студентів, аспірантів і молодих вчених (м. Чернігів, 26-27 квітня 2023 р.). – Чернігів : НУ «Чернігівська політехніка», 2023. – 770 с. Режим доступу: <http://ir.stu.cn.ua/handle/123456789/28308> (0,35 ум. друк. арк.) (Особистий внесок здобувача: вплив гейміфікації на навчання) (0,18 ум. друк. арк.)
12. Khyzhniak Andrii, Olga Prila, Svitlana Lytvyn An automated system for generating personalised practical tasks and their automatic assessment in distance learning. Юність науки – 2024 : збірник тез доповідей XIV Міжнар. наук.-практ. конф. студ., асп. і мол. вчен. (м. Чернігів, 24-26 квіт. 2024 р.). – Чернігів : НУ «Чернігівська політехніка», 2024. pp. 765-767. Режим доступу: <http://ir.stu.cn.ua/handle/123456789/30262> (0,35 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,12 ум. друк. арк.)

- 13.Хижняк А.В. , Пріла О.А. Актуальні проблеми дистанційного навчання ІТ-спеціалістів. Новітні технології сучасного суспільства (НТСС-2024) : V Міжнародна науково-практична конференція (м. Чернігів, 12 грудня 2024 р.) : тези доповідей – Чернігів : НУ «Чернігівська політехніка», 2025. – 346 с. Режим доступу: https://inel.stu.cn.ua/ntss/NTSS_2024_zbirnyk.pdf (0,35 ум. друк. арк.) (Особистий внесок здобувача: проблеми контролю академічної доброчесності при дистанційному навчанні) (0,18 ум. друк. арк.)
- 14.Khyzhniak A., Prila O. Modeling and implementation of the system for automatized generation and evaluation of personalized practical assessments. Global Trends in Science, Technology, and Economy: Collection of Scientific Papers "International Scientific Unity" with Proceedings of the 1st International Scientific and Practical Conference. April 16-18, 2025. Graz, Austria. 328 p. DOI: 10.70286/ISU-16.04.2025. (0,35 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,18 ум. друк. арк.)
- 15.Хижняк А.В., Пріла О.А. Концептуальна модель системи автоматизації створення та оцінювання персоналізованих практичних завдань. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 98): матеріали Міжнародної наукової інтернет-конференції, (м. Тернопіль, Україна, м. Ополе, Польща, 15-16 квітня 2025 р.) /редкол. : О. Патряк та ін. ГО “Наукова спільнота”, WSZIA w Opolu. Тернопіль : ФОП Шпак В.Б. 2025. 82 с. – ISSN 2522-932X. Режим доступу: http://www.konferenciaonline.org.ua/data/downloads/file_1747665751.pdf (0,47 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,23 ум. друк. арк.)
- 16.Khyzhniak A., Kazymyr V., Zabasta A. domain specific language for personalized practical learning tasks description. Abstracts of XXVII International Scientific and Practical Conference. July 07-09, 2025. Munich, Germany. pp.127-130. Режим доступу:

- <https://eu-conf.com/wp-content/uploads/2025/05/CURRENT-TRENDS-IN-THE-DEVELOPMENT-OF-MODERN-EDUCATIONAL-TECHNOLOGIES.pdf> (0,47 ум. друк. арк.) (Особистий внесок здобувача: вимоги до DSL, яка покриває домен персоналізованих практичних інженерних завдань, формальна граматика домен-специфічної мови) (0,16 ум. друк. арк.)
- 17.Khyzhniak A. Levels and Index of Personalization of Practical Tasks in IT Education. 5th International on educational technology conference and online learning, ICETOL-2025. Abstract proceedings. 26-29 August 2025. Balikesir, Turkey. Режим доступу: https://www.icetol.com/wp-content/uploads/2025/10/icetol2025_abstract_proceedings.pdf (0,12 ум. друк. арк.) (Особистий внесок здобувача: система кількісних показників оцінки персоналізації) (0,12 ум. друк. арк.)
- 18.Khyzhniak A. Information Technology for Personalized Practical Task Creation and Assessment. 5th International on educational technology conference and online learning, ICETOL-2025. Abstract proceedings. 26-29 August 2025. Balikesir, Turkey. Режим доступу: https://www.icetol.com/wp-content/uploads/2025/10/icetol2025_abstract_proceedings.pdf (0,12 ум. друк. арк.) (Особистий внесок здобувача: розробка архітектури системи, створення інформаційної технології) (0,12 ум. друк. арк.)
- 19.Khyzhniak A. A Domain-Specific Language for Describing Personalized Practical Tasks in IT Education. 5th International on educational technology conference and online learning, ICETOL-2025. Abstract proceedings. 26-29 August 2025. Balikesir, Turkey. Режим доступу: https://www.icetol.com/wp-content/uploads/2025/10/icetol2025_abstract_proceedings.pdf (0,12 ум. друк. арк.) (Особистий внесок здобувача: вимоги до DSL, яка покриває домен персоналізованих практичних інженерних завдань, формальна граматика домен-специфічної мови) (0,12 ум. друк. арк.)
- 20.Khyzhniak A. Models and methods of personalization in engineering education.

- Modern Challenges in Economic and Technological Innovation: Collection of Scientific Papers with Proceedings of the 1st International Scientific and Practical Conference. International Scientific Unity. October 15-17, 2025. Bologna, Italy. pp. 158-162. DOI: <https://doi.org/10.70286/isu-15.10.2025> (0,58 ум. друк. арк.) (Особистий внесок здобувача: удосконалення методів автоматичної генерації практичних завдань, автоматичного розгортання навчального середовища та автоматичної перевірки таких завдань) (0,58 ум. друк. арк.)
21. Хижняк А.В., Казимир В.В. Вбудована модель персоналізованого практичного завдання в онлайн-освіті IT-фахівців. МОДС 2025: тези доповідей XX міжнародної конференції (10 – 12 листопада 2025 р., м. Чернігів) / М-во освіти і науки України; Нац. Акад. наук України; Академія технологічних наук України; Інженерна академія України та ін. – Чернігів : НУ «Чернігівська політехніка», 2025. – с.67-72.(0,70 ум. друк. арк.) (Особистий внесок здобувача: формальну модель практичного завдання, що визначає його структуру, параметри, середовище виконання та критерії оцінювання, придатну до автоматизованої обробки; порівняння можливостей різних LLM-моделей для автоматизованої генерації) (0,35 ум. друк. арк.)
22. Khyzhniak Andrii, Kazymyr Volodymyr. Hybrid methods for automated generation, deployment, and verification of personalized practical tasks. Новітні технології сучасного суспільства (НТСС-2025): VI Міжнародна науково-практична конференція (м. Чернігів, 11 грудня 2025 р.) : тези доповідей – Чернігів : НУ «Чернігівська політехніка», 2026. pp 164-165. (0,35 ум. друк. арк.) (Особистий внесок здобувача: удосконалення методів автоматичної генерації практичних завдань, автоматичного розгортання навчального середовища та автоматичної перевірки таких завдань) (0,18 ум. друк. арк.)
23. Khyzhniak A., Mylytsia A. On the lack of unified requirements for virtual learning environments in IT education. 5th International Scientific and Practical Conference «Modern Trends in the Development of Economy, Technology and Industry» Collection of Scientific Papers. January 7-9, 2026. Toronto, Canada. pp

515-517.(0,35 ум. друк. арк.) (Особистий внесок здобувача: визначення вимог до формалізації практичних завдань і навчальних середовищ) (0,18 ум. друк. арк.)

