НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

Головний редактор

Сініцин Ігор Петрович

чл.-корр. НАН України, д.т.н., професор,
директор Інституту програмних систем
НАН України

✉ Інститут програмних систем НАН України
проспект Академіка Глушкова, 40, корп. 5
03187, Київ-187

☎ Тел.+380 (44) 526 5507

E-mail: ips2014@ukr.net
http://www.pp.isofts.kiev.ua

Журнал «Проблеми програмування» за **категорією «Б»** включений до переліку наукових фахових видань України, в яких можуть публікуватися результати дисертаційних робіт на здобуття наукових ступенів доктора наук, кандидата наук та ступеня доктора філософії **за кластером «Інформаційні технології та електроніка», галузі науки F2, F3, F4, F5, F6** (Наказ МОН України від 19.01.2026 №56).

Журнал має Свідоцтво про державну реєстрацію друкованого засобу масової інформації, серія КВ, № 7490, а також Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції, серія ДК, № 5778.

Всі матеріали, прийняті до публікації в журналі, підлягають обов'язковому зовнішньому і внутрішньому рецензуванню.

Затверджено до друку вченою радою Інституту програмних систем НАН України.
Протокол №6 від 11.06.2026.

Редактор *З.В. Єгорова*
Комп'ютерна верстка *С. Кравченко*

Підписано до друку 05.06.2026. Дата друку (розміщення онлайн) 29.06.2026. Формат 60x84/8. Папір офс.
Ум. друк. арк. 14,90. Обл.-вид. арк.13,80. Тираж 120 прим. Ціна договірна. Замовлення 108.

НАЦІОНАЛЬНА
АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 2 квітень - червень 2026

Заснований у березні 1999 р.

ЗМІСТ

Свідоцтво про державну реєстрацію КВ № 7490 від 01.07.2003

Науковий журнал "Проблеми програмування" занесений до переліку наукових фахових видань України, в яких можуть публікуватися основні результати дисертаційних робіт.

ISSN 1727-4907

NATIONAL ACADEMY
OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS IN PROGRAMMING

scientific journal

№ 2 ***April – June*** ***2026***

Founded in March, 1999

CONTENT

УДК 004.4:004.8 https://doi.org/10.15407/pp2026.02.004

О.В. Нестеренко, В.В. Федоров, П.П. Яцук

AI-ЛАНДШАФТ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Штучний інтелект стає невід'ємною частиною сучасної програмної інженерії, розширюючи можливості автоматизації, оптимізації й підтримки технічних рішень протягом усього життєвого циклу програмного забезпечення. У статті запропоновано методологію визначення інструментарію штучного інтелекту для вирішення задач програмної інженерії. Базою цієї методології є системна структуризація на основі галузей знань (Knowledge Area) нової версії міжнародного посібника SWEBOK V4 та сукупності певних типів процесів або діяльності (Topics), з яких складаються галузі знань. Для структуризації вибрано головні області (шість областей для розроблення ПЗ) та допоміжні організаційні сфери (шість областей, що забезпечують інженерію та керування розробкою ПЗ). Розглянуто застосування в програмній інженерії таких моделей і алгоритмів штучного інтелекту як машинне навчання (ML), глибинне навчання (DL), великі мовні моделі (LLM), обробка природної мови (NLP), генеративний ШІ (Generative AI), графові алгоритми, метаевристики та оптимізаційні алгоритми, експерті системи та аналітика процесів. У результаті побудовано загальну онтологію AI-ландшафту програмної інженерії. Детально розглянуто карту застосування ШІ в програмній інженерії, яка відображає відповідність між галузями знань SWEBOK, процесами, ШІ-технологіями та інструментами. Запропонована картографія демонструє три важливі принципи інтеграції AI в програмну інженерію: технологічний рівень; процесний та інструментальний рівні. Такий підхід дозволяє систематизувати використання ШІ, уникнути фрагментарного впровадження технологій, а також оцінювати рівень інтелектуалізації процесів розробки. Окремо в статті розглянуто модель зрілості для впровадження штучного інтелекту в галузі програмної інженерії (AI-Driven Software Engineering CMM), яка створена для допомоги підприємствам та організаціям оцінити й вдосконалити свої зусилля з інтеграції ШІ та програмної інженерії. Результати дослідження можуть бути використані для перетворення цього симбіозу з ідеї на робочу парадигму, зокрема, з урахуванням надійності, пояснюваності та етичних аспектів використання ШІ.

Ключові слова: штучний інтелект, SWEBOK, моделі і алгоритми, модель зрілості, програмне забезпечення, людино-машинна взаємодія.

O.V. Nesterenko, V.V. Fedorov, P. P. Yatsuk

AI LANDSCAPE OF SOFTWARE ENGINEERING

Artificial intelligence is becoming an integral part of modern software engineering, expanding the possibilities of automation, optimization and support of technical solutions throughout the software life cycle. The article proposes a methodology for defining artificial intelligence tools for solving software engineering problems. The basis of this methodology is a system structuring based on the knowledge areas of the new version of the international manual SWEBOK V4 and a set of certain types of processes or activities (Topics), which consist of knowledge areas. The main areas (six areas for software development) and auxiliary organizational areas (six areas that provide engineering and management of software development) were selected for structuring. The application of such artificial intelligence models and algorithms in software engineering as machine learning (ML), deep learning (DL), large language models (LLM), natural language processing (NLP), Generative AI, graph algorithms, metaheuristics and optimization algorithms, expert systems and process analytics is considered. As a result, a general ontology of the AI landscape of software engineering is built. A map of the application of AI in software engineering is considered in detail, which reflects the correspondence between SWEBOK knowledge areas, processes, AI technologies and tools. The proposed cartography demonstrates three important principles of integrating AI into software engineering: technological level; process level and instrumental level. This approach allows you to systematize the use of AI, avoid fragmented implementation of technologies, and also assess the level of intellectualization of development processes. Separately, the article discusses the AI-Driven Software Engineering Maturity Model, which was created to help enterprises and organizations assess and improve their efforts to integrate AI and SE. The results of the study can be used to transform this symbiosis from an idea into a working paradigm, in particular, taking into account the reliability, explainability and ethical aspects of using AI.

Keywords: artificial intelligence, SWEBOK, models and algorithms, maturity model, software, human-machine interaction.

Вступ

Використання у різних сферах діяльності засобів на основі технологій штучного інтелекту (ШІ) стрімко зростає. Не оминули нові підходи й вирішення задач програмної інженерії (ПІ). ШІ стає невід'ємною частиною сучасної програмної інженерії, розширюючи можливості автоматизації, оптимізації й підтримки технічних рішень протягом усього життєвого циклу програмного забезпечення (ПЗ). Застосування таких технологій, як машинне навчання (ML), глибинне навчання (DL), великі мовні моделі (LLM) та обробка природної мови (NLP) вже змінило традиційні підходи до розробки ПЗ. Результатом стало визначення перспективних напрямків застосування ШІ для покращення якості програмного забезпечення як от управління життєвим циклом розробки ПЗ, прогнозування помилок/дефектів та оцінки зусиль, інтелектуалізація програмної інженерії в управлінні кодом, обробка природної мови в інженерії вимог, а також пошук даних з репозиторіїв програмного забезпечення [1]. Найпоширеніше використання ШІ охоплює такі фази ПІ, як вимоги, проєктування, розробка, тестування, випуск та обслуговування [2].

Застосування технологій штучного інтелекту для вирішення проблем програмної інженерії не є виключно новою тенденцією, а радше результатом багаторічних зусиль спільноти програмної інженерії з розробки автоматизованих та інтелектуальних інструментів та фреймворків для забезпечення ефективного досвіду розробки ПЗ [3]. У програмній інженерії відбувається зміна парадигм, і саме системи штучного інтелекту відіграють дедалі важливішу роль у підвищенні продуктивності цих процесів. Основне питання, яке повинні вирішувати як дослідницька спільнота ШІ, так і програмної інженерії полягає у тому, як синергетично інтегрувати новий «обчислювальний інтелект» та людський інтелект [4].

Варто зазначити, що через свою «чорну скриньку» ці перспективні моделі та технології на основі ШІ все ще далекі від практичного впровадження в програмній інженерії. Зокрема, ШІ викликає багато занепокоєнь щодо його відповідального використання. Результати багатьох досліджень свідчать не лише про зростаючу інтеграцію ШІ в різні галузі людської діяльності, а й вказують, у першу чергу, на збільшення уваги до етичних та правових аспектів використання ШІ [5]. Відсутність пояснень у рішеннях, що пропонує ШІ, створює небажані ризики для їхнього застосування в критично важливих завданнях ПІ, таких, наприклад, як виявлення вразливостей, де прозорість ухвалення рішень має першорядне значення [6]. У зв'язку із цим існує потреба зрозуміти та розробити принципи того, що передбачає відповідальна розробка ПЗ на практиці, а також ретельно оцінити результативність впровадження інструментів та методів ШІ в програмну інженерію [7].

Хоча чимало досліджень підкреслюють вагому роль автоматизації на базі ШІ у покращенні ухвалення рішень, оптимізації робочих процесів та скороченні часу розробки, однак звертається увага на те, шо саме традиційні методології все ще забезпечують структуру, відповідність та надійність. Тому вважається, що майбутнє за гібридною моделлю, яка поєднує традиційні підходи з автоматизацією на основі ШІ [8].

Отже, епоха Software Engineering 2.0, пов'язана з розвитком мікроелектроніки, поширенням Інтернету та впровадженням методологій командної роботи відходить у минуле. Інтеграція ШІ в програмну інженерію відкриває перспективи нової ери Software Engineering 3.0, що має величезний потенціал для підвищення ефективності та якості ПЗ. Технології ШІ несуть переосмислення традиційних практик розробки програмного забезпечення, революціонізують програмну інженерію, впроваджуючи інтелектуальну автоматизацію у життєвий цикл ПЗ [9].

Хоча за останні роки відбувся значний прогрес у перетині програмної інженерії та штучного інтелекту, щоб перетворити цей симбіоз з ідеї на робочу парадигму необхідні значні дослідницькі зусилля, зокрема щодо надійності, пояснюваності та етичних аспектів використання ШІ.

Методика визначення AI-інструментарію

Розглядаючи питання визначення AI-інструментарію для застосування в програмній інженерії, доцільно виходити з того, що ПІ як інженерна дисципліна – це сукупність прийомів виконання діяльності, пов'язаної з виготовленням програмних продуктів і систем (ППС) для різних видів цільових об'єктів із застосуванням різних методів, засобів та інструментів як складових програмної інженерії. З інженерної точки зору в ПІ розв'язуються задачі розробки ПЗ та виготовлення ППС, подані як технологічні процеси формування вимог, проєктування і супроводу продукту, а також перевірки операцій базового процесу щодо правильності виконання різних функціональних задач та виконання робіт за проєктом.

Основні необхідні компетентності в професійній практиці програмної інженерії як сфері діяльності з розробки ПЗ базуються на міжнародних рекомендацій-них документах з програмної інженерії. Основним з них є ядро знань ПІ – посіб-ник SWEBOK (Software Engineering Body of Knowledge). Це набір теоретичних концепцій і формальних визначень методів і засобів розробки та керування програмними проектами, які можуть застосовуватися в інженерії програмування. Починаючи з пробної версії, опублікованої 2001 року, версії 2004 року та широкого використання версії 2014 року (SWEBOK V3) цей посібник в межах спільноти розробників програмного забезпечення відіграє важливу роль як флагманський основоположний та структурний документ для створення програмних продуктів.

Метою розробки оновлення цього посібника 2025 року (SWEBOK V4) є покращення його актуальності, читабельності, узгодженості та зручності використання, формування єдиної концептуальної основи для організації, управління та вдосконалення процесів створення ПЗ. Зміст нової версії цього посібника зріс уже до 18 галузей знань (Knowledge Area), кожна з яких відповідає сукупності певних типів процесів або діяльності (Topics).

Понятійний апарат галузей знань (областей) SWEBOK можна умовно поділити на головні (шість областей для розроблення ПЗ), допоміжні організаційні області (шість областей, що забезпечують інженерію та керування розробкою ПЗ) та базові галузі знань.

Головні галузі – це Вимоги до ПЗ (Software Requirements), Архітектура ПЗ (Software Architecture), Проєктування ПЗ (Software Design), Конструювання ПЗ (Software Construction), Тестування ПЗ (Software Testing) та Супровід ПЗ (Software Maintenance). Допоміжні – це Операції програмної інженерї (Software Engineering Operations), Керування конфігурацією ПЗ (Software Configuration Management), Процес програмної інженерії (Software Engineering Process), Моделі та методи інженерії ПЗ (Software Engineering Models and Methods), Управління програмною інженерією (Software Engineering Management) та Якість ПЗ (Software Quality).

Враховуючи, що SWEBOK відіграє ключову роль у формуванні системного підходу до виробництва програмних продуктів, забезпечуючи структуровану модель знань, яка підтримує організацію, управління та вдосконалення процесів ПІ, вважається за доцільне спиратися на запропоновану в SWEBOK структуризацію для визначення інструментарію штучного інтелекту, що знаходить застосування в програмній інженерії. Такий підхід дозволяє пов'язати AI-технології з конкретними сферами діяльності в ПІ, системно визначити, де саме і які AI-інструменти можуть бути використані.

Таким чином методика визначення AI-інструментарію полягає у реалізації кількох кроків:

а) ідентифікація галузі знань за SWEBOK,

б) аналіз ключових процесів на основі топіків,

в) визначення задач автоматизації за допомогою ШІ,

г) відбір відповідних AI-технологій (моделей і алгоритмів),

д) визначення конкретних інструментів.

Використання SWEBOK як методологічної основи визначення AI-інструментарію має низку переваг, як от забезпечення повноти аналізу завдяки охопленню всіх аспектів програмної інженерії, системність через чітку класифікацію застосувань AI, порівнюваність через можливість оцінювання різних інструментів, нарешті, узгодженість із міжнародними стандартами.

AI моделі і алгоритми

Серед моделей штучного інтелекту основним вважається машинне навчання (Machine Learning – ML), що вже тривалий час розвивається й успішно застосовується в різних галузях. В останні роки використання методів машинного навчання стало одним з основних і для вирішення багатьох завдань програмної інженерії (ML4SE). Типовими серед них є прогнозування дефектів, аналіз якості коду, оцінювання ризиків проєкту, оптимізація процесів розробки. Цього було досягнуто завдяки використанню найсучасніших моделей ML, які, як правило, є складнішими та нестандартними. Однак їх використання призводить до отримання менш з'ясованих рішень, що знижує довіру професіоналів галузі до використання рішень ML4SE. Одним з потенційних напрямків для зниження негативу відсутньої пояснивості є пропонування з'ясованих методів штучного інтелекту (explainable AI – XAI). На цей підхід (XAI4SE) на сьогодні значною мірою звертається увага у спільноті програмної інженерії [10].

Подальший розвиток машинного навчання, пов'язаний із навчанням з підкріпленням (Reinforcement Learning – RL) та глибинним навчанням (Deep Learning – DL), а також глибинним навчанням з підкріпленням (DRL) на сучасному етапі відіграє важливу роль у створенні просунутих засобів ШІ. Ці моделі успішно використовуються й для автоматизації складних завдань програмної інженерії, таких як тестування ігор, вирішення задач планування робіт, а також для навчання ефективної та економічно вигідної поведінки в різних середовищах спеціалізованих програмних агентів [11].

Застосування вищезазначених моделей забезпечило розвиток напряму обробки природної мови (Natural Language Processing – NLP), що відчутно спростило практику ПІ в автоматизації класифікації вимог, аналізі настроїв користувачів та управлінні проєктами, зокрема й щодо супроводу програмного забезпечення [12]. Типовими задачами, що вирішуються на основі NLP є аналіз текстових вимог, генерація документації, пошук невідповідностей у специфікаціях, аналіз технічної документації тощо.

Безумовно, на сьогодні найпопулярнішим є впровадження інструментів штучного інтелекту на основі великих мовних моделей (Large Language Models – LLM). Результати показують, що дослідники ПІ також визнають перевагу внеску LLM, однак лише коли ці моделі застосовується до невеликих, вузькоспеціалізованих та перевірюваних завдань, а не до складних, що не мають запланованого завершення і можуть розвиватися кількома способами (open-ended tasks). LLM використовується як додатковий інструмент до традиційних методів програмної інженерії щодо таких завдань з високими ставками, які потребують справжніх людських зусиль та емоційної відданості [13].

Також необхідно зазначити, що для моделей ШІ, в тому числі і LLM, головною потребою є дані. Ефективність цих моделей залежить від максимізації джерел високоякісних даних. Однак дані, особливо високої якості, часто мають комерційну або конфіденційну цінність, що робить їх менш доступними для застосування засобів ШІ в програмних проєктах, особливо в проєктах створення ПЗ з відкритим кодом. Ця реальність створює значну перешкоду для розробки та впровадження інструментів на основі ШІ в спільноті ПІ [14].

Серйозними задачами програмної інженерії є перевірки коду. У цьому за своєю суттю людиноцентричному процесі важливо зрозуміти, як інженерам-програмістам орієнтуватися у впровадженні ШІ у спільні з ним робочі процеси, адже діяльність із перевірки коду є багатовимірною, охоплюючи когнітивні, емоційні та поведінкові виміри. Впровадження

перевірок за допомогою LLM впливає на деякі з цих атрибутів. Наприклад, у роботі з LLM потреба в емоційній регуляції та механізмах подолання стає меншою, однак когнітивне навантаження іноді вище, зокрема, під час роботи зі зворотним зв'язком, згенерованим LLM, через його надмірну деталізацію. Водночас сприйняття зворотного зв'язку від LLM обмежене недовірою та відсутністю пояснювального контексту в перевірці [15].

Перспективним напрямком є застосування в програмній інженерії багатоагентних автономних систем (Multi-agent autonomous systems – MAS), які краще справляються із завданнями, що охоплюють кілька сфер, ніж окремі автономні агенти. Сучасні дослідження MAS в ПІ зосереджені на інтеграції LLM в ядро автономних агентів для створення багатоагентних систем. Однак впровадження таких систем створює безліч проблем, серед яких однією з основних є стратегічний розподіл завдань між людьми та MAS надійним чином [16].

Отже, LLM розширюють дослідницькі можливості в галузі ПІ завдяки прискореному генеруванню ідей та автоматизованим процесам, роблячи деякі традиційні практики застарілими. Однак людиноцентрична перспектива залишається важливою. Забезпечення людського нагляду та інтерпретації необхідне для підтримки наукової точності, сприяння етичній відповідальності та стимулювання прогресу в цій галузі [17].

В програмній інженерії як суттєвий чинник розглядається вплив поточного стану команди розробників на процеси розробки програмного забезпечення. Визначається, що управління проєктами має базуватися не лише на його параметрах та моделі предметної області як основи бази знань для підтримки управлінських рішень, а й на особистісних характеристиках програмістів відповідно до методології групової динаміки та комунікацій [18]. У нових умовах вже прослідкується командна робота людина-машина (Human-machine teaming - HMT), тобто взаємодія людей і машини як членів команди. Цей підхід може бути корисним у ШІ4SE, але питання впливу HMT на ефективність команди отримали ще мало уваги у спільноті програмної інженерії [19].

Емерджентні властивості LLM привносять новизну та креативність у застосування в усьому спектрі діяльності програмної інженерії, включаючи кодування, проєктування, вимоги, виправлення, рефакторинг, покращення продуктивності, документацію та аналітику. Однак ці ж емерджентні властивості створюють і значні технічні проблеми. Потрібні методи, які можуть надійно виявляти неправильні рішення, притаманні LLM, такі як галюцинації. Тому ключову роль у розробці та впровадженні надійних, ефективних та результативних методів ПІ на основі LLM мають відігравати гібридні методи (традиційна ПІ плюс LLM) [20].

Досягнення в сфері LLM привели до широкого поширення генеративного ШІ (Generative AI), або чат-ботів великих мовних моделей. В галузі програмної інженерії використання генеративного ШІ також може бути корисним інструментом, адже за його допомогою можливо підтримувати всі фази розробки програмного забезпечення, вирішуючи такі типові задачі як аналіз вимог, генерація програмного коду, створення тестів, автоматичне створення прототипів, генерація API та документації [21]. Завдяки використанню генеративних моделей, таких як попередньо натреновані трансформери (GPT), система здатна розуміти складні запити користувачів, підтримувати контекстну зв’язність у розмовах і автономно виконувати завдання з мінімальним втручанням людини [22]. Однак, можуть виникнути й різні проблеми, зокрема щодо правдивості відповідей, наприклад, згенерованих ChatGPT. Це потрібує додатковиї засобів для виявлення некоректностей у відповідях [23].

Оцінюючи інструменти генеративного ШІ на основі таких критеріїв, як зручність використання, ефективність та інтеграція в існуючі робочі процеси, дослідження підкреслюють важливість співпраці людини та ШІ. Припускається, що хоча генеративний ШІ може суттєво підтримувати завдання розробки програмного забезпечення, залишаються важливими люд-

ський нагляд та критична оцінка результатів, створених ШІ [24].

Для ефективного розв'язання реальних задач програмування перспективними методами є застосування за допомогою штучного інтелекту графових алгоритмів та структур даних. Типовими задачами, що вирішуються, є аналіз структури програмних систем, дослідження залежностей між компонентами, аналіз конфігурацій, impact-analysis змін. У конструюванні ефективних і оптимізованих рішень для конкретної проблеми ПІ важливим є акцент на практичну імплементацію алгоритмів, що особливо актуально для систем із великими обсягами даних (Big Data, фінансові або медичні системи).

Також ефективність у вдосконаленні таких завдань ПІ, як управління проєктами, прогнозування розташування функцій та дій з модифікації програмного забезпечення демонструють метаевристики та оптимізаційні алгоритми, підвищуючи точність та ефективність у цих завданнях [12]. Також типовими задачами можуть бути оптимізація архітектур, оптимізація тестових наборів, оптимізація планування розробки.

Не залишаються поза увагою й застосування ШІ в таких інструментах, як Process Mining, експерті системи та аналітика процесів. Типові задачі – аналіз процесів розробки ПЗ, оцінювання зрілості процесів, вдосконалення SDLC, управління змінами, зокрема в таких SWEBOK-областях як базовий процес програмної інженерії, управління проєктами, управління конфігурацією тощо.

Експертні методи, які певною мірою дають змогу вирішувати поставлені задачі, доцільні для вирішення багатокритеріальних задач, що зазвичай викликають значні когнітивні навантаження. Однією з таких, зокрема, є задача ранжування вимог. Вважається, що для ухвалення рішень у такому середовищі в сучасних умовах цифровізації перспективним є забезпечення підтримки рішень на основі вичерпного подання інформаційної моделі предметної області та опрацювання за допомогою інформаційних технологій кількісних оцінок альтернатив. Також важливою можливістю сучасних технологій є забезпечення візуалізації процесів, пов'язаних з ухваленням рішень [25]. Ці задачі якнайкраще розв'язуються із застосуванням засобів ШІ.

Отже, штучний інтелект більше не є футуристичною концепцією. Сьогодні це множина щоденних інструментів у наборі засобів сучасних розробників ПЗ. Починаючи від GitHub Copilot, OpenAI Codex до автоматизованих фреймворків ці інструменти призначені для підвищення продуктивності розробника, гарантування найвищої якості програмного забезпечення та скорочення часу виведення продукту на ринок. Тематичні дослідження відомих технологічних компаній демонструють суттєві покращення ефективності та надійності в результаті використання інструментів на базі ШІ, що підвищує впевненість у розробці інтелектуальних, масштабованих та адаптивних програмних систем [26].

Таким чином, загальну онтологію AI-ландшафту програмної інженерії можна представити рис. 1.

Картографія застосування AI в програмній інженерії

SWEBOK організовує знання програмної інженерії у вигляді галузей знань (Knowledge Areas), кожна з яких відповідає певному типу процесів або діяльності. Спираючись на цю структуру, можна застосувати картографічний підхід.

На основі структури SWEBOK можна сформувати карту застосування AI в програмній інженерії, яка відображає відповідність між галузями знань, процесами, AI-технологіями, інструментами. Такий підхід дозволяє систематизувати використання AI, уникнути фрагментарного впровадження технологій, а також оцінювати рівень інтелектуалізації процесів розробки.

Системно карту застосування штучного інтелекту в програмній інженерії доцільно будувати на основі структури галузей знань, визначених у SWEBOK Guide як багаторівневу модель, що складається з таких рівнів як сфери діяльності, типи задач, технології AI та практичні інструменти (табл. 1).

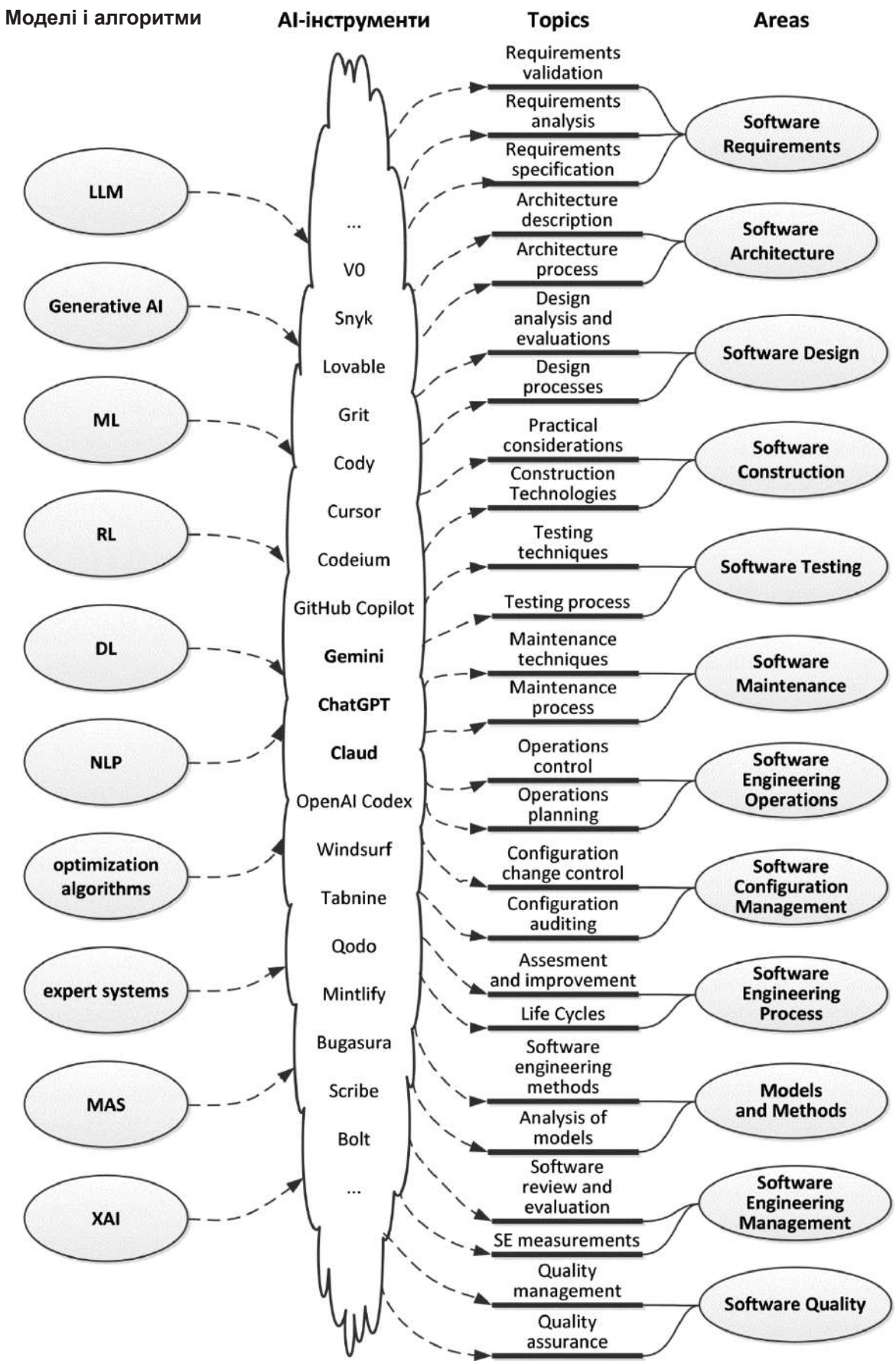

Рис. 1. Онтологія AI-ландшафту програмної інженерії

Таблиця 1.