- 24.Хижняк А., Милиця А., Казнадій С., Горваль Д., Бобришев Є. Проблеми та перспективи сучасного практичного навчання інженерів. Львівський науковий форум. Матеріали XVII міжнародної науково-практичної конференції “Пріоритетні напрями досліджень в науковій та освітній діяльності”. 9-10 січня 2026 року. Львів, Україна.с.49-53. (0,58 ум. друк. арк.) (Особистий внесок здобувача: концепція персоналізованого навчання інженерних спеціальностей) (0.12 ум. друк. арк.)

Додаток Б. Довідки про впровадження

CERTIFICATE

of application of A.V. Khyzhniak's PhD thesis
"Models, methods and information technology of personalized learning
in engineering specialties"

This certificate confirms the results of A.V. Khyzhniak's PhD thesis that include new methods of engineering learning based on the developed Learning Task Definition Language (LTDL), integrated with LLM model and AI-assistant embedded to the Digital Learning Ecosystem (DLE) in framework of ERASMUS+ 101127683 — DIGITRANS — ERASMUS-EDU-2023-CBHE project. The implemented software solution consists of:

- ✓ A DLE plugin serving as an interface for a student's personal AI-assistant;
- ✓ AI-agents as part of the AI Assistant for practical tasks generation, execution environment deployment and automatic verification of the students' work results.

Proposed methods allow to increase the level of digitalization in engineering education to acquire by students the sustainable practical skills along an individual learning trajectory with online environment.

Dr Sc Ing Anātolījs Zabašta
DIGITRANS
Project Coordinator
Riga Technical University

Dr Sc Ing Oskars Krievs
Director of Institute of Industrial
Electronics, Electrical Engineering and
Energy,
Riga Technical University



Riga, 9th April, 2026

МІНІСТЕРСТВО ОСВІТИ І
НАУКИ УКРАЇНИ



MINISTRY OF EDUCATION AND
SCIENCE OF UKRAINE

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

вул. Шевченка, 95, Чернігів, 14030,
Україна

тел. +38(0462) 665-103;
факс +38(0462) 665-105
E-mail: estu@stu.cn.ua
www.stu.cn.ua
Код ЄДРПОУ 05460798

CHERNIHIV POLYTECHNIC
NATIONAL UNIVERSITY

95, Shevchenko str., Chernihiv, 14030,
Ukraine

02/04.1026 № 102/20-200

На № _____ від _____

ДОВІДКА ПРО ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Хижняка Андрія Васильовича
“Моделі, методи та інформаційна технологія персоналізованого навчання з
інженерних спеціальностей”, представленої на здобуття наукового ступеня доктора
філософії за спеціальністю 122 - Комп'ютерні науки.

Основні положення та результати дисертаційного дослідження Хижняка Андрія Васильовича на тему “Моделі, методи та інформаційна технологія персоналізованого навчання з інженерних спеціальностей” використовуються в навчальному процесі Національного університету «Чернігівська політехніка»:

1. На кафедрі інформаційних та комп'ютерних систем при проведенні лекцій та лабораторних робіт з дисциплін “Операційні системи”, “Організація комп'ютерних мереж” та “Сучасні телекомунікаційні системи та IP-телефонія” в процесі навчання бакалаврів та магістрів спеціальності F7 (123) – комп'ютерна інженерія та з дисципліни «Методи та технології математичного та комп'ютерного моделювання складних систем» в процесі навчання аспірантів спеціальності F3 (122) – комп'ютерні науки.

2. В загально університетській системі Moodle:

- як плагін до цифрової навчальної екосистеми “DLE” у складі розподіленого середовища моделювання «SMSE» та розподіленого середовища віддаленого експерименту “SREE”, який забезпечує використання студентами персонального ШІ-асистента при виконанні індивідуальних практичних завдань;
- як веб-застосунок, який дозволяє викладачам створювати та редагувати згенеровані за допомогою розробленої інформаційної технології персоналізовані практичні завдання.

Проректор з науково-педагогічної роботи
д.т.н., професор



В.В. Кальченко



SendPulse Inc

220 East 23rd St, Office 401
New York
NY 10010, USA
tel. 650-504-5968

CERTIFICATE OF IMPLEMENTATION
of the results of the PhD thesis by A.V. Khyzhniak, titled "Models, Methods and
Information Technology of personalized learning in engineering specialties"

To Whom It May Concern,

April 2nd, 2026

This is to certify that the results of the Doctor of Philosophy (PhD) thesis
"Models, Methods and Information Technology of personalized learning in engineering
specialties" (author: A.V. Khyzhniak)

have been implemented in the company's educational activities, in particular
through the introduction of a personalized AI assistant.

The results obtained were used to:

- improve the process of educational content development;
- enhance personalized learning mechanisms;
- optimize knowledge assessment processes;
- improve the effectiveness of interaction between learners and the digital learning environment;
- support the development of LMS platform functionality;
- enable the implementation of intelligent learning support systems;
- facilitate the scaling of online courses with personalized learning scenarios.

The results of this research **demonstrate practical value and are recommended for further application** in digital educational environments.

Sincerely,

Evgeny Medvednikov
Chief Executive Officer

ТОВ „ПОРТА УАН-ЧЕРНІГІВ”
14001, Україна, м. Чернігів, вул. Івана Мазепи, 100
тел. моб. +380631756549 e-mail: portaone.chernihiv@gmail.com

Додаток В. Сертифікати участі в конференціях



CERTIFICATE OF PARTICIPATION

Andrii Khyzhniak

has participated successfully in the International Conference of Educational Technology and Online Learning and presented the paper entitled "Information Technology for Personalized Practical Task Creation and Assessment" held in Cunda, Ayvalık, Balıkesir, Türkiye, Aug 26-29, 2025.

Prof. Dr. Gürhan DURAK
Conference Chair





5th INTERNATIONAL CONFERENCE
ON EDUCATIONAL TECHNOLOGY AND ONLINE LEARNING

CERTIFICATE OF PARTICIPATION

Andrii Khyzhniak

has participated successfully in the International Conference of Educational Technology and Online Learning and presented the paper entitled "A Domain-Specific Language for Describing Personalized Practical Tasks in IT Education" held in Cunda, Ayvalık, Balıkesir, Türkiye, Aug 26-29, 2025.

Prof. Dr. Gürhan DURAK
Conference Chair



5th INTERNATIONAL CONFERENCE
ON EDUCATIONAL TECHNOLOGY AND ONLINE LEARNING

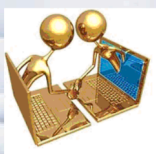
CERTIFICATE OF PARTICIPATION

Andrii Khyzhniak

has participated successfully in the International Conference of Educational Technology and Online Learning and presented the paper entitled "A Domain-Specific Language for Describing Personalized Practical Tasks in IT Education" held in Cunda, Ayvalık, Balıkesir, Türkiye, Aug 26-29, 2025.