Карта застосування AI в програмній інженерії

№ з/п	Область знань програмної інженерії	Основні задачі	AI-технології	Приклади інструментів
1.	Інженерія вимог	Аналіз вимог, виявлення неоднозначностей, трасування	NLP, класифікація тексту, knowledge graphs	ChatGPT-подібні системи, вимогові аналізатори
2.	Архітектура і проєктування ПЗ	Архітектурні рішення, моделювання, вибір патернів	Рекомендаційні системи, генеративні моделі	AI-архітектурні помічники
3.	Конструювання ПЗ	Програмування, рефакторинг, генерація коду	Великі мовні моделі, deep learning	GitHub Copilot
4.	Тестування ПЗ	Генерація тестів, виявлення дефектів	ML-класифікація, генеративні моделі	SonarQube
5.	Супровід ПЗ	Локалізація дефектів, модернізація систем	Defect prediction, code analysis	AI bug prediction systems
6.	Управління конфігурацією	контроль версій, merge-конфлікти	аналіз репозиторіїв, графові моделі	Git
7.	Базовий процес ПІ	оцінювання та вдосконалення процесів	process mining, ML-аналітика	AI-process analytics
8.	Методи інженерії	евристичні, формальні, прототипні методи	метаевристики, symbolic AI	AI-оптимізаційні системи
9.	Управління програмною інженерією та проєктами	прогнозування строків, ризик-аналіз	predictive analytics, ML-прогнозування	Jira

У цій карті AI виконує чотири ключові функції: асистент розробника (допомагає створювати код, моделі та документацію), аналітик системи (аналізує дані про проєкт і кодову базу), прогностичний модуль (прогнозує дефекти, ризики та строки), а також оркестратор процесів (інтегрує інструменти DevOps і оптимізує workflow). Запропонована карта демонструє три важливі принципи інтеграції AI в програмну інженерію: технологічний рівень – типи AI-алгоритмів; процесний рівень – галузі знань програмної інженерії; інструментальний рівень – конкретні програмні засоби. Це дозволяє формувати цілісну модель AI-орієнтованої програмної інженерії (AI-Driven Software Engineering).

Модель зрілості AI-Driven Softare Engineering

Парадигма програмної інженерії запровадила мислення на основі моделі зрілості, яка надає компаніям дорожню карту для покращення їхньої продуктивності з обраних точок зору, відомих як ключові можливості. Capability Maturity Model Integration (CMMI) – це комплексна модель продуктивності та зрілості, що представляє набір моделей (методологій) вдосконалення процесів в організаціях різних розмірів і видів діяльності. CMMI містить набір рекомендацій у вигляді практик, реалізація яких дозволяє досягти цілей, необхідних для повної реалізації сфер діяльності.

На основі цієї методології Інститутом програмної інженерії (Software Engineering Institute – SEI) запропоновано використовувати модель зрілості можливостей людей (People CMM). Це інструмент, який допомагає успішно вирішувати критичні проблеми з людьми в організації, яка, зокрема, працює над розробкою ПЗ. P-CMM використовує систему процесів надзвичайно успішної моделі зрілості можливостей для програмного забезпечення як основу для моделі найкращих практик для управління та розвитку робочої сили організації. Базовий підхід моделі P-CMM – це усвідомлення цінності кожного працівника як особистості та необхідності його подальшого розвитку.

На основі цих концепцій дослідники і фахівці пропонують модель зрілості для впровадження штучного інтелекту в галузі програмної інженерії (AI-Driven Software Engineering CMM), щоб допомогти організаціям оцінити та вдосконалити свої зусилля з інтеграції ШІ та ПІ. Ґрунтуючись на екосистемі багатьох зацікавлених сторін, що включає академічні кола, дослідників, гравців галузі та урядові установи, ця модель визначає ключові етапи впровадження ШІ та умови, необхідні для прогресу в цій сфері.

Переосмислюючи розрив між інноваціями та практикою через цю структуровану призму, модель пропонує практичні висновки для узгодження результатів досліджень з промисловою готовністю та прискорення ефективного впровадження штучного інтелекту в середовищах програмної інженерії [27]. Модель об'єднує наукові та практичні висновки і створює рекомендації для розробників програмного забезпечення та описує структуру цієї моделі зрілості для оцінки та покращення використання розробки із застосуванням ШІ, зокрема LLM [28].

Модель зрілості AI-DSE CMM передбачає п'ять рівнів зрілості (табл. 2). Таким чином ця модель не лише описує ступінь інтеграції технологій штучного інтелекту в процеси розробки програмного забезпечення, а й. дозволяє оцінити, наскільки організація використовує ШІ для підтримки життєвого циклу ПЗ, та визначити напрямки подальшого розвитку. Концептуально модель спирається на структуру галузей знань, визначених у SWEBOK Guide, а також на підходи до оцінювання зрілості процесів, подібні до CMMI.

Узагальнено структуру моделі зрілості AI-Driven Software Engineering можна представити рис. 2.

Оцінювання зрілості AI-Driven Software Engineering може проводитися за такими критеріями та параметрами:

1) інструментальна інтеграція – рівень використання AI-інструментів;

2) процесна інтеграція – ступінь використання AI у SDLC;

3) дані та аналітика – доступність історичних даних проєктів;

4) автоматизація – рівень автоматизації процесів розробки;

5) організаційна готовність – компетентності персоналу, наявність стратегії впровадження AI.

Практичне значення використання моделі зрілості AI-Driven Software Engineering полягає у тому, що вона дозволяє оцінювати рівень інтелектуалізації процесів розробки, планувати стратегію впровадження AI, порівнювати різні організації виробництва, а також визначати напрямки ефективного розвитку програмної інженерії.

Висновки

Для визначення та класифікації інструментів штучного інтелекту в програмній інженерії може використовуватися як концептуальна рамка структуризація знань, запропонована у SWEBOK. Вона дозволяє побудувати системну модель інтеграції AI в усі сфери діяльності, що формують життєвий цикл програмного забезпечення.

Проведено огляд можливостей і обмежень застосування основних моделей штучного інтелекту, таких як машинне навчання (ML), навчання з підкріпленням (RL) та глибинне навчання (DL), обробка природної мови (NLP), великі мовні моделі (LLM), генеративний ШІ (Generative AI) та ін.

На цій основі визначено картографію застосування AI в програмній інженерії та побудовано загальну онтологію AI-ландшафту програмної інженерії. Запропонована картографія демонструє три важливі принципи інтеграції AI в програмну інженерію: технологічний рівень, процесний рівень та інструментальний рівень. Це дозволяє сформувати цілісну модель AI-орієнтованої програмної інженерії.

Розглянуто структуру моделі зрілості AI-Driven Softare Engineering та її практичні переваги застосування, що дозволяють оцінити, наскільки організація використовує ШІ для підтримки життєвого циклу ПЗ.

Таблиця 2.

Рівні моделі зрілості AI-Driven Software Engineering

Рівень	Характеристика рівня	Типові приклади	Результат
1. Початковий (Ad-hoc AI Use)	AI застосовується епізодично. Інструменти використовуються окремими розробниками.. Відсутня загальна стратегія використання AI	Використання AI-асистентів програмування, окремі експерименти з генерацією коду, застосування AI для аналізу окремих задач	AI використовується як індивідуальний інструмент, але не інтегрований у процес розробки ПЗ
2. Інструментальна інтеграція (AI-Assisted Development)	AI інтегрується в інструментальне середовище розробки. З'являється підтримка окремих процесів SDLC. Використовуються AI-DevOps інструменти.	Автоматична генерація коду, AI-аналіз якості коду, генерація тестів, інтелектуальний аналіз вимог	AI стає частиною інструментальної екосистеми розробки ПЗ
3. Процесна інтеграція (AI-Enabled Processes)	AI використовується в ключових процесах життєвого циклу ПЗ. Дані проєктів системно аналізуються. Ухвалення рішень підтримується аналітичними моделями	Прогнозування дефектів, аналіз продуктивності, команд, автоматизоване тестування, оптимізація CI/CD процесів.	AI стає невід'ємною частиною процесів програмної інженерії
4. Інтелектуальна організація виробництва (AI-Optimized Engineering)	AI використовується для оптимізації процесів розробки. Впроваджено аналітику процесів і process mining. Використовується прогнозна аналітика в управлінні проєктами	AI-аналіз ризиків проєкту, оптимізація архітектурних рішень, інтелектуальна підтримка управління конфігураціями, автоматичний аналіз технічного боргу	Виробництво ПЗ переходить до data-driven управління програмною інженерією
5. Автономна програмна інженерія (Autonomous AI-Driven Engineering)	AI виконує значну частину інженерних задач. Системи здатні адаптувати процеси розробки. Використовується самонавчальна інфраструктура	Автоматичне створення архітектури систем, генерація та оптимізація коду, автономне тестування та верифікація, self-adaptive software systems	AI виступає активним учасником інженерної діяльності, а не лише інструментом

Рис. 2. Модель зрілості AI-Driven Software Engineering

References

1. P. Kokol, The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature, in: Information, 2024, 15, 354, doi: 10.3390/info15060354
2. H. Aditya Pai, K. R. Sharath, K. Babalad, et all, Integrating AI into Software Engineering: A Critical Review and Future Directions, 2025 International Conference on Intelligent Control, Computing and Communications (IC3 2025), in: Proceedings of the Conference, 2025, pp. 20–25. doi: 10.1109/IC363308.2025.10957449
3. M. J. Hossain Faruk, H. Pournaghshband, H. Shahriar, AI-Oriented Software Engineering (AIOSE): Challenges, Opportunities, and New Directions, in: Lecture Notes in Networks and Systems, 2023, 576 LNNS, pp. 3–19. doi: 10.1007/978-3-031-20322-0_1
4. V. Terragni, A. Vella, P. Roop, K. Blincoe, The Future of AI-Driven Software Engineering, in: ACM Transactions on Software Engineering and Methodology, 2025, 34, 5, art. no. 120. doi: 10.1145/3715003
5. O. Nesterenko, P. Yatsuk, AI (not) against AI, in: Environmental Safety and Natural Resources, 2025, 56, 4, pp. 134–153. doi: 10.32347/2411-4049.2025.4.134-153. [in Ukrainian]
6. S. Cao, X. Sun, R. Widyasari, et all, A Systematic Literature Review on Explainability for ML/DL-based Software Engineering, in: ACM Computing Surveys, 2025, 58, 4, art. no. 95, pp. 1–34, doi: 10.1145/3763230.
7. R. Ulfsnes, N. B. Moe, J. Emmerhoff, et all, Responsible AI in Agile Software Engineering – An Industry Perspective. in: Lecture Notes in Business Information Processing, 2025, 524, pp. 33–41. doi: 10.1007/978-3-031-72781-8_4.
8. Q. U. Ain, M. Haq, A. A. A. Jilani, K. Sohail, A comparison between traditional software engineering practices and AI-driven methodologies, in: Generative AI in Software Engineering, 2025, pp. 57–92. doi: 10.4018/979-8-3373-0370-3.ch002.
9. M. Alenezi, M. Akour, AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions, in: Applied Sciences, 2025, 15, 3, art. no. 1344, doi: 10.3390/app15031344.
10. A. Mohammadkhani, N. Bommi, M. Daboussi, et all, A Systematic Literature Review of Explainable AI for Software Engineering, 2023, arXiv.2302.06065, doi: 10.48550/arXiv.2302.06065.
11. P. S. N. Mindom, A. Nikanjam, F. Khomh, Harnessing pre-trained generalist agents for software engineering tasks, in: Empirical Software Engineering, 2024, 30, 1, art. no. 39. doi: 10.1007/s10664-024-10597-8.
12. U. K. Durrani, M. Akpinar, M. F. Adak, et all, A Decade of Progress: A Systematic Literature Review on the Integration of AI in Software Engineering Phases and Activities (2013-2023), in: IEEE ACCESS, 2024, 12, pp. 171185–171204. doi: 10.1109/ACCESS.2024.3488904.
13. V. T. Wivestad, A. M. Barbala, Attitudes Toward LLM Use Among Software Engineering Researchers: Results from a Two-Phase Survey Study, in: Companion proceedings of the 33RD ACM International conference on the foundations of software engineering (FSE), 2025, pp. 1531–1535. doi: 10.1145/3696630.3731671
14. Z. H. Lin, W. Ma, T. Lin, et all, Open Source AI-based SE Tools: Opportunities and Challenges of Collaborative Software Learning, in: ACM transactions on software engineering and methodology, 2025, 34, 5, pp. 1535–1560. doi: 10.1145/3708529.
15. A. Alami, N. Ernst, Human and Machine: How Software Engineers Perceive and Engage with AI-Assisted Code Reviews Compared to Their Peers, 2025 IEEE/ACM 18th International conference on cooperative and human aspects of software engineering (CHASE), in: Proceedings of the Conference, 2025, pp. 63–74. doi: 10.1109/CHASE66643.2025.00016.
16. K. Ronanki, Facilitating Trustworthy Human-Agent Collaboration in LLM-based Multi-Agent System oriented Software Engineering, in: Companion proceedings of the 33rd ACM International conference on the foundations of software engineering (FSE 2025), arXiv:2505.0425. doi: 10.48550/arXiv.2505.04251.
17. B. Trinkenreich, F. Calefato, G. Hanssen, et all, Get on the Train or be Left on the Station: Using LLMs for Software Engineering Research, in: Companion proceedings of the 33rd ACM international conference on the foundations of software engineering (FSE COMPANION 2025), 2025, pp. 1503–1507. doi: 10.1145/3696630.3731666.
18. O. Nesterenko, Y. Selin, The Teams Information Model for Software Engineering Management, in: Proceedings of 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), 2021, 1, pp. 341–344, doi: 10.1109/CSIT52700.2021.9648737.

19. I. Rauf, H. Sharp, T. Lopez, M. Wermelinger, Human-Machine Teaming and Team Effectiveness in AI tools for Software Engineering, 2025 Conference on Cooperative and Human Aspects of Software Engineering (CHASE), in: Proceedings of the Conference, 2025, pp. 75–80. doi: 10.1109/CHASE66643.2025.00017/
20. A. Fan, B. Gokkaya, M. Harman, et all, Large Language Models for Software Engineering: Survey and Open Problems, 2023 IEEE/ACM International conference on software engineering: future of software engineering (ICSE-FOSE), in: Proceedings of the Conference, 2023, pp. 31–53. doi: 10.1109/ICSE-FoSE59343.2023.00008.
21. A. Striapunin., V. Kharchenko, Using AI tools in requirements engineering: opportunity analysis and chatbot for validation, in: Aerospace Technic and Technology, 2024, 2, pp. 91–101. doi: 10.32620/aktt.2024.2.10. [in Ukrainian]
22. D. Nikitin, V. Golian, Methods for integrating artificial intelligence and knowledge engineering into automaton-based real-time software systems, in: Herald of Khmelnytsskyi National University, 2025, 2, pp. 285–292. doi: 10.31891/2307-5732-2025-349-42.
23. M. H. Tanzil, J. Y. Khan, G. Uddin, ChatGPT Incorrectness Detection in Software Reviews, in: Proceedings of the IEEE/ACM 46th International conference on software engineering (ICSE 2024), arXiv:2403.16347. doi: 10.1145/3597503.3639194.
24. M. Fischer, C. Lanquillon, Evaluation of Generative AI-Assisted Software Design and Engineering: A User-Centered Approach, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2024, 14734 LNAI, pp. 31–47. doi: 10.1007/978-3-031-60606-9_3.
25. O. Nesterenko, O. Trofymchuk, Decision Support for Requirements Prioritization in Software Engineering, in: CEUR Workshop Proceedings, 2024, 3806, 50–61. URL: https://ceur-ws.org/Vol-3806/S_31_Nesterenko_Trofymchuk.pdf.
26. K. Shantha Kumari, B. Sundaravadivazhagan, V. Indumathy, D. Maladhy, Detailing AI techniques and tools for software engineering acceleration and automation, in: Advances in Computers, 2026, 141, pp. 183– 209. doi: 10.1016/bs.adcom.2025.07.007.
27. H. Lhazmir, S. Samhale, K. Louzaoui, K. Benlhachmi, The Uneven Journey of AI in Software Testing: A Maturity Model for Industry Adoption, 8th IEEE International Congress on Information Science and Technology (CIST 2025), in: Proceedings of the Conference, 2025, pp. 132–137. doi: 10.1109/CiSt65886.2025.11224099.
28. S. L. France, Navigating software development in the ChatGPT and GitHub Copilot era, in: Business Horizons, 2024, 67, 5, pp. 649–661. doi: 10.1016/j.bushor.2024.05.009

Дата першого надходження до видання: 27.03.2026
Внутрішня рецензія отримана: 13.04.2026
Зовнішня рецензія отримана: 22.04.2026
Дата прийняття статті до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Нестеренко Олександр Васильович,
доктор технічних наук, професор,
завідувач кафедри
Nesterenko Olexandr,
Ph.D. (doctor, technical sciences),
professor, head of department
http://orcid.org/0000-0001-5329-889X .

Федоров Володимир Володимирович,
кандидат фізико-математичних наук,
доцент
Fedorov Volodymyr,
Ph.D. (physical and mathematical sciences),
associate professor
https://orcid.org/0009-0004-2901-3646.

Яцук Петро Петрович,
кандидат технічних наук, доцент
Yatsuk Petro,
Ph.D. (technical sciences), associate professor
https://orcid.org/ 0009-0002-7124-4849.

Місце роботи авторів:

Міжнародний європейський університет,
Кафедра інформаційних технологій
International European University,
Department of Informational Technologies
Tel.: +380 97 757 27 96
E-mail: oleksandr_nesterenko@ieu.edu.ua
https://business.ieu.edu.ua/kafedry/kafedra-informatsiinykh-tekhnolohii

УДК 004.8+004.9 https://doi.org/10.15407/pp2026.02.016

В.Г. Гуськова, В.І. Школьніков, Б.С. Лисов, А.А. Халигов

ФОРМУВАННЯ РЕКОМЕНДАЦІЙ ДЛЯ ОБ'ЄКТІВ КРИТИЧНОЇ ІНФРАСТРУКТУРИ НА ОСНОВІ ПОТОКОВИХ ДАНИХ, МАШИННОГО НАВЧАННЯ ТА ПІДХОДІВ ШТУЧНОГО ІНТЕЛЕКТУ

У статті розглянуто проблему формування рекомендацій для об'єктів критичної інфраструктури в умовах зростання кіберзагроз, великого обсягу потокових даних та необхідності оперативного ухвалення рішень. Актуальність дослідження зумовлена потребою підвищення стійкості критичної інфраструктури шляхом використання інтелектуальних систем підтримки ухвалення рішень. Метою роботи є розроблення інтегрованого підходу до формування рекомендацій на основі потокових даних, методів машинного навчання, байєсівського ранжування та пояснюваного штучного інтелекту. У дослідженні використано методи виявлення аномалій, класифікації загроз, прогнозування ризиків, rule-based фільтрації та XAI-підходи для пояснення сформованих рекомендацій. Запропонована архітектура забезпечує обробку даних у режимі реального часу, враховує ризики, політики безпеки та контекст функціонування об'єкта. Експериментальна перевірка на наборах даних показала високу ефективність системи: F1 = 0,90, AUROC = 0,96, затримка обробки не перевищує 0,8 с при навантаженні до 10 000 повідомлень за секунду. Встановлено, що механізм адаптивного самооновлення дозволяє зменшити кількість хибних тривог на 43 % та підвищити рівень довіри операторів до рекомендаційної системи. Отримані результати підтверджують перспективність практичного використання запропонованого підходу для підтримки ухвалення рішень на об'єктах критичної інфраструктури.

Ключові слова: об'єкти критичної інфраструктури, рекомендаційна система, потокові дані, машинне навчання, пояснюваний штучний інтелект, оцінювання ризиків, виявлення аномалій, байєсівське ранжування, підтримка ухвалення рішень

V.G. Huskova, V.I. Shkolnikov, B.S. Lysov, A.A. Khalygov

FORMATION OF RECOMMENDATIONS FOR CRITICAL INFRASTRUCTURE OBJECTS BASED ON STREAMING DATA, MACHINE LEARNING, AND ARTIFICIAL INTELLIGENCE APPROACHES

The article addresses the problem of generating recommendations for critical infrastructure objects under conditions of increasing cyber threats, large volumes of streaming data, and the need for rapid decision-making. The relevance of the study is determined by the necessity to enhance the resilience of critical infrastructure through the use of intelligent decision support systems. The purpose of the research is to develop an integrated approach to recommendation generation based on streaming data, machine learning methods, Bayesian ranking, and explainable artificial intelligence. The study employs methods of anomaly detection, threat classification, risk forecasting, rule-based filtering, and XAI approaches for explaining generated recommendations. The proposed architecture provides real-time data processing and takes into account risks, security policies, and the operational context of the infrastructure object. Experimental validation on datasets demonstrated high system efficiency: F1 = 0.90, AUROC = 0.96, while processing latency did not exceed 0.8 s under a load of up to 10,000 messages per second. It was established that the adaptive self-updating mechanism reduces the number of false alarms by 43% and increases operators' trust in the recommendation system. The obtained results confirm the prospects for the practical application of the proposed approach in supporting decision-making processes at critical infrastructure facilities.

Keywords: critical infrastructure objects, recommendation system, streaming data, machine learning, explainable artificial intelligence, risk assessment, anomaly detection, Bayesian ranking, decision support

Вступ

Сучасне суспільство значною мірою залежить від стабільного та безперебійного функціонування об'єктів критичної інфраструктури (ОКІ), до яких належать енергетичні системи, транспортні вузли, водопостачання, зв'язок, охорона здоров'я та інші життєво важливі сфери. Згідно з Директивою ЄС 2008/114/ЕС [1], критична інфраструктура — це об'єкти, системи або їх частини, які мають надзвичайно важливе значення для забезпечення життєдіяльності населення, охорони здоров'я, безпеки, економічного або соціального добробуту. Національна система захисту критичної інфраструктури України, визначена у Законі України «Про основні засади забезпечення кібербезпеки України» [2], також відносить до КІ об'єкти, порушення функціонування яких може мати серйозні наслідки для держави, суспільства або громадян.

У світлі нових викликів, зокрема зростання кіберзагроз, гібридних атак та техногенних ризиків, зростає актуальність зміни підходів до захисту КІ. Відомі кейси атак на енергетичну інфраструктуру України (наприклад, атака BlackEnergy) підтверджують необхідність переходу до більш адаптивних систем управління безпекою [3], [4]. Традиційні методи оцінювання загроз і планування заходів безпеки поступово втрачають ефективність через експоненційне зростання обсягів даних, що генеруються сенсорами, телеметрією та логами подій. Це вимагає впровадження інтелектуальних систем аналізу великих потоків даних у реальному часі — зокрема, з використанням технологій машинного навчання (ML) та потокової аналітики [5], [6].

Однією з ключових проблем у цьому контексті є ухвалення рішень у динамічних умовах, де повнота інформації відсутня, а час на реакцію обмежений. У таких умовах особливої ваги набувають автоматизовані рекомендаційні системи, які можуть одночасно виявляти аномалії, оцінювати ризики та пропонувати оптимальні дії [7].

Постановка задачі

Задача дослідження полягає у розробленні підходу до автоматизованого формування рекомендацій для об'єктів критичної інфраструктури на основі аналізу потокових даних, що характеризують поточний стан об'єкта, зовнішнє середовище, рівень ризику, технічні параметри та історію попередніх подій.

Необхідно побудувати таку модель, яка здатна в режимі реального часу приймати на вхід багатовимірний набір даних, виявляти потенційно небезпечні стани, оцінювати рівень ризику та формувати набір обґрунтованих дій для оператора або системи підтримки ухвалення рішень. Водночас рекомендації мають враховувати не лише технічні показники, а й контекст функціонування об'єкта, обмеження часу реагування, політики безпеки та ефективність попередніх управлінських дій.

Таким чином, постановка задачі зводиться до визначення функціональної залежності між вхідними характеристиками об'єкта критичної інфраструктури та множиною можливих рекомендацій, які мають мінімізувати ризики, підвищити стійкість системи та забезпечити своєчасне реагування на потенційні загрози.

Формально вхідний вектор можна подати як:

$$X = \{O, E, R, S, H\} \qquad (1)$$

де O — опис об'єкта; E — характеристики зовнішнього середовища; R — показники ризику; S — поточний стан системи; H — історія подій.