Prof. Dr. Gürhan DURAK
Conference Chair





www.konferenciaonline.org.ua

C E R T I F I C A T E

is hereby granted to

Хижняк Андрій Васильович

for participation in the International Scientific Internet Conference «Information society: technological, economic and technical aspects of formation» (issue 98)

with a publication on the topic:

**«КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ АВТОМАТИЗАЦІЇ
СТВОРЕННЯ ТА ОЦІНЮВАННЯ ПЕРСОНАЛІЗОВАНИХ
ПРАКТИЧНИХ ЗАВДАНЬ».**

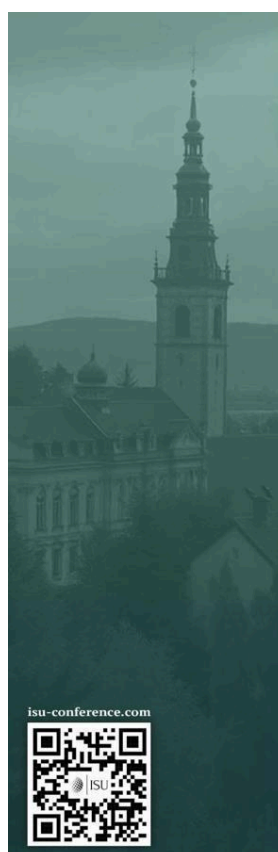
Form of participation: remotely, duration of conference is 18 hours 0,6 ECTS credits

Conference was held with the assistance and participation of Public Organization "Scientific Community" and Wyższa Szkoła Zarządzania i Administracji w Opolu



Ternopil - Opole
April 15-16, 2025

KO №01701



CERTIFICATE

of conference participant

it is hereby certified, that

ANDRII KHYZHNIAK

took part in the 1st International Scientific and Practical Conference
«GLOBAL TRENDS IN SCIENCE, TECHNOLOGY, AND ECONOMY»

April 16-18, 2025, Graz, Austria
24 Hours of Participation
(0,8 ECTS credits)



Head of the
organizing committee



Viktoriia Tsiundyk

ISU-25/0416-020





isu-conference.com



ISU
INTERNATIONAL SCIENTIFIC UNITY

CERTIFICATE

of conference participant

it is hereby certified, that
ANDRII KHYZHNIAK
took part in the 1st International Scientific and Practical Conference
«**MODERN CHALLENGES IN ECONOMIC AND
TECHNOLOGICAL INNOVATION**»
October 15-17, 2025, Bologna, Italy
24 Hours of Participation
(0,8 ECTS credits)

doi g iku R ISO OPEN ACCESS

Head of the
organizing committee



Viktoriia Tsiundyk

ISU-25/1015-010




isu-conference.com



ISU
INTERNATIONAL SCIENTIFIC UNITY

CERTIFICATE

of conference participant

it is hereby certified, that
ANDRII KHYZHNIAK
took part in the 5th International Scientific and Practical Conference
«**MODERN TRENDS IN THE DEVELOPMENT OF ECONOMY,
TECHNOLOGY AND INDUSTRY**»
January 7-9, 2026, Toronto, Canada
24 Hours of Participation
(0,8 ECTS credits)

doi g iku R ISO OPEN ACCESS

Head of the
organizing committee



Viktoriia Tsiundyk

ISU-26/0107-178



CERTIFICATE OF PARTICIPATION

The XXVII International Science Conference
«Current trends in the development
of modern educational technologies»

This is to certify the participation in the conference and the publica-
tion of the article in the corresponding proceedings

Khyzhniak Andrii

12 Hours of Participation (0,4 ECTS credits)
JULY 09-11, 2025
MUNICH, GERMANY



ЛЬВІВСЬКИЙ НАУКОВИЙ ФОРУМ

СЕРТИФІКАТ

№ 2601-016

Даний сертифікат підтверджує, що

Хижняк А., Милиця А., Казнадій С., Горваль Д., Бобришев Є.
взяв(ла) участь у роботі

XVII Міжнародній науково-практичній конференції
«Пріоритетні напрями досліджень в науковій та освітній діяльності»
тривалістю 15 годин / 0,5 кредиту ЄКТС



9-10 січня 2026 року

Голова оргкомітету конференції

В. Бондар

Додаток Г. Засоби реалізації вимог до VLE

Таблиця Г.1

Засоби реалізації вимог до VLE

Вимога	Засоби реалізації
Віддалене розгортання	Клієнт-серверна архітектура Розгортання у хмарних провайдерів Обмежений доступ по SSH Дикористанням гіпервізорів чи систем оркестрації
Реалістичність	Імітація архітектури мікросервісів
Автентичність	Оркестрація Provisioning Scripts
Обструфікованість	Санітарна обробка середовища (Environment Sanitization) Очищення історії Використання інструментів керування конфігурацією, які не залишають слідів у локальній оболонці VLE Видалення тимчасових файлів Запікання образу (Image Baking)
Ізольованість	Ізоляція файлових систем Апаратна ізоляція Використання VLAN Мережеві політики
Безпечність	Запуск процесів без прав суперкористувача Обмеження системних викликів Апаратна віртуалізація
Контрольованість	Обмеження ресурсів, контейнеризація Обмеження мережі, контроль вихідного трафіку
Інтероперабельність	Стандартизовані освітні протоколи (зокрема LTI) Наскрізна аутентифікація та засоби єдиного входу Інтеграція через API, вебхуки та пакетну синхронізацію
Кросплатформність	Рівень абстракції Модульна архітектура
Продуктивність	Резервування, кластеризація Load Balancing, моніторинг навантаження Реплікація баз даних, оптимізація запитів до БД Оркестрація та контейнеризація Image Baking та Golden Image

Додаток Д. Застосування процедури декомпозиції для типових завдань

Таблиця Д.1.

Декомпозиція завдань на 7 компонентів

Дисципліна	Приклад завдання	Контекст	Вхідні дані	Мета	Метод (невиявний)	Середовище	Інструменти	Критерій якості
Програмна інженерія	"Реалізуй функцію сортування масиву за допомогою швидкого сортування"	Структури даних; задача сортування	Невідсортований масив цілих чисел	Відсортований масив у порядку зростання	Розділяй і володарюй, розбиття, рекурсія, алгоритм: швидке сортування	Середовище виконання мови програмування; платформа виконання	Мова програмування (Python, C++); IDE; опціональні AI-асистенти коду	Коректність; часова складність $O(n \log n)$; читабельність коду
Системи баз даних	"Напиши SQL-запит для пошуку студентів з $GPA > 3.5$, зарахованих на CS"	Реляційна БД; Таблиці: Students, Enrollments	Схема БД; атрибути таблиць	Результуюча множина, що задовольняє умовам	SQL SELECT з JOIN та WHERE	Середовище виконання СУБД	SQL-рушій; редактор запитів; клієнтські інструменти БД	Коректність запиту; ефективність; правильне використання індексів
Комп'ютерні мережі	"Розрахуй пропускну здатність TCP-з'єднання при $RTT=100\text{мс}$, втрати пакетів=1%, розмір вікна=64КБ"	Протокол TCP; модель продуктивності мережі	RTT, рівень втрат, розмір вікна	Значення пропускну здатності (Мбіт/с)	Формула пропускну здатності TCP; модель розмірів вікна	Аналітичне / обчислювальне середовище	Калькулятор; інструменти скриптингу; електронні таблиці	Розмірна коректність; припущення протоколу; числова точність
Алгоритми	"Визнач часову складність вкладеного циклу: зовнішній виконується n разів, внутрішній — $\log(n)$ разів"	Теорія обчислювальності складності	Опис структури циклу	Вираз у нотації Big-O	Правила аналізу складності, логарифмічне зростання	Теоретичне (невиконуване) середовище	Міркування на папері; інструменти символьного аналізу	Коректна асимптотична класифікація; обґрунтовані міркування
Термодинаміка	"Розрахуй роботу ідеального газу при ізотермічному розширенні $V_1=1\text{л}$ до $V_2=3\text{л}$ при $T=300\text{К}$ "	Модель ідеального газу; ізотермічний процес	V_1, V_2, T , газова стала R	Робота W (Джоулі)	Аналітичне інтегрування термодинамічних рівнянь $W = nRT \ln(V_2/V_1)$	Фізичне моделювання з припущеннями	Таблиці формул; калькулятори; інструменти симуляції	Узгодженість одиниць; фізична обґрунтованість; числова точність

Додаток Е. Опис компонентів формальної моделі РРЕТ

Компонент Контекст С (Context) концептуально описує фреймворк, в рамках якого інтерпретується завдання. Компонент Контекст визначається як

$$C = \langle C_{domain}, C_{topic}, C_{desc}, C_{cons}, C_{comp} \rangle$$

де:

C_{domain} визначає предметну область (наприклад, «структури даних», «мережеві протоколи», «системи баз даних», «криптографія»);

C_{topic} позначає тему, що аналізується (наприклад, «реляційна база даних», «стек TCP/IP», «бінарне дерево пошуку»);

C_{desc} опис змісту завдання;

C_{cons} набір припущень та обмежень (наприклад, «ідеальні умови», «максимальна тривалість»);

C_{comp} характеристика обчислювальних або когнітивних зусиль, пов'язаних з виконанням завдання.