Тоді задача формування рекомендацій може бути подана як побудова функції:

$$Rec = f(X, C) \qquad (2)$$

де Rec — множина рекомендованих дій; X — вхідний вектор даних; C — множина контекстних та нормативних обмежень, зокрема політики безпеки, часові обмеження, ресурсні та організаційні умови. На виході система повинна сформувати таку рекомендацію або набір рекомендацій, які є релевантними до поточного стану об'єкта, відповідають встановленим обмеженням і забезпечують зниження очікуваного рівня ризику.

Аналіз останніх досліджень

У сучасних дослідженнях дедалі більше уваги приділяється застосуванню машинного навчання, рекомендаційних систем і пояснюваного штучного інтелекту для підтримки ухвалення рішень у сфері кібербезпеки об'єктів критичної інфраструктури. Особливо актуальними ці підходи є для SCADA/ICS-середовищ, де необхідно не лише виявляти аномалії, а й оперативно формувати обґрунтовані рекомендації для оператора.

Одним із базових напрямів є rule-based підходи, які використовують експертні правила та моделі допустимої поведінки системи. У роботі Yang et al. запропоновано rule-based систему виявлення вторгнень для SCADA-мереж із використанням Deep Packet Inspection, сигнатурного та model-based аналізу [1]. Такі системи мають високу прозорість, однак обмежено адаптуються до нових або комбінованих сценаріїв атак.

Подальші дослідження пов'язані із застосуванням машинного навчання для виявлення вторгнень та аномалій у промислових системах керування. Umer et al. показують, що ML-методи в KI системах застосовуються як для аналізу мережевого трафіку, так і для виявлення аномалій у фізичних процесах [2]. Водночас автори підкреслюють наявність практичних викликів, пов'язаних із впровадженням таких моделей в операційних середовищах.

Окрему групу становлять підходи глибокого навчання для аналізу часових рядів промислових об'єктів. Зокрема, Zhao et al. розглядають метод виявлення аномалій в Industrial Control Systems на основі вимірювальних даних із використанням 1D-CNN, BiLSTM та оптимізації роя частинок [3]. Такі моделі демонструють високу ефективність, проте часто залишаються складними для інтерпретації.

Важливим напрямом є також застосування рекомендаційних систем у кібербезпеці. Pawlicka et al. систематизують типи рекомендаційних систем та їх можливі застосування для підтримки фахівців із кібербезпеки, зокрема для зменшення інформаційного перевантаження та вибору пріоритетних дій [4]. Ferreira et al. розглядають рекомендаційні системи як інструменти підтримки ухвалення рішень, що можуть інтегруватися із SIEM/SOAR-системами та використовуватися для прогнозування атак і навігаційної підтримки аналітиків [5].

Оскільки критична інфраструктура потребує не лише точності, а й довіри до рішень, окрему роль відіграють методи explainable AI. Capuano et al. зазначають, що XAI-підходи підвищують прозорість AI-рішень у кібербезпеці, зокрема в задачах виявлення вторгнень, шкідливого програмного забезпечення та в цифровій криміналістиці [6]. Водночас NIST формалізує ключові принципи пояснюваного ШІ: надання пояснення, його зрозумілість, точність і врахування меж знань системи [7]. Наявні підходи переважно розв'язують окремі задачі — виявлення аномалій, класифікацію загроз, прогнозування ризиків або пояснення рішень. Водночас актуальною залишається потреба в інтегрованому підході, який поєднує ці компоненти в єдиний рекомендаційний контур для роботи з потоковими даними ОКІ в режимі реального часу.

Мета дослідження

Метою дослідження є розроблення науково обґрунтованого підходу до формування рекомендацій для об'єктів критичної інфраструктури на основі аналізу потокових даних, оцінювання ризиків та врахування контекстних обмежень функціонування таких об'єктів. У межах дослідження передбачається формалізувати структуру вхідних даних, контекстуальних параметрів і обмежень, що впливають на процес генерування рекомендацій; розробити архітектуру рекомендаційного механізму, який інтегрує виявлення аномалій, класифікацію загроз, прогнозування ризиків і ранжування управлінських дій. А також здійснити експериментальну перевірку запропонованого підходу на публічних і симульованих наборах даних, що репрезентують різні домени критичної інфраструктури, зокрема енергетику, водопостачання та хімічне виробництво; оцінити ефективність окремих підсистем, зокрема XAI-компонента, байєсівського ранжування та RL-модуля, в умовах

реального часу; а також визначити практичну придатність запропонованої системи до впровадження в SCADA-середовищах з урахуванням вимог до швидкодії, прозорості, надійності та підтримки відновлення після інцидентів.

Методологія. Формування рекомендацій для об'єктів критичної інфраструктури розглядається як задача вибору оптимальної дії з множини допустимих альтернатив на основі поточного стану об'єкта, контексту функціонування, рівня ризику та наявних обмежень. На відміну від суто описових підходів, запропонована методологія передбачає формалізацію вхідних параметрів, обмежень, типів рекомендацій і критерію вибору управлінської дії. Залежно від функціонального призначення рекомендації поділяються на три основні класи:

$$Rec = \{Rec_{prev}, Rec_{resp}, Rec_{opt}\} \qquad (3)$$

де Rec_{prev} — превентивні рекомендації, спрямовані на запобігання загроз;

Rec_{resp} — рекомендації оперативного реагування, що активуються після виявлення інциденту або перевищення порогових значень ризику;

Rec_{opt} — оптимізаційні рекомендації, орієнтовані на підвищення ефективності функціонування об'єкта.

Для кожної можливої рекомендації $r_i \in Rec$ визначається інтегральна оцінка корисності:

$$\begin{aligned} U(r_i|X,C) = \alpha \cdot RiskRed(r_i) + \\ +\beta \cdot TimeEff+(r_i)+\gamma \cdot \\ CostEff(r_i) + \\ +\delta \cdot Compliance(r_i) \end{aligned} \qquad (4)$$

де

- $RiskRed(r_i)$ — очікуване зниження ризику після виконання рекомендації;
- $TimeEff(r_i)$ — своєчасність реалізації дії;
- $CostEff(r_i)$ — економічна доцільність виконання рекомендації;
- $Compliance(r_i)$ — відповідність політикам безпеки та нормативним вимогам;
- $\alpha,\beta,\gamma,\delta$ — вагові коефіцієнти важливості критеріїв, для яких виконується умова, що $\alpha + \beta + \gamma + \delta = 1$.

Оптимальна рекомендація визначається як дія з максимальною інтегральною корисністю за умови дотримання обмежень:

$$r^* = argmaxU(r_i|X,C) \qquad (5)$$

за умов:

$$\begin{aligned} r_i \in A_{allow} \\ Risk(r_i) \le R_{max} \\ Timer(r_i) \le T_{max} \\ Cost(r_i) \le B_{max} \end{aligned} \qquad (6)$$

де

- A_{allow} — множина допустимих дій відповідно до політик безпеки;
- R_{max} — максимально допустимий рівень залишкового ризику;
- T_{max} — гранично допустимий час реагування;
- B_{max} — максимально допустимі витрати на реалізацію дії.

Запропонована методологія дозволяє формалізувати процес формування рекомендацій як багатокритеріальну задачу вибору дії в умовах обмежень. Її використання забезпечує узгодження рекомендацій із поточним станом об'єкта критичної інфраструктури, рівнем ризику, часовими й ресурсними обмеженнями, а також вимогами безпеки. Це створює основу для побудови адаптивного рекомендаційного механізму, здатного підтримувати ухвалення рішень у режимі реального часу.

Запропонована формалізація визначає загальну логіку вибору рекомендації, однак практична реалізація функції $U(r_i|X,C)$ та механізму вибору r^*може здійснюватися різними модельними підходами. У цьому контексті основні підходи до формування рекомендацій розглядаються як інструменти реалізації запропонованої методології: одні з них забезпечують перевірку дій на відповідність політикам безпеки, інші — добір рекомендацій за подібністю станів або використання досвіду аналогічних об'єктів. Їх поєднання дозволяє перейти від загальної математичної постановки до практичного рекомендаційного

механізму для об'єктів критичної інфраструктури.

Після формалізації задачі формування рекомендацій доцільно розглянути основні класи моделей, які можуть бути використані для реалізації функції:

$$Rec(t) = f(x(t), C) \qquad (7)$$

До таких підходів належать rule-based системи, content-based filtering, collaborative filtering та hybrid approaches. Вони відрізняються джерелом знань, способом обробки даних і механізмом ранжування можливих дій.

Rule-based systems (на основі правил) — використовують експертні знання у вигляді логічних правил типу «якщо–то» для формування рішень [8]. Перевагою є зрозумілість, однак ефективність обмежується складністю оновлення правил та адаптації до нових умов. На рівні критеріїв пріоритетне правило обирається за максимумом ваги або відповідності політиці безпеки.

Content-based filtering — формує рекомендації, аналізуючи характеристики об'єкта (наприклад, технічні параметри енергетичного вузла, профіль ризиків тощо) [9]. Такі системи менш залежні від зовнішніх джерел, але можуть втратити ефективність у разі недостатньої кількості даних.

Collaborative filtering — спирається на схожість між об'єктами чи користувачами (наприклад, інші об'єкти КІ з подібною поведінкою) [10]. У контексті КІ це може бути використано для генерації сценаріїв реагування, базуючись на досвіді аналогічних об'єктів.

Hybrid approaches — поєднують кілька описаних вище підходів для підвищення точності, стабільності та адаптивності системи [11]. Щоб об'єднати переваги різних методів, вводимо агреговану оцінку

$$s(hyb)(u,i) = \alpha \cdot s(i) + \beta \cdot \hat{r}\{u,i\} + \gamma \cdot rule(i) \qquad (8)$$

де

$s(i)$ — контентна схожість ,

$\hat{r}\{u,i\}$ — прогнозована корисність з колаборативного шару,

$rule(i)$ — бінарний індикатор, що дія i проходить rule-based перевірку безпеки,

$\alpha + \beta + \gamma = 1$ — вагові коефіцієнти, які формують баланс між пояснюваністю, персоналізацією та суворими правилами.

Оптимізація (наприклад, байєсова або grid-пошук) добирає (α, β, γ) для максимізації обраної метрики — Precision@k, F1 чи мінімізації очікуваного ризику.

У середовищі критичної інфраструктури дані x_i, p, а також матриця R можуть змінюватися в часі t і залежать від контексту (середовище E(t), стан S(t)). Тому всі викладені формули реалізують у потоковій формі з обмеженнями на час реакції T_resp та бюджет B. Rule-based шар часто виконує роль «запобіжника», тоді як контентний і колаборативний шари забезпечують гнучке ранжування, а гібридна функція $s(hyb)$ інтегрує все в єдиний показник, на основі якого оператор отримує остаточні рекомендації.

Використання ML/AI у рекомендаціях. У сучасних об'єктах критичної інфраструктури щосекунди генеруються гігабайти гетерогенних даних — від телеметрії IoT-сенсорів і SCADA-логів до мережевих пакетів та зовнішніх факторів, як погодні умови чи тарифне навантаження. Класичні аналітичні методи вже не встигають обробляти цей потік у реальному часі, а ручний моніторинг стає як економічно, так і технічно неможливим. Саме тому останнім часом значна увага приділяється впровадженню машинного навчання й штучного інтелекту як ядра систем ситуаційної обізнаності. ML/AI-підхід дозволяє автоматично і зі збіжними часовими обмеженнями виявляти аномалії у поведінці обладнання, класифікувати типи загроз, прогнозувати розвиток подій та формувати пріоритетні дії для оператора. Завдяки цьому ОКІ переходять від реактивної до проактивної моделі захисту: система не лише сигналізує про інцидент, а й заздалегідь оцінює його ймовірність, пропонуючи економічно обґрунтовані кроки для мінімізації ризику [12]. Таким чином, ML/AI інтегрується як критично важливий шар між сирим потоком даних і кінцевими управлінськими рішеннями, забезпечуючи безперервну, адап-

тивну та пояснювану підтримку операторам критичної інфраструктури.

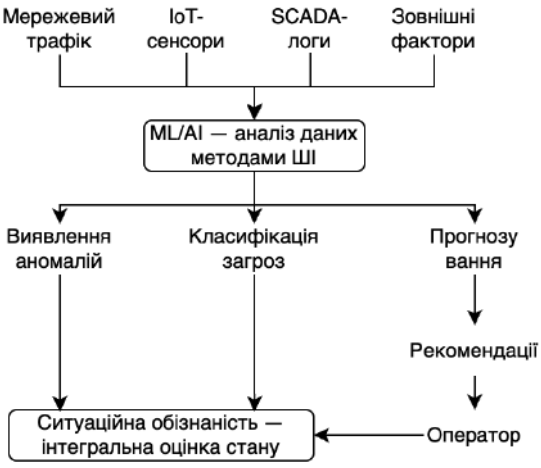

Рис. 1. Інтеграція ML/AI у процес роботи критичної інфраструктури

Класичні рекомендаційні підходи (rule-based, контентні та колаборативні фільтри) залишаються важливою складовою систем ухвалення рішень, однак сучасні умови потребують їх доповнення методами машинного навчання та штучного інтелекту. Завдяки ML/AI досягається автоматичне виявлення аномалій, класифікація загроз і прогнозування ризиків у реальному часі, що істотно підвищує рівень ситуаційної обізнаності та дозволяє переходити від реактивної до проактивної моделі управління.