Зазначені елементи утворюють мінімально достатній інваріантний ядро, що забезпечує семантичну узгодженість специфікацій завдань, без перерахування всіх можливих контекстуальних уточнень.

Визначимо простори значень, з яких вибираються елементи: D – множина допустимих предметних областей (доменів); T – множина допустимих тем (тем); S — множина рядків (текстів) для опису (Strings); K – множина всіх можливих обмежень (Constraints); L – множина рівнів складності (наприклад, {1,...,5} або {Low,Medium,High}).

Компонент С є дійсним (Valid(C)) тоді і тільки тоді, коли сукупність всіх умов є істинною.

$$Valid(C) \Leftrightarrow (C_{domain} \in D) \wedge (C_{topic} \in T) \wedge Rel(C_{topic}, C_{domain})$$

$$\wedge (C_{desc} \in S \wedge C_{desc} \neq \emptyset) \wedge (C_{comp} \in L) \wedge (C_{cons} \subseteq K \wedge Consistent(C_{cons}))$$

Компонент Дані D (Data) представляє типізовані вхідні параметри з відповідними обмеженнями і визначається як

$$D = \langle P_{static}, D_{vars}, S_{init} \rangle,$$

де:

$P_{static} = \{sp_1, sp_2, \dots, sp_n\}$ є набором незмінних параметрів, спільних для всіх випадків завдання. Кожен $sp_i = \langle name, type, value \rangle$.

$D_{vars} = \{v_1, v_2, \dots, v_n\}$ є набором параметризованих даних вхідних, які використовуються для параметризації екземпляра завдання, і кожен v_i визначається як $v_i = \langle name, type, validity \rangle$. Конкретні значення присвоюються тільки на етапі інстанціювання завдання. Валідність елемента представляє набір предикатів, що накладають допустимі умови на один або декілька елементів даних.

S_{init} представляє декларативні артефакти конфігурації, необхідні для переведення середовища в початковий стан (наприклад, дампи баз даних, файли конфігурації, попередньо завантажені файлові структури). Операційна логіка застосування цих артефактів делегована Компоненту Середовища (E).

Для забезпечення семантичної узгодженості з Компонентом Контексту (C) допустима система типів визначається як об'єднання універсальних і доменних типів:

$$type \in Primitive_{types} \cup Composite_{types} \cup$$

$$Abstract_{types} \cup Specific_{types}(C_{domain}),$$

де $Specific_{types}$ є функцією відображення, яка повертає домен-специфічні типи для заданого домену в Компоненті С (Context).

Зазначені елементи утворюють мінімально достатнє незмінне ядро, що містить початкову інформацію, необхідну для виконання завдання. Ця структура підтримує формальну перевірку допустимості та узгодженості вхідних даних, дозволяючи при цьому розширення даних для конкретної області під час інстанціювання завдання.

Компонент Ціль G (Goal) визначає бажані результати РРЕТ та набір обов'язкових вимог, яким має відповідати рішення, щоб вважатися дійсним. Він діє як формальна специфікація очікуваного результату, незалежно від схеми оцінювання. Компонент Ціль представляє розбиття практичного завдання на набір чітких, перевірюваних цілей (етапів). Замість монолітної специфікації, ціль визначається як сукупність атомарних підцілей, де кожна підціль інкапсулює цільовий об'єкт та його конкретні вимоги до прийнятності.

$$G = \{g_1, g_2, \dots, g_n\}.$$

Кожна підціль g_i визначається як кортеж, що пов'язує цільовий стан/артефакт з його логікою перевірки і $S_{criteria}$:

$$g_i = \langle S_{target}, O_{spec}, V_{logic}, S_{criteria} \rangle,$$

де

S_{target} - це набір предикатів, що описують обов'язковий стан середовища (E), пов'язаного з цією конкретною підціллю (наприклад, «сервіс працює», «порт відкритий»). Якщо підціль передбачає лише створення артефакту без постійної зміни середовища, $S_{target} = \emptyset$.

O_{spec} описує структурне визначення артефакту, створеного в рамках цієї підцілі, визначається як $O_{spec} = \langle O_{type}, O_{sc}, O_c \rangle$, де O_{type} є очікуваною категорією даних або артефактів; O_{sc} є логічними властивостями, яким повинен відповідати вихідний результат, включаючи вимоги до формату,

кодування та обмеження щодо представлення; O_c вказує, чи завдання очікує на один результат, набір результатів або послідовність результатів. Якщо підзавдання передбачає лише конфігурацію системи (наприклад, «увімкнення брандмауера») без повернення конкретного файлу/значення, $O_{spec} = \emptyset$.

V_{logic} визначає абстрактну процедуру оцінки, що застосовується для визначення відповідності цільового стану або артефакту вимогам. На відміну від Компонента Інструменти S , який містить перелік конкретних інструментів, логіка верифікації визначає клас верифікації (наприклад, «порівняння знімків», «тестування вводу-виводу», «перевірка статичних обмежень»).

$S_{criteria}$ — це набір вимог до прийнятності (жорстких обмежень), характерних для цієї підцілі (наприклад, обмеження часу очікування, межі використання пам'яті, заборонені ключові слова). Ці критерії повинні бути суворо дотримані, щоб підціль вважалася «досягнутою». Елемент $S_{criteria} = \{sc_1, sc_2, ..., sc_n\}$. На відміну від компонента Якість Q , який оцінює, наскільки добре виконано завдання, $S_{criteria}$ визначають, чи є завдання дійсним.

Внутрішня структура Компонента Ціль (Goal) розділяє те, що необхідно досягти, від того, що треба, аби отриманий результат вважався «досягнутим», відображаючи відмінність між специфікацією результату та логікою оцінки. Завдання вважається виконаним у цілому, якщо і тільки якщо досягнуті всі підцілі в наборі G .

$$Task_{complete} \Leftrightarrow$$

$$\forall g_i \in G: (Satisfied(S_{target}) \wedge Satisfied(O_{spec}) \wedge Satisfied(S_{criteria}))$$

Компонент Метод M (Method) визначає абстрактні парадигми вирішення проблем, застосовні до завдання. Він визначає, як завдання може бути вирішене на концептуальному рівні, не прописуючи конкретних інструментів або реалізацій.

Визнаючи, що інженерні проблеми часто допускають кілька дійсних підходів, M визначається як набір допустимих методів:

$$M = \{m_1, m_2, \dots, m_n\}.$$

Кожен допустимий метод m_i визначається як кортеж:

$$m_i = \langle M_{paradigm}, M_{prerequisites}, M_{procedure}, M_{complexity} \rangle,$$

де

$M_{paradigm}$ визначає загальний підхід до вирішення проблеми (наприклад, аналітичне виведення, чисельне наближення, метод «розділяй і володарюй», симуляція, оптимізація).

$M_{prerequisites}$ позначають необхідні концептуальні знання та залежності даних, пов'язані з методом. Ці залежності формалізуються за допомогою відношення залежності $\rho_{dep} \subseteq D \times M$, що пов'язує конкретні вимоги до вхідних даних з обраним методом.

$M_{procedure}$ представляє абстрактну процедурну або декларативну структуру виконання, незалежну від будь-якої конкретної реалізації.

$M_{complexity}$ характеризує обчислювальні (нотація Big-O) або когнітивні зусилля, пов'язані із застосуванням методу.

Компонент Method M визначає теоретичний підхід. Його конкретна реалізація здійснюється за допомогою спеціальних інструментів, визначених у Компоненті Інструменти **Tools** S . Таке розділення дозволяє моделі розрізняти концептуальну помилку (неправильний метод) та інструментальну помилку (неправильне використання інструменту). Методи можуть бути визначені явно або неявно. У явних випадках завдання безпосередньо обмежує застосовний метод. У неявних випадках вибір відповідного методу залишається за учнем в межах допустимого набору методів, визначеного завданням.

Компонент Якість Q (Quality) визначає модель кількісної оцінки, яка використовується для оцінювання поданого рішення. Компонент Goal (G) визначає

бінарні умови для валідності завдання (via $S_{criteria}$), а компонент Quality вимірює, наскільки добре було отримано валідний результат. Він встановлює відповідність між артефактами рішення та числовим балом на основі педагогічних пріоритетів.

Компонент визначається як скінченна множина критеріїв якості:

$$Q = \{q_1, q_2, \dots, q_n\}.$$

Кожен критерій $q_i = \langle Q_{metric}, Q_{threshold}, Q_{weight}, Q_{applicability} \rangle$,

де

Q_{metric} це вимірювана властивість результату (наприклад, час виконання, відсоток покриття коду, цикломатична складність).

$Q_{threshold}$ визначає діапазон цільових значень (інтервал) для метрики.

Попадання в цей діапазон вважається виконанням умови якості. Це відрізняє оптимальну продуктивність від просто прийнятної продуктивності, визначеної в Компоненті G.

Q_{weight} відносна важливість цього критерію ($w_i \in [0, 1]$), де сума всіх ваг дорівнює 1.

$Q_{applicability}$ визначає сферу застосування критерію. Він зіставляє метрику з конкретною підціллю $g_i \in G$ або конкретним елементом у ній (наприклад, застосування метрики «Стиль коду» конкретно до O_{spec} підцілі «Логіка бекенду»).

Компонент Quality не визначає методи, інструменти або умови виконання. Goal визначає, що є прийнятним результатом, а Quality визначає, як оцінюється відповідність. У навчальних VLE, що підтримуються AI і мають багатий набір інструментів, якість не може розглядатися як неявне або суто якісне поняття. У запропонованій моделі РРЕТ компонент якості чітко реалізується шляхом зіставлення критеріїв оцінки з вимірюваними показниками та порогами прийнятності. Такі виміри, як функціональна правильність, безпека, продуктивність,

якість та зручність обслуговування коду, а також надійність, складають мінімальне, незалежне від галузі ядро, яке є як педагогічно значущим, так і технічно перевірюваним.

Вони не призначені як вичерпна таксономія, а скоріше як репрезентативні виміри, які часто виникають в інженерній освіті та програмних РРЕТ. Кожен вимір пов'язаний із конкретними показниками (наприклад, відсоток успішного проходження тестів, рівні серйозності вразливостей, міри складності), чіткими порогами та автоматизованими або напівавтоматизованими механізмами перевірки. Така операціоналізація дозволяє формально визначити та об'єктивно оцінити обмеження якості, незалежно від того, чи результат був отриманий вручну, чи за допомогою AI. Таким чином, Quality підтримує перевіряєму, незалежну від інструментів оцінку РРЕТ в освітніх середовищах, доповнених AI.

Загальний показник якості обчислюється як:

$$Score(result, Task) = \sum_{i=1}^N w_i \cdot (result, q_i) \vee, \text{ де } w_i \text{ задовольняє } \sum_{i=1}^N w_i = 1.$$

Компонент Інструменти S (Tool) чітко визначає програмні інструменти, бібліотеки, платформи та AI-агенти, які використовуються для виконання завдання. Компонент «Інструменти» операціоналізує Компонент Method, обмежуючи конкретні засоби, які можуть бути використані під час виконання завдання. Він не описує логіку вирішення проблем (як Method) і не оцінює результати (як Quality), а обмежує конкретні інструменти, за допомогою яких можуть бути реалізовані методи. Основна функція Компонента Tool полягає в екстерналізації припущень, які зазвичай залишаються неявними в описах завдань, тим самим забезпечуючи узгодженість між заявленими цілями навчання та фактичним процесом виконання. На відміну від традиційних моделей, які розглядають процес вирішення як «чорний ящик», модель РРЕТ розглядає використання інструментів як формальний вимір завдання. Це дозволяє розрізняти різні педагогічні модальності (наприклад, «ручне кодування» проти «генерування за допомогою AI»), навіть якщо кінцевий результат є ідентичним.

Завдяки формальному зв'язку з Method, компонент Tool гарантує, що специфікації завдань розрізняють спосіб отримання результату та те, що становить прийнятний результат.

Хоча традиційні моделі завдань непрямо розглядають інструменти як нерелевантні, незважаючи на їх вирішальний вплив на мислення, процес і результати, запропонована модель не передбачає процедур перевірки та не нав'язує конкретних політик щодо використання AI, а навпаки, дозволяє формально визначити їх як першокласні обмеження завдань. Tool концептуально впроваджено не для контролю за студентами або виявлення використання AI як такого, а для усунення прихованих припущень у формалізації завдань шляхом чіткого визначення засобів виконання.

Компонент визначається як скінченна множина специфікацій інструментів:

$$S = \{s_1, s_2, \dots, s_n\}.$$

Кожен елемент компоненту s_i визначається як

$$s_i = \langle S_{name}, S_{category}, S_{am}, S_{configuration} \rangle,$$

де:

S_{name} це унікальний ідентифікатор інструменту (наприклад, «GCC Compiler», «Docker Engine», «ChatGPT-4», «Kubernetes CLI»).

$S_{category}$ класифікує інструмент для підтримки автоматизованого розгортання та аналізу. Це можуть бути основні утиліти (компілятори, інтерпретатори, оболонки), інфраструктура (контейнерні середовища виконання, бази даних, мережеві симулятори), GenAI (LLM, GitHub Copilot, підказки), довідкові ресурси (офіційна документація, онлайн-підручники та курси, спільноти та форуми, наукові статті), співпраця між людьми (консультації з колегами, настанови викладачів, послуги з перевірки коду, парне програмування).

S_{am} визначає політику дозволів щодо цього інструменту (дозволено, заборонено, обов'язково, можливо за певних умов). Цей вимір є вирішальним

для контролю академічної доброчесності в середовищах, насичених AI. Дозволені інструменти явно дозволені для виконання завдань, заборонені інструменти явно заборонені (порушення призводить до невдачі завдання), обов'язкові інструменти є обов'язковими для виконання завдань, а інструменти, що використовуються за певних умов, дозволені за певних обставин з уточненням інформації.