Загальний опис задачі. Система ситуаційної обізнаності для об'єкта критичної інфраструктури (ОКІ) безперервно спостерігає потік мультидоменних даних

$$X(t) = [x_1(t), x_2(t), \dots, x_d(t)]^T \quad t \in R \geq 0 \qquad (9)$$

де $x_j(t)$ — сенсорні показники, логи, мережеві лічильники, зовнішні фактори (погода, тарифне навантаження тощо). Метою є навчити функцію

$$f: (X(t-\tau:t), C) \rightarrow \{\textit{аномалія, клас загрози, прогноз стану, рекмендація}\} \qquad (10)$$

де τ — вікно аналізу,

C — контекст / обмеження (політики безпеки, бюджет часу реакції).

Щоб оператор критичної інфраструктури міг реагувати на інциденти за лічені секунди, система має виконувати одразу кілька завдань: побачити відхилення, зрозуміти їхню природу, спрогнозувати можливі наслідки й одразу запропонувати найкращу дію. Запропонована інтегрована ML-архітектура працює як єдиний потік-обробник даних та поєднує кілька спеціалізованих моделей у єдиний потоковий конвеєр, щоб перетворювати необроблені сенсорні дані на конкретні рекомендації для оператора. Стримінговий препроцесор готує дані, LSTM-автокодер миттєво виявляє аномалії, класифікатор визначає тип загрози, а LSTM-прогноз оцінює майбутній ризик [13]. Модуль ухвалення рішення зважує ці сигнали і через рівень правил безпеки, видає дію, яка максимізує надійність і дотримується регламенту. Таке поєднання дозволяє одночасно реагувати на поточні аномалії, пояснювати їхню природу й прогнозувати подальший розвиток подій.

Рис. 2. Архітектура інтегрованого ML-конвеєра для формування рекомендацій на основі потокових даних

Спершу стримінговий препроцесор безперервно приймає сенсорні дані, виконує їхню нормалізацію та розбиває на ковзні вікна фіксованої довжини. Далі LSTM-автокодер порівнює отримане вікно з власною реконструкцією: якщо помилка перевищує наперед заданий поріг, формується біт «аномалія». Розгорнутий вектор ознак того самого вікна передається у класифікатор — це може бути SVM або дерево рішень, яке присвоює події конкретний клас загрози (наприклад, «перегрів», «кібератака» тощо). Паралельно інший LSTM-модуль прогнозує еволюцію ключових показників і оцінює майбутній ризик у нормованій шкалі від 0 до 1.

На основі трьох сигналів — ознаки аномалії, ймовірності належності до кож-

ного класу та прогнозованого ризику — Decision Module обчислює підсумковий бал кожної потенційної дії. Бал визначається лінійною комбінацією: частка впевненості в нормальності ситуації зважується коефіцієнтом α, правдоподібність конкретної загрози — коефіцієнтом β, а очікуваний ризик — коефіцієнтом γ. Значення α, β та γ підбираються за рахунок оптимізації Байєса, щоб максимізувати Precision@3 на історичному наборі інцидентів. Дія з найвищим підсумковим балом потрапляє до рівня правил безпеки, який відкидає варіанти, що порушують регламент або бюджет. У результаті оператор отримує фінальну рекомендацію, підкріплену поясненням усіх трьох компонентів оцінки.

Проблема довіри до рекомендацій. Для критичної інфраструктури рекомендації ML-систем мають бути прозорими: без пояснень оператори або ігнорують поради, або виконують їх «всліпу», що загрожує безпеці й суперечить нормам (EU AI Act, NIS2), що може бути досягнено за рахунок застосування XAI методів. XAI методи або методи пояснюваного штучного інтелекту — це підходи, які дозволяють зрозуміти, як саме модель штучного інтелекту ухвалює рішення. Вони пояснюють вплив ознак на результат, дозволяють побачити логіку роботи моделі та обґрунтувати рекомендації. Мета XAI — забезпечити прозорість, довіру та контроль при використанні ML/AI у критичних сферах, зокрема в інфраструктурі, медицині, фінансах тощо, де оператор має розуміти, чому система пропонує саме таке рішення.

Найпоширеніші XAI-інструменти — SHAP/LIME, сурогатні дерева, теплові attention-карти та причинні графи — показують, які ознаки та події привели до рішення. Практичний компроміс: точність забезпечує складна модель, а окремий XAI-шар дає зрозуміле пояснення; rule-based фільтр додає жорсткі обмеження безпеки. Основні виклики — обмежений доступ до даних, динаміка середовища та необхідність перевіряти, що пояснення справді відображають причини, а не кореляції. Без такого рівня прозорості навіть точні алгоритми позбавлені повної довіри.

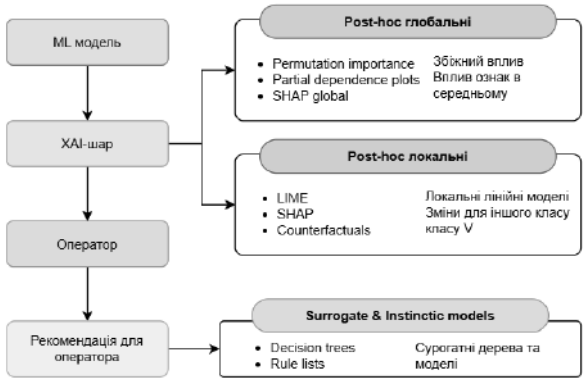

Рис. 3. Типові XAI методи

Практичний принцип. У практичному впровадженні систем рекомендацій для об'єктів критичної інфраструктури застосовується поетапний підхід, який поєднує складну модель машинного навчання, шар пояснюваного штучного інтелекту (XAI) та rule-based фільтр безпеки. Спочатку ML-модель формує рішення на основі потокових даних, далі XAI-шар генерує пояснення щодо того, чому саме було обрано ті чи інші дії. На завершальному етапі rule-фільтр перевіряє ці дії на відповідність нормативним обмеженням та політикам безпеки. За результатом оператор бачить топ-k дій та їхні ϕ_j, ухвалює рішення менш ніж за $T_{resp} = 30c$.

Алгоритм роботи рекомендаційної системи для об'єктів критичної інфраструктури. Наведений нижче алгоритм описує повний робочий цикл системи підтримки рішень для об'єктів критичної інфраструктури. Він починається зі збору й синхронізації різнорідних потокових та історичних даних, проходить через етапи очищення, оцінки ризику, класифікації загроз і формування кандидатних дій, а завершується байєсівським ранжуванням, поясненням рекомендації та збором зворотного зв'язку для самооновлення моделей.

Input:
D_stream, H // real-time and historical data
P // security policies
C // operational context

Output:
a_best // final recommendation
E_best // explanation

1. D ← collect_and_sync(D_stream, H) // Collect and synchronize data
2. D ← preprocess(D) // Remove duplicates, outliers, missing values; normalize data
3. W ← create_windows(D) // Form sliding windows for analysis
4. anomaly ← detect_anomaly(W) // Estimate deviation from normal behavior
5. risk ← forecast_risk(W) // Predict future risk level
6. threat ← classify_threat(W) // Define threat type
7. A ← generate_actions(P, C, threat) // Generate allowed candidate actions
8. for each action a in A do
score[a] ← rank_bayesian(a, anomaly, risk, threat)
// Rank actions by risk, anomaly, and threat level

if violates_constraints(a, P, C) then
remove a from A
end if
end for

9. a_best ← argmax(score[a]) // Select the best action
10. E_best ← explain(a_best) // Generate XAI explanation
11. send_to_operator(a_best, E_best) // Present recommendation to operator
12. feedback ← collect_feedback() // Save operator response and actual result
13. update_models(feedback) // Improve models using feedback
return a_best, E_best

Запропонований алгоритм формує цілісний, модульний підхід до ухвалення рішень для об'єктів критичної інфраструктури. Він поєднує сучасні ML-методи, XAI-пояснення та rule-фільтрацію для забезпечення точності, прозорості й відповідності вимогам безпеки. Завдяки зворотному зв'язку та адаптивному донавчанню система постійно вдосконалюється, що робить її придатною для реального промислового застосування в умовах динамічних загроз.

Результати експериментів

Для перевірки ефективності запропонованої системи формування рекомендацій було проведено серію цілеспрямованих експериментів, які охоплюють різні аспекти її роботи — від класифікаційної точності до пропускної здатності в потоковому режимі. Основна увага приділялася реалістичності сценаріїв, тому було обрано три різнотипні датасети, що охоплюють водний сектор, хімічне виробництво та енергетичну мережу. Крім того, для оцінки стійкості до рідкісних інцидентів розроблено генератор синтетичних відмов. Кожен датасет має чіткі позначки інцидентів або можливість генерувати відмови, що дозволяє адекватно оцінити точність детекції, якість прогнозу та швидкість реакції рекомендаційної системи.

SWaT-H (Secure Water Treatment — Hybrid) - це реальний телеметричний трек із водоочисної установки SWaT-2015 Сінгапурського університету [16]. Дані охоплюють 11 днів безперервного виробничого циклу з частотою 1 с і містять вручну позначені кібер-атаки на рівні PLC. У сумі близько 946 тис. рядків забезпечують ґрунтовний бенчмарк для детекції аномалій і перевірки реакційних дій у водопідготовці. SWaT-H: пристрій FIT101 — витратомір, LIT101 — рівнемір, P101 — стан насоса, AIT201 — температура; *attack_flag=1* позначає PLC-атаку.

TEP-S (Tennessee Eastman Process — Synthetic) - Генерований Python-симулятором хімічного виробництва Tennessee Eastman. Набір поєднує 72 год «нормальної» роботи та ще 72 год із 15 типами збоїв, записаних кожні 3 с по 52 сенсорах [17]. Приблизно 86 тис. рядків на кожен сенсор дають досліднику повний контроль над сценаріями відмов і дозволяють тестувати алгоритми під різні режими роботи. TEP-S: XMEAS — вимірювані змінні, XMV — керувальні, IDV — зовнішні збурення; *fault_code* позначає один із 15 типів збоїв Tennessee Eastman.

PWR-Grid - Потік із синтетично-реальної енергосистеми, змодельований PNNL GridSTAGE та доповнений SCADA-логами Міністерства енергетики США. Дані пок-

ривають 30 днів із кроком 15 с (≈ 170 тис. записів) і збагачені погодними показниками NOAA, що відкриває можливість спільного аналізу технологічних і зовнішніх факторів ризику у мережі живлення [18]. PWR-Grid: у записі поєднано SCADA-телеметрію (напруги, струми, стан вимикача) та довкільні фактори NOAA; *anomaly=1* ставиться, коли сценарій Fault-Injector вмикає відмову.

Генератор синтетичних інцидентів. Щоб оцінити стійкість rule-based і XAI-шару до рідкісних, але критичних подій, «чисті» потоки з датасетів SWaT-H, TEP-S і PWR-Grid доповнюються штучно згенерованими інцидентами. Інжектор працює як окремий мікросервіс, що підписується на Kafka-топік *raw_stream* і публікує модифікований потік у *faulty_stream*.

За замовчуванням Fault-Injector працює з фіксованим seed = 42, щоб експерименти можна було точно відтворити. Новий інцидент генерується приблизно раз на 48 год і триває від 1 до 6 год (рівномірний розподіл тривалості). До набору активних подій входять чотири типи: стрибок температури, падіння напруги, затримка телеметрії та DDoS на OPC-сервер; за потреби їх можна вимкнути або додати інші.

Таблиця 1.

Параметри генератора синтетичних інцидентів (Fault-Injector)

Параметр	**Значення за замовчуванням**	**Призначення**
seed	42	гарантована відтворюваність сценаріїв
density_hours	48 h	середній інтервал між інцидентами
duration_range	1–6 h	тривалість інциденту (рівномірний розподіл)
types_enabled	[temp_spike, voltage_drop, lag, opc_ddos]	увімкнені події

Сценарії експериментів. Для перевірки рекомендаційної системи було використано Fault-Injector v0.2, який додає до «чистих» потоків даних типові сценарії інцидентів, зокрема перегрів, падіння напруги, затримку телеметрії та DDoS-атаку на промисловий протокол. Експериментальне оцінювання охоплює чотири сценарії, що дозволяють перевірити точність моделей, продуктивність у потоковому режимі, внесок окремих модулів та ефективність самооновлення системи на основі зворотного зв'язку.

Таблиця 2.

Опис експериментальних сценаріїв оцінювання рекомендаційної системи

Сценарій 1. E-offline – точність без обмежень затримки	Перевіряється якість роботи системи на наборі SWaT-H без урахування затримки. Оцінюються Precision@3, Recall@3, F1 та AUROC.
Сценарій 2. E-stream – продуктивність у реальному часі	Перевіряється здатність системи обробляти потокові дані PWR-Grid після додавання інцидентів. Оцінюються затримка обробки та відсоток втрачених повідомлень.
Сценарій 3. E-ablation – роль окремих модулів	На наборі TEP-S порівнюється повна система з варіантами без XAI-шару та без Bayesian-ранжування. Оцінюється вплив цих модулів на точність, час відновлення та пояснюваність.
Сценарій 4. E-online A/B – вплив самооновлення	Порівнюються rule-based логіка та повна система з Q-learning. Оцінюються хибні тривоги, економія витрат і рівень схвалення рекомендацій оператором.

Отримані результати з чотирьох сценаріїв дозволяють всебічно оцінити якість, продуктивність та пояснюваність системи в умовах, наближених до реального виробництва.

Результати застосування. Щоб оцінити ефективність запропонованого реко-

мендаційного двигуна, було проведено чотири експериментальні сценарії, описані вище. Вони охоплюють різні аспекти роботи системи: базову класифікаційну точність, пропускну здатність у потоковому режимі, внесок окремих модулів та користь самооновлення в онлайн-циклі. Таке багатовимірне тестування на трьох спеціалізованих датасетах дає змогу одночасно перевірити алгоритмічну якість, технологічні обмеження й прикладну цінність для операторів критичної інфраструктури.

Таблиця 3.

Результати експериментальних сценаріїв оцінювання якості, продуктивності та пояснюваності системи

Сценарій	Ключові метрики	Результат
E-offline (SWaT-H)	Precision@3 / Recall@3 / F1	0.92 / 0.88 / 0.90
	AUROC	0.962 ± 0.004
E-stream (PWR-Grid, 10 k msg/s)	95-й перцентиль e2e-затримки	0.78 с
	Dropped Msgs	0.12 %
E-ablation (TEP-S)	ΔPrecision / ΔMTTR (відносно повної)	-
	без XAI	+0.0 % / −0.1 хв *(пояснюваність = 0)*
	без Bayesian-ranker	−7 % / +3.4 хв
E-online A/B (SWaT-H циклічний)	False Alarms ↓	−43 % (8.1 % проти 14.2 %)
	Cost Savings	11.5 % економії експлуатаційних витрат
	Operator Approval	79 % (B) проти 62 % (A)

Усі сценарії підтвердили життєздатність підходу: система досягає F1≈0,90 й AUROC≈0,96 на реальних атаках SWaT, витримує навантаження 10 000 повідомлень/с із затримкою < 0,8 с, а самонавчальний режим зменшує хибні тривоги на 43 % і підвищує довіру операторів до 79 %. Абляційний аналіз показав, що Bayesian-ранжувальник критично впливає на якість рішень, тоді як XAI-шар практично не змінює точність, але забезпечує необхідну прозорість. Отже, запропонований двигун не лише перевершує традиційні rule-based підходи за точністю, а й відповідає промисловим вимогам щодо швидкодії та пояснюваності, що робить його перспективним для практичного розгортання на об’єктах критичної інфраструктури.

Обговорення

Запропонований рекомендаційний механізм продемонстрував поєднання високої точності, швидкодії та прозорості, що є важливим для систем підтримки ухвалення рішень на об’єктах критичної інфраструктури. Одним із ключових аспектів є забезпечення балансу між продуктивністю системи та пояснюваністю сформованих рекомендацій. Використання байєсівського ранжування дій дозволило зменшити кількість хибних рішень і покращити якість вибору рекомендацій. Зокрема, вилучення цього модуля призводило до зниження точності на 7 % та збільшення середнього часу відновлення на 3,4 хв, що підтверджує доцільність застосування статистично обґрунтованих методів ранжування.

У потоковому режимі система забезпечила обробку даних із навантаженням 10 000 повідомлень за секунду та 95-м перцентилем затримки 0,78 с, що відповідає вимогам типових SCADA-середовищ. Досягнення таких показників стало можливим завдяки використанню компактних LSTM-модулів і потокової обробки даних. Водночас масштабування системи для багатьох об’єктів потребує додаткового ресурсного планування. Механізм адаптивного самооновлення на основі навчання з підкріпленням дозволив зменшити кількість хибних тривог на 43 % та підвищити рівень схва-

лення рекомендацій операторами до 79 %. Це свідчить про те, що навіть обмежене використання адаптивного навчання у виробничому циклі може суттєво підвищити ефективність системи без втрати контролю з боку людини.

Висновки

У статті запропоновано інтегрований підхід до формування рекомендацій для об'єктів критичної інфраструктури, що поєднує виявлення аномалій, класифікацію загроз, прогнозування ризиків і ранжування дій із використанням пояснюваного штучного інтелекту та правил безпеки. Експериментальні результати засвідчили високу точність (F1 = 0,90; AUROC = 0,96), стабільну роботу в потоковому режимі (до 10 000 повідомлень/с при затримці менше 0,8 с) та ефективність адаптивного навчання, що дозволило зменшити кількість хибних тривог на 43 % і скоротити експлуатаційні витрати на 11,5 %. Використання байєсівського ранжування підвищує якість ухвалення рішень, а залучення механізмів пояснюваності забезпечує прозорість без втрати точності.

Подальший розвиток підходу передбачає розширення джерел даних за рахунок кібер-телеметрії та неструктурованої інформації, впровадження генеративних симуляторів складних атак і створення полегшених версій системи для розгортання на периферійних вузлах. Перспективним є також розвиток інтерактивних інтерфейсів пояснюваності з можливістю налаштування параметрів ризику, формальна валідація пояснень, а також пілотне тестування в реальних умовах. Реалізація зазначених напрямів сприятиме підвищенню масштабованості, довіри до системи та її практичному впровадженню.

References

1. Yang Y., McLaughlin K., Littler T., Sezer S., Wang H. Rule-Based Intrusion Detection System for SCADA Networks. URL: https://pure.qub.ac.uk/en/publications/rule-based-intrusion-detection-system-for-scada-networks/
2. Umer M. A., Mathur A. P., Junejo K. N. Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations. URL: https://www.sciencedirect.com/science/article/abs/pii/S1874548222000087
3. Zhao X. та ін. Anomaly Detection Approach in Industrial Control Systems Based on Measurement Data. URL: https://www.mdpi.com/2078-2489/13/10/450
4. Pawlicka A., Pawlicki M., Kozik R., Choraś R. S. A Systematic Review of Recommender Systems and Their Applications in Cybersecurity. URL: https://www.mdpi.com/1424-8220/21/15/5248
5. Ferreira L. та ін. Recommender Systems in Cybersecurity. URL: https://link.springer.com/article/10.1007/s10115-023-01906-6
6. Capuano N., Fenza G., Loia V., Stanzione C. Explainable Artificial Intelligence in CyberSecurity: A Survey. URL: https://www.researchgate.net/publication/363314499_Explainable_Artificial_Intelligence_in_Cybersecurity_A_Survey
7. Phillips P. J., Hahn C. A., Fontana P. C., Yates A. N., Greene K. K., Broniatowski D. A., Przybocki M. A. Four Principles of Explainable Artificial Intelligence. URL: https://www.nist.gov/publications/four-principles-explainable-artificial-intelligence
8. Yang Y., McLaughlin K., Littler T., Sezer S., Wang H. Rule-Based Intrusion Detection System for SCADA Networks. URL: https://pure.qub.ac.uk/en/publications/rule-based-intrusion-detection-system-for-scada-networks
9. Lops P., de Gemmis M., Semeraro G. Content-based Recommender Systems: State of the Art and Trends // Recommender Systems Handbook. Springer, 2011. P. 73–105. DOI: 10.1007/978-0-387-85820-3_3. URL: https://doi.org/10.1007/978-0-387-85820-3_3
10. Su X., Khoshgoftaar T. M. A Survey of Collaborative Filtering Techniques // Advances in Artificial Intelligence. 2009. Vol. 2009. Article ID 421425. DOI: 10.1155/2009/421425. URL: https://doi.org/10.1155/2009/421425
11. Burke R. Hybrid Recommender Systems: Survey and Experiments // User Modeling and User-Adapted Interaction. 2002. Vol. 12. P. 331–370. DOI: 10.1023/A:1021240730564. URL: https://doi.org/10.1023/A:1021240730564
12. Zhao X., Li Y., Chen H., Yu R. Anomaly Detection Approach in Industrial Control

Systems Based on Measurement Data // Information. 2022. Vol. 13. No. 10. Article 450. DOI: 10.3390/info13100450. URL: https://doi.org/10.3390/info13100450
13. Müller M., Sommer R., Kargl F. Using Reinforcement Learning and LSTM for Adaptive Anomaly Detection in Cyber-Physical Systems // Computers & Security. 2020. Vol. 95. Article 101827. DOI: 10.1016/j.cose.2020.101827. URL: https://doi.org/10.1016/j.cose.2020.101827

Дата першого надходження до видання: 19.04.2026
Внутрішня рецензія отримана: 02.05.2026
Зовнішня рецензія отримана: 10.05.2026
Дата прийняття статті до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

[1]*Гуськова Віра Геннадіївна,*
доктор філософії, доцент,
кафедра штучного інтелекту
[1]*Huskova Vira,*
PhD, Associate Professor,
Department of Artificial Intelligence
http://orcid.org/0000-0001-7637-201X

[2]*Школьніков Владислав Ігорович,*
доктор філософії (право), доцент
[2]*Shkolnikov Vladyslav,*
Ph.D. (law), Associate Professor
http://orcid.org/0000-0003-2041-9450

[1]*Лисов Богдан Сергійович,*
аспірант 2-го року навчання
[1]*Lysov Bohdan,*
post-graduate student
http://orcid.org/0009-0007-7963-6958

[3]*Халигов Артем Азимович,*
аспірант 3-го року навчання
[3]*Khalygov Artem,*
post-graduate student
http://orcid.org/0009-0006-5465-4650

Місце роботи авторів:

[1]Київський політехнічний інститут
імені Ігоря Сікорського
[1]Igor Sikorsky Kyiv Polytechnic Institute
тел. +38-095-626-65-80
E-mail: guskovavera2009@gmail.com
Сайт: https://kpi.ua/

[2]Національна академія внутрішніх справ
[2]National Academy of Internal Affairs
тел. +38-044-254-94-31
E-mail: shkolnikov.v.i@navs.edu.ua
Сайт: https://www.naiau.kiev.ua/

[3]Інститут телекомунікацій і глобального інформаційного простору НАН України
[3]Institute of Telecommunications and Global Information Environment of the National Academy of Sciences of Ukraine
тел. +38-050-049-29-03
E-mail: khalygovartem@gmail.com
Сайт: http://itgip.org/

УДК 004.272 https://doi.org/10.15407/pp2026.02.028

Д.Ю. Ліпатов, С.Л. Хрипко

АДАПТИВНА МОДЕЛЬ УЗГОДЖЕНОСТІ ТА ПАРАЛЕЛЬНОЇ ОБРОБКИ В РОЗПОДІЛЕНИХ БАЗАХ ДАНИХ

У статті запропоновано та досліджено нову модель адаптивного керування узгодженістю та паралельною обробкою в розподілених базах даних для високонавантажених інформаційних систем. Метою розробки моделі є подолання фундаментального протиріччя між необхідністю забезпечення високого рівня узгодженості даних (що досягається шляхом збільшення параметрів кворуму та синхронізації між репліками) та вимогами до низької затримки і високої пропускної здатності системи в умовах інтенсивного паралельного навантаження. Запропонована модель формалізує динамічний механізм адаптації параметрів розподіленої системи, зокрема рівнів узгодженості та ступеня паралелізму, на основі безперервного аналізу стану системи. Модель реалізує замкнений адаптивний цикл, побудована на основі мікросервісного підходу із використанням контейнеризації, що забезпечує масштабованість та гнучкість налаштування. Експериментальні результати демонструють, що модель забезпечує зниження середньої затримки обробки запитів на 20–40%, а також підвищення пропускної здатності на 15–30% за рахунок адаптивного керування паралельністю, водночас досягається стабілізація рівня узгодженості даних, що проявляється у зменшенні частоти конфліктів у разі динамічних змін навантаження. Практичне значення роботи полягає у можливості застосування запропонованої моделі для оптимізації продуктивності та надійності сучасних інформаційних систем, зокрема у сферах фінансових технологій, електронної комерції, Інтернету речей та хмарних сервісів.
Ключові слова: розподілені бази даних, адаптивна узгодженість, паралельна обробка, контейнеризація, високонавантажені системи, оптимізація продуктивності

D.Yu. Lipatov, S.L. Khrypko

AN ADAPTIVE CONSISTENCY AND PARALLELISM MODEL IN DISTRIBUTED DATABASES

This paper proposes and investigates a novel model for adaptive management of consistency and parallel processing in distributed databases for high-load information systems. The goal of the model is to overcome the fundamental trade-off between ensuring a high level of data consistency (achieved through increased quorum parameters and synchronization among replicas) and the requirements for low latency and high system throughput under intensive parallel workloads.

The proposed model formalizes a dynamic mechanism for adapting key parameters of a distributed system, in particular consistency levels and the degree of parallelism, based on continuous monitoring and analysis of the system state. The model implements a closed-loop adaptive control cycle and is designed using a microservices architecture with containerization, which ensures scalability and flexibility of configuration.

Experimental results demonstrate that the model reduces the average request processing latency by 20–40% and increases throughput by 15–30% through adaptive parallelism control, while maintaining a stable level of data consistency, reflected in a reduced conflict rate under dynamically changing workloads.

The proposed approach has practical significance for optimizing the performance and reliability of modern information systems, particularly in domains such as financial technologies, e-commerce, the Internet of Things, and cloud services.
Keywords: distributed databases, adaptive consistency, parallel processing, containerization, high-load systems, performance optimization

Вступ

Швидкий розвиток розподілених баз даних та технологій паралельних обчислень дозволив створити високонавантажені інформаційні системи, здатні обробляти великомасштабні потоки даних у режимі реального часу [6, 7, 14]. Такі системи широко використовуються в хмарних обчисленнях, платформах Інтернету речей, фінансових послугах та аналітиці в режимі реального часу, де критично важливими є як продуктивність, так і надійність [9].