$S_{configuration}$ це набір обмежень використання або параметрів, характерних для версії або налаштувань інструменту.

Запропонована модель оцінює результати з огляду на заявлені цілі та критерії якості, розглядаючи інструменти як чітко визначені обмеження виконання, а не об'єкти оцінки. Обмеження щодо використання інструментів визначають допустимість виконання завдання, а не сприяють оцінці якості. Хоча інструменти не є частиною самого відношення якості, певні критерії якості можуть вимагати перевірки відповідності обмеженням, пов'язаним з інструментами. Ця залежність відображається опосередковано через Компонент Tool, який гарантує, що оцінка якості проводиться тільки для відповідей, отриманих за допомогою дозволених і належним чином розкритих засобів.

Виконання вважається дійсним стосовно Tool тоді і тільки тоді, коли всі використані інструменти та ресурси належать до дозволених або умовно дозволених підмножин S , всі необхідні інструменти присутні, а заборонені інструменти не виявлені.

$$ValidExecution \Leftrightarrow$$

$$(Usedtools \cap Prohibited = \emptyset) \wedge (Mandatory \subseteq Usedtools)$$

Якщо використовується заборонений інструмент, виконання вважається недійсним і виключається з подальшої оцінки. Критерії якості застосовуються тільки до рішень, які відповідають заявленим обмеженням щодо інструментів та VLE, тим

самим зберігаючи чітке розмежування між дійсністю виконання та оцінкою результату.

Механізми верифікації, пов'язані з Компонентом S, навмисно залишені абстрактними в базовій моделі. Вони вказують на те, що відповідність обмеженням, пов'язаним з інструментами, повинна бути, в принципі, верифікованою, дозволяючи при цьому визначати конкретні стратегії верифікації на рівні реалізації системи або інституційної політики.

Компонент Середовище E (Environment) визначає набір допустимих технологічних інфраструктур та контекстів виконання, в яких завдання може бути виконано належним чином. Враховуючи, що сучасні РРЕТ часто передбачають переносимість між різними платформами (наприклад, локальні контейнери проти хмарних VM), модель визначає E не як окремий екземпляр, а як простір сумісних середовищ виконання.

$$E = \{e_1, e_2, \dots, e_n\}.$$

Кожна специфікація середовища e_i визначається як кортеж:

$$e_i = \langle E_i, E_{run}, E_{res}, EC \rangle,$$

де:

E_i визначає межі технології віртуалізації або виконання (наприклад, Docker Container, Kubernetes Pod, Virtual Machine, Network Simulator).

E_{run} визначає програмний стек і базову конфігурацію, включаючи версію ОС, попередньо встановлені бібліотеки та залежності на рівні системи, до яких застосовуються інструменти користувача та S_{init} .

E_{res} визначає обчислювальні обмеження для забезпечення справедливості та відтворюваності (наприклад, обмеження процесора, квоти пам'яті, пропускна здатність диска).

ЕС визначає операційні політики, включаючи правила доступу до мережі (наприклад, обмеження доступу до Інтернету) та необхідні канали телеметрії для спостереження.

Компонент Environment визначає, де і за яких умов відбувається виконання завдання. Він не визначає, як вирішується завдання (Method) і які інструменти використовуються (Tools), але обмежує і те, і інше. Сучасна освіта працює на різних платформах, що має значний вплив на правильність, продуктивність і відтворюваність. Наприклад: поведінка коду відрізняється в Python 3.8 і 3.11, Windows і Linux; час виконання залежить від апаратного забезпечення, планування ОС, доступності ресурсів тощо. Крім того, VLE безпосередньо впливає на якість та безпеку (результат залежить від конфігурації VLE).

Компонент E прямо стосується критичного усвідомлення того, що код або артефакти завдань не виконуються у вакуумі – VLE фундаментально формує якість та перевірюваність результату. Хоча компонент Environment формально визначений, дане дослідження абстрагується від конкретних технологій розгортання та зосереджується на концептуальних обмеженнях виконання, а не на реалізаціях на рівні інфраструктури. Концептуальна межа між компонентом Environment та іншими компонентами моделі проілюстрована нижче в таблиці E1.

Таблиця E1.

Концептуальна межа між Environment та іншими компонентами

E vs C (Context)	C: "Web application with REST API" (conceptual architecture) E: "Node.js 18.x on Ubuntu 22.04 with PostgreSQL 14" (concrete infrastructure)
E vs S (Tools)	S: "May use Express.js framework, GitHub Copilot prohibited" (development tools) E: "Runs on Node.js runtime with 2GB memory limit" (execution platform)
E vs D (Data)	D: "Input: array of integers, size $\leq 10^6$ " (problem inputs) E: "Available memory: 512MB" (execution resources)

Хоча компонент Environment безпосередньо не бере участі в оцінці якості, він визначає умови, за яких створюються сліди виконання та артефакти результатів. Отже, параметри VLE можуть впливати на виміряні значення показників якості

(наприклад, продуктивність, споживання ресурсів, відтворюваність), не створюючи прямої залежності між E та Q на рівні моделі.

Взаємодія між Компонентом Data (D) і Компонентом Environment (E) формалізується за допомогою відношення ρ_{prov} . Це відношення визначає сумісність між артефактами, що надаються в S_{init} і можливостями обраного середовища.

Нехай $S_{init} \in D$ множина артефактів конфігурації. Нехай $e \in E$ обране базове середовище. Визначимо функцію переходу Apply:

$$E_{ready} = Apply(e, S_{init})$$

Розгортання вважається дійсним, якщо і тільки якщо Середовище E володіє необхідним Runtime and Infrastructure для інтерпретації та розгортання всіх артефактів в S_{init} .

$$Valid(e, S_{init}) \Leftrightarrow \forall a \in S_{init}, Supported(e, type(a))$$

Наприклад, якщо S_{init} містить SQL дамп, E повинен містити DBMS сервіс.

Додаток Є. Тестування розробленої мови LTDL

В рамках тестування парсингу за допомогою мови Python створені парсинг тести на валідні конструкції та неправильний синтаксис. Для лексичного тестування граматики проведено тестування токенізації, тести на спеціальні символи, граничні випадки, порожні файли, зайві пробіли та довгі вирази.

Для побудови дерева виводу граматики виконано вивід завдання, в якому є параметризовані змінні, треки зі вкладеннями етапів та окремі етапи з перевіркою дією. Опис цього завдання представлено в лістингу Є.1.

```
task "complex_task": {  
  variables = [ variable "topvar1": {  
    type = "RandomString" }]  
  tracks = [ track "t1": {  
    stages = [ stage "stage1": {  
      checks = [check "check1": {action = "a1" }]],  
      stage "stage2": {  
        checks = [ check "check2": {action = "a2" }]]]]]  
}
```

Лістинг Є.1. Опис практичного завдання з кількома етапами.

В результаті синтаксичного та лексичного аналізу побудовано дерево виводу, представлене на рисунку Є.1.

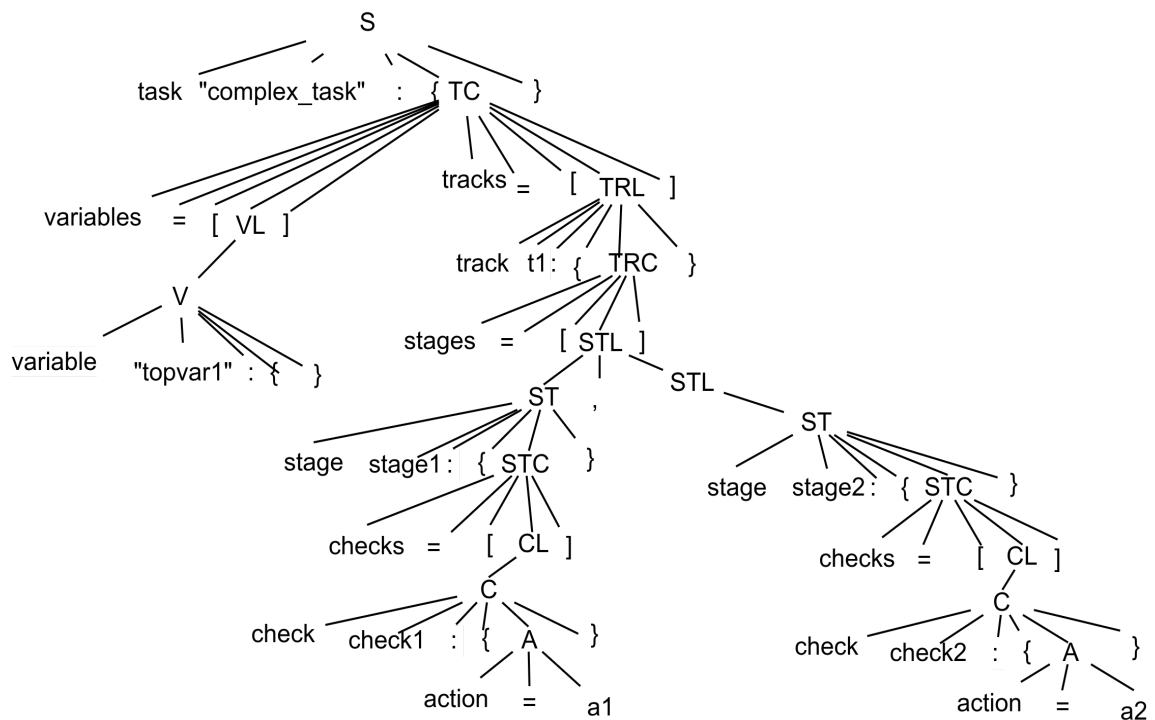


Рис. 6.1. Дерево виводу опису практичного завдання з кількома етапами.

Окремим, але важливим елементом PPET є опис VLE, тому для нього теж побудовано синтаксичне дерево. Для прикладу створено опис VLE, яке являє собою VM, запущену за допомогою гіпервізора VirtualBox. Для налаштування середовища описані інструкції, які створюють користувача і задають йому пароль. Опис VLE середовища представлено в лістингу 6.2.

```
task "just_le": { learning_environment "main": {
variables = [ variable "login: {type = "RandomString" },
               variable "pass: {type = "RandomString" }]
create_instructions = [
instruction "startvm": {
action = "VBoxManage startvm"}]
provision_instructions = [
instruction "create_user": {
```


2. Наявності альтернативних правил виводу;
3. Правильній структурі граматики без циклічних залежностей;
4. Можливості виводу мінімальних конструкцій для кожного правила.

Символ A є досяжним, якщо існує вивід $S \Rightarrow^* \alpha A \beta$, де S - стартовий символ, α і β - довільні рядки символів. Символ A є недосяжним, якщо не існує виводу $S \Rightarrow^* \alpha A \beta$ для жодних α і β .

ГраMATика LTDL не містить недосяжних символів. Всі нетермінальні символи можуть бути досягнуті від стартового символу $task$ через певну послідовність виводів.

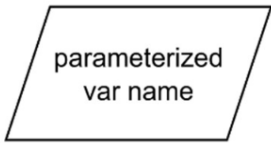


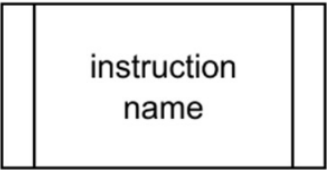
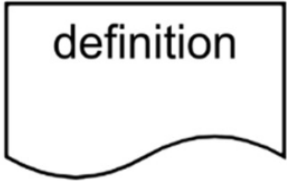
Ключові особливості, які забезпечують досяжність всіх символів:

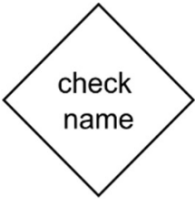

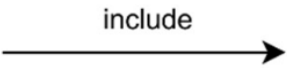
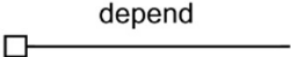
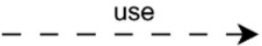
1. Ієрархічна структура – граMATика побудована як дерево з чітким корінням ($task$);
2. Центральне правило – аксіома, що об'єднує всі основні блоки граматики;
3. Взаємозв'язки – всі структури пов'язані через правила виводу;
4. Відсутність ізольованих правил – немає правил, що не використовуються.

Додаток Ж. Графічні зображення символів алфавіту мови LTDL

Таблиця Ж.1.

Графічні зображення символів алфавіту мови LTDL

Графічне представлення символу	Сутність	Опис
	task	Це основна сутність і початкове правило граматики, включає в себе опис завдання, його метадані, змінні, треки і етапи, опис VLE.
	variable	Задає параметризацію завдання для застосування в темплейті або інструкції. Змінна може бути рівня всього завдання, треку або етапу.
	track	Використовується для групування етапів, які залежать один від одного. Це лінійна послідовність виконання кроків, складається зі змінних та етапів.
	stage	Описує один з етапів виконання завдання і складається зі змінних, попередніх умов, опису та перевірок. Попередні умови - це список інструкцій, які треба виконати в екземплярі VLE, перед тим як починається виконання завдання.
	instruction	Описує дію, яка повинна бути виконана для запуску VLE або в ньому. Використовується в описі VLE та в попередніх умовах етапу (stage).
	definition	Описує темплейт, який видається студенту перед тим як він почне виконувати завдання і описує що потрібно виконати в конкретному етапі (stage).

	check	Задає дії, які треба виконати для відслідковування успішності виконання конкретного етапу (stage). Виконуються при підключенні до VLE, в залежності від протоколу.
	learning environment	Опис VLE, в якому буде виконуватися завдання. Має параметри та інструкції по запуску і налаштуванню конкретного екземпляра VLE.
	include	Тип зв'язку, який встановлює вкладеність однієї сутності в іншу - наприклад етап є частиною треку.
	depend	Тип зв'язку який встановлює залежність етапів один від одного.
	use	Тип зв'язку який показує де буде використана параметризована змінна.

Додаток 3. Системний промт для LLM

Застосований системний промт для експерименту.

Role: You are an expert DevOps engineer and IT instructor creating practical engineering tasks for university students.

Task: Generate a comprehensive, automatically deployable laboratory environment based on the user's topic.

Requirements:

Instruction: Provide a clear description of the task, the objective, and step-by-step instructions for the student.

Infrastructure: Generate the necessary configuration files (e.g., docker-compose.yml, Dockerfile, or initialization Bash scripts) to deploy an isolated Virtual Learning Environment (VLE).

Validation Logic: Write an automated test script (e.g., a Bash script check.sh) that verifies if the student has successfully completed the task. It must return exit code 0 on success and 1 on failure.

Personalization: Ensure the task is dynamic. Use parameterized variables (like \${STUDENT_PORT}, \${DB_PASSWORD}) in both the infrastructure code and validation scripts to prevent cheating. Provide an .env template.

Output Format: Output strictly in Markdown format. Use clear headings for each section. Enclose all code in appropriate markdown code blocks with language tags (e.g., ``yaml, ``bash). Do not include conversational filler.

Додаток І. Оцінка якості генерації завдань

Таблиця І.1.

Оцінка якості генерації завдань по різних дисциплінам

Пром т	Текст овий опис та інстру кції	Інфра струк турни й код VLE	Логік а автом атичн ої переві рки	Пара метри зація та динам ічні змінні	Q_J, LLM	Текст овий опис та інстру кції	Інфра струк турни й код VLE	Логік а автом атичн ої переві рки	Пара метри зація та динам ічні змінні	Q_J,L LM+L LM	Текст овий опис та інстру кції	Інфра струк турни й код VLE	Логік а автом атичн ої переві рки	Пара метри зація та динам ічні змінні	Q_J,L LM+L TDL
1.1	0,50	0,50	0,50	1,00	0,55	0,50	1,00	0,50	1,00	0,78	0,50	1,00	1,00	1,00	0,93
1.2	0,50	0,00	0,50	1,00	0,33	0,50	1,00	0,50	1,00	0,78	0,50	1,00	1,00	1,00	0,93
1.3	1,00	0,50	0,00	1,00	0,48	1,00	1,00	0,50	1,00	0,85	1,00	0,50	1,00	1,00	0,78
1.4	0,50	0,00	0,00	0,50	0,13	1,00	1,00	0,50	1,00	0,85	0,50	1,00	1,00	1,00	0,93
1.5	1,00	0,50	0,00	0,50	0,43	1,00	0,50	0,50	1,00	0,63	1,00	1,00	0,50	1,00	0,85
2.1	1,00	0,50	0,50	1,00	0,63	1,00	0,50	0,50	1,00	0,63	0,00	1,00	1,00	0,00	0,75
2.2	1,00	0,00	0,50	1,00	0,40	1,00	0,50	0,50	1,00	0,63	1,00	1,00	1,00	1,00	1,00
2.3	0,00	0,50	0,00	1,00	0,33	1,00	1,00	0,50	1,00	0,85	1,00	1,00	1,00	1,00	1,00
2.4	0,50	1,00	0,00	1,00	0,63	1,00	1,00	0,50	1,00	0,85	1,00	1,00	0,50	1,00	0,85
2.5	1,00	1,00	0,00	1,00	0,70	1,00	1,00	0,00	0,50	0,65	0,50	1,00	1,00	0,50	0,88
3.1	0,50	1,00	0,00	1,00	0,63	1,00	0,00	1,00	1,00	0,55	0,50	1,00	1,00	1,00	0,93
3.2	0,50	0,00	0,00	1,00	0,18	1,00	0,50	1,00	1,00	0,78	1,00	1,00	1,00	1,00	1,00
3.3	0,50	1,00	0,00	1,00	0,63	1,00	0,00	0,50	1,00	0,40	0,50	1,00	1,00	1,00	0,93
3.4	0,50	1,00	0,50	1,00	0,78	0,50	0,00	1,00	1,00	0,48	0,50	1,00	1,00	1,00	0,93
3.5	0,50	0,50	0,50	1,00	0,55	1,00	0,00	0,00	0,00	0,15	0,50	1,00	1,00	1,00	0,93
4.1	1,00	0,00	0,00	0,00	0,15	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
4.2	1,00	0,00	0,50	0,50	0,35	1,00	0,50	1,00	1,00	0,78	1,00	1,00	1,00	1,00	1,00
4.3	1,00	0,00	1,00	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
4.4	1,00	0,50	1,00	0,50	0,73	1,00	1,00	1,00	0,50	0,95	1,00	0,50	0,50	1,00	0,63
4.5	1,00	0,50	1,00	0,50	0,73	1,00	1,00	0,50	1,00	0,85	0,50	1,00	1,00	1,00	0,93
5.1	1,00	1,00	0,50	1,00	0,85	1,00	0,00	0,50	1,00	0,40	1,00	1,00	1,00	1,00	1,00

5.2	0,50	0,00	0,00	0,50	0,13	1,00	0,00	0,00	1,00	0,25	1,00	1,00	0,50	1,00	0,85
5.3	0,50	1,00	0,00	0,50	0,58	1,00	0,50	1,00	1,00	0,78	1,00	1,00	0,00	1,00	0,70
5.4	1,00	0,50	0,00	1,00	0,48	1,00	0,00	1,00	1,00	0,55	1,00	1,00	0,50	0,50	0,80
5.5	0,50	0,50	0,50	1,00	0,55	0,00	1,00	0,50	1,00	0,70	1,00	1,00	0,50	0,50	0,80
6.1	0,50	0,00	0,00	0,50	0,13	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
6.2	0,50	0,50	0,50	1,00	0,55	0,50	0,50	0,50	1,00	0,55	1,00	0,50	1,00	1,00	0,78
6.3	1,00	1,00	0,50	0,50	0,80	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,00	1,00	0,70
6.4	0,50	0,00	0,00	0,50	0,13	1,00	1,00	0,50	1,00	0,85	1,00	1,00	1,00	1,00	1,00
6.5	1,00	1,00	0,50	0,50	0,80	1,00	1,00	0,50	1,00	0,85	1,00	1,00	0,50	1,00	0,85
7.1	0,50	0,00	0,50	0,40	0,27	1,00	1,00	0,00	1,00	0,70	1,00	1,00	0,50	1,00	0,85
7.2	0,50	0,00	0,00	0,50	0,13	1,00	1,00	1,00	0,50	0,95	1,00	0,50	1,00	1,00	0,78
7.3	1,00	0,50	0,50	0,50	0,58	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,50	1,00	0,85
7.4	1,00	0,50	0,00	0,50	0,43	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
7.5	0,50	0,50	0,00	0,00	0,30	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
8.1	0,50	0,50	0,00	0,50	0,35	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,50	1,00	0,85
8.2	1,00	0,50	0,00	0,00	0,38	0,00	1,00	1,00	1,00	0,85	1,00	1,00	1,00	1,00	1,00
8.3	0,50	0,00	0,00	0,50	0,13	1,00	1,00	0,50	1,00	0,85	1,00	1,00	1,00	0,50	0,95
8.4	0,50	0,00	0,00	0,50	0,13	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
8.5	1,00	0,00	0,00	0,00	0,15	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
9.1	0,00	0,00	0,00	0,00	0,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
9.2	0,00	0,50	0,00	0,00	0,23	0,50	0,50	1,00	1,00	0,70	1,00	1,00	1,00	1,00	1,00
9.3	0,00	0,00	0,50	0,50	0,20	0,50	1,00	1,00	0,50	0,88	1,00	1,00	1,00	1,00	1,00
9.4	0,00	0,00	0,00	0,50	0,05	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
9.5	0,50	0,00	0,00	0,00	0,08	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
10.1	0,00	0,50	0,50	0,00	0,38	0,00	0,50	1,00	1,00	0,63	0,50	1,00	1,00	1,00	0,93
10.2	0,00	0,50	0,50	0,00	0,38	0,50	0,50	1,00	1,00	0,70	0,50	1,00	0,50	1,00	0,78
10.3	0,50	0,50	0,00	0,00	0,30	0,50	0,50	1,00	1,00	0,70	1,00	1,00	1,00	1,00	1,00
10.4	0,50	0,00	0,00	0,00	0,08	0,50	0,50	1,00	0,00	0,60	0,50	1,00	0,00	1,00	0,63
10.5	0,50	0,00	0,50	0,00	0,23	0,50	1,00	1,00	1,00	0,93	0,50	1,00	1,00	1,00	0,93
AVG	0,61	0,38	0,24	0,55	0,39	0,84	0,74	0,75	0,92	0,78	0,85	0,96	0,83	0,94	0,90