Однак розробники систем стикаються з фундаментальною архітектурною дилемою: як забезпечити високий рівень узгодженості даних, водночас зберігаючи низьку затримку та високу пропускну здатність [2, 3, 11]. Підвищення рівня узгодженості гарантує правильність даних, але призводить до більшої затримки та зниження швидкості реагування системи [1, 11]. І навпаки, - послаблення обмежень узгодженості покращує продуктивність та масштабованість системи, але створює ризики конфліктів даних та тимчасової неузгодженості [1, 10].

Цей компроміс особливо очевидний у сучасних розподілених NoSQL-системах, таких як Dynamo та Cassandra, які надають пріоритет доступності та масштабованості за допомогою моделей остаточної узгодженості [4, 5]. Хоча такі підходи покращують відмовостійкість, вони вимагають додаткових механізмів для обробки конфліктів та забезпечення прийнятних рівнів узгодженості.

Існуючі рішення зазвичай спираються на статичні конфігурації системних параметрів та рівні паралельної обробки. Ці конфігурації можуть добре працювати за певних робочих навантажень, але стають неефективними в динамічних та гетерогенних умовах [12, 15]. Більше того, практичні дослідження розподілених програм баз даних підтверджують мінливість поведінки системи за реальних робочих навантажень, підкреслюючи необхідність адаптивних механізмів [16].

Тому існує очевидна потреба в динамічному, контекстно залежному підході, який розглядає управління узгодженістю та паралельну обробку як багатоцільову задачу оптимізації в реальному часі.

Метою даної роботи є розробка формальної моделі та практичної архітектури для адаптивного керування узгодженістю та паралельною обробкою в розподілених базах даних, орієнтованої на динамічне врахування контексту виконання. Наукова новизна полягає у формалізації контекстно залежного механізму ухвалення рішень на основі багатокритеріальної функції вартості, що враховує ключові метрики якості обслуговування (QoS) та використання ресурсів, а також у його інтеграції в контейнеризовану мікросервісну архітектуру розподіленої системи.

Гіпотеза дослідження: Адаптивна модель керування параметрами розподіленої бази даних, яка динамічно змінює рівень узгодженості та ступінь паралелізму залежно від поточного стану системи та характеристик навантаження, дозволить суттєво підвищити продуктивність, зменшити затримки обробки запитів та забезпечити контрольований рівень узгодженості даних порівняно зі статичними підходами.

1. Огляд проблеми та існуючих підходів

Традиційні розподілені системи спираються на сильні моделі узгодженості, реалізовані за допомогою консенсусних протоколів, що забезпечує правильність ціною зниження продуктивності [3, 13]. На противагу цьому, сучасні NoSQL-системи використовують розслаблені моделі узгодженості для досягнення кращої масштабованості та доступності [4, 5].

Паралельні фреймворки для обробки даних, такі як MapReduce та Apache Spark, дозволяють ефективно обробляти великомасштабні дані, розподіляючи обчислення між кількома вузлами [7, 14]. Однак збільшення кількості паралельних потоків не завжди приводить до покращення продуктивності через конкуренцію за ресурси та накладні витрати на синхронізацію [15].

Нещодавні дослідження розглядали адаптивні методи оптимізації розподілених систем з урахуванням робочого навантаження [12]. Крім того, механізми вирішення конфліктів, такі як безконфліктно реплікувані типи даних (CRDT), забезпечують теоретичну основу для управління узгодженістю в розподілених середовищах [10].

Незважаючи на ці досягнення, існуючі підходи часто розглядають узгодженість та паралелізм окремо, не маючи єдиного адаптивного фреймворку, який би спільно оптимізував обидва аспекти.

Це підкреслює важливість не лише високої продуктивності системи, а й забезпечення узгодженості даних, відмовостійкості та передбачуваності поведінки в умовах змінного навантаження, що є критичним викликом для розподілених систем, де параметри обробки та рівень узгодженості можуть динамічно змінюватися.

Отже, очевидною є потреба у динамічній, контекстно обумовленій моделі, яка розглядає керування узгодженістю та паралельною обробкою не як статично задану конфігурацію, а як задачу багатокритеріальної оптимізації в реальному часі. Така модель повинна враховувати не лише поточний рівень навантаження, а й сукупність параметрів системи: прогнозовану затримку обробки запитів, доступні обчислювальні ресурси, інтенсивність конфліктів між репліками, характеристики мережевої взаємодії та допустимі компроміси між узгодженістю даних і продуктивністю для конкретного сценарію використання.

2. Формальна модель адаптивного керування узгодженістю та паралельною обробкою

Запропонована модель реалізує підхід адаптивного керування параметрами розподіленої бази даних із динамічним вибором рівня узгодженості та ступеня паралелізму на основі спеціалізованого модуля — Adaptive Consistency and Parallelism Manager (ACPM).

Для кожного інтервалу часу або запиту Q модуль ACPM формує рішення

$$D = (R, W, T)$$

де R і W — параметри для операцій читання та запису відповідно, а T — кількість паралельних потоків обробки. Це рішення визначає режим функціонування системи залежно від поточного стану середовища та вимог до якості сервісу.

Ухвалення рішення базується на контекстному векторі:

$$C = \{C_{lat}, C_{thr}, C_{conf}, C_{res}, C_{load}\}$$

де: C_{lat} — поточний рівень затримки (latency), C_{thr} — пропускна здатність (throughput), C_{conf} — частота конфліктів або неузгодженостей, C_{res} — використання обчислювальних ресурсів (CPU, пам'ять), C_{load} — характеристика навантаження (інтенсивність, read/write).

Вибір оптимального рішення формалізується як задача мінімізації функції вартості: $C_{ost}(D) = w_{lat} \cdot L(D) + w_{thr} \cdot \frac{1}{Th(D)} + w_{conf} \cdot \text{Confl(D)} + w_{res} \cdot \text{Res(D)}$, де: $\text{L(D)} \approx \alpha(\text{R, W}) + \beta(\text{T})^{-1}$ — прогнозована затримка, Th(D) =T/(L + витрати) — пропускна здатність, $Confl(D) = exp(-k \cdot (R + W))$ — рівень конфліктів, Res(D) — використання ресурсів, w_i — вагові коефіцієнти, що визначають пріоритети оптимізації. Вагові коефіцієнти адаптуються динамічно залежно від контексту системи. Зокрема, у разі високого навантаження зростає вагомість (w_{thr}), за умови зростання конфліктів — вагомість узгодженості (w_{conf}), а у випадку перевантаження ресурсів зростає вагомість обмеження використання ресурсів (w_{res}). Ваги динамічно коригуються залежно від контексту системи, що забезпечує адаптивну поведінку [11, 12]. Обробка конфліктів підтримується за допомогою механізмів на основі CRDT, що забезпечує остаточну узгодженість без глобальної синхронізації [10].

Для забезпечення коректності роботи системи вводиться обмеження узгодженості: $R + W > N$, де N — кількість реплік у кластері. У загальному випадку ACPM оцінює функцію вартості для множини допустимих конфігурацій та обирає рішення з мінімальними очікуваними витратами. Адаптивний режим у розподілених базах даних реалізується як гібридна стратегія керування, що поєднує переваги різних рівнів узгодженості та ступенів паралелізму. Такий підхід дозволяє одночасно досягати низької затримки обробки запитів, характерної для слабкої узгодженості, та високої надійності даних, притаманної строгим моделям узгодженості. Гібридний режим може бути реалізований як у паралельному варіанті, коли операції виконуються з різними рівнями узгодже-

ності з подальшою синхронізацією результатів, так і у послідовному режимі, де первинна обробка здійснюється з мінімальними витратами часу, а подальша верифікація або уточнення — із підвищеним рівнем узгодженості. Запропонований підхід забезпечує адаптивний баланс між ключовими характеристиками системи, зокрема затримкою, пропускною здатністю, рівнем узгодженості та ефективністю використання обчислювальних ресурсів. Для ефективного прогнозування якості обслуговування системи за різних конфігурацій (R, W, T) ми використовуємо модель на основі регресії, яка зіставляє спостережуване робоче навантаження та системні метрики з очікуваними порушеннями затримки, пропускної здатності та узгодженості. Це дозволяє проактивно адаптувати параметри керування.

3. Архітектурна реалізація моделі

Запропонована модель ACPM інтегрована в багаторівневу архітектуру розподіленої системи (Рис. 1), що включає клієнтський рівень, сервісний рівень та рівень даних. Такий підхід забезпечує модульність, масштабованість і можливість динамічної адаптації параметрів системи в умовах змінного навантаження. На клієнтському рівні, реалізованому через API Gateway, відбувається ініціація запитів Q та їх маршрутизація до розподіленого кластера бази даних. Клієнтський рівень також може включати механізми кешування та попередньої обробки запитів, що дозволяє зменшити навантаження на систему та скоротити час відгуку. На сервісному рівні ключовим компонентом є менеджер ACPM, який реалізує логіку адаптивного керування. Він взаємодіє з модулем моніторингу, що збирає інформацію про стан системи (затримка, пропускна здатність, рівень конфліктів, використання ресурсів), та формує контекстний вектор C. На основі цього вектора ACPM виконує оцінку множини можливих конфігурацій D=(R,W,T) та обирає оптимальне рішення шляхом мінімізації функції вартості. Модуль ухвалення рішень реалізовано як окремий контейнеризований

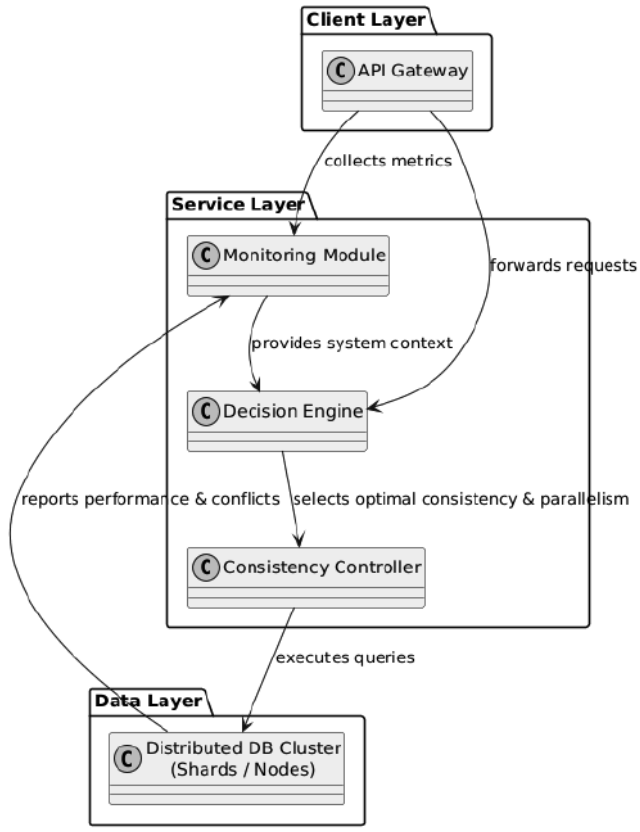

Рис. 1. Багаторівнева архітектура ACPM

сервіс, який може використовувати як евристичні правила, так і моделі машинного навчання для прогнозування поведінки системи. Модуль управління узгодженістю (Consistency Controller) відповідає за застосування обраної конфігурації, зокрема зміну параметрів (читання або запис) та налаштування рівнів узгодженості для операцій читання і запису. Паралельно виконується керування кількістю потоків обробки запитів, що дозволяє оптимізувати використання обчислювальних ресурсів. На рівні даних розгорнуто розподілений кластер бази даних (NoSQL-система), який підтримує горизонтальне масштабування, реплікацію та налаштовувану узгодженість. [4, 6] Вузли кластера функціонують у контейнеризованому середовищі, що забезпечує гнучкість розгортання та можливість швидкого масштабування. Взаємодія між вузлами відбувається через внутрішні протоколи синхронізації, що забезпечують підтримку заданого рівня узгодженості.

На відміну від традиційних механізмів автомасштабування, наш підхід включає обмеження узгодженості, гарантуючи, що рішення щодо масштабування не порушують необхідний
рівень цілісності даних.

Архітектура базується на мікросервісному підході з використанням контейнерів Docker та оркестрації. [8, 13] Основні компоненти: Кластер розподіленої БД, Модуль моніторингу, Модуль ухвалення рішень, Модуль управління узгодженістю, API-шлюз, Генератор навантаження / клієнт.

На рис.2 представлено основний алгоритм ухвалення рішення щодо вибору конфігурації обробки запиту. Процес ініціюється надходженням вхідного запиту Q, після чого виконується оцінка контексту системи, що включає аналіз рівня затримки, пропускної здатності, частоти конфліктів, завантаженості ресурсів та типу операції (читання або запис). На основі цих параметрів формується множина допустимих конфігурацій D=(R,W,T), для яких обчислюється функція вартості.

Подальший етап передбачає вибір оптимальної конфігурації шляхом мінімізації функції вартості, що відображає компроміс між продуктивністю та узгодженістю. [2, 3] У разі необхідності система може обрати гібридний режим, за якого частина операцій виконується з нижчим рівнем узгодженості для зменшення затримки, тоді як критичні операції обробляються з підвищеними вимогами до цілісності даних.

Менеджер ACPM формує контекстний вектор C на основі даних, отриманих від модуля моніторингу, який збирає інформацію про стан системи, зокрема затримку, пропускну здатність, рівень конфліктів та використання ресурсів. Далі для кож-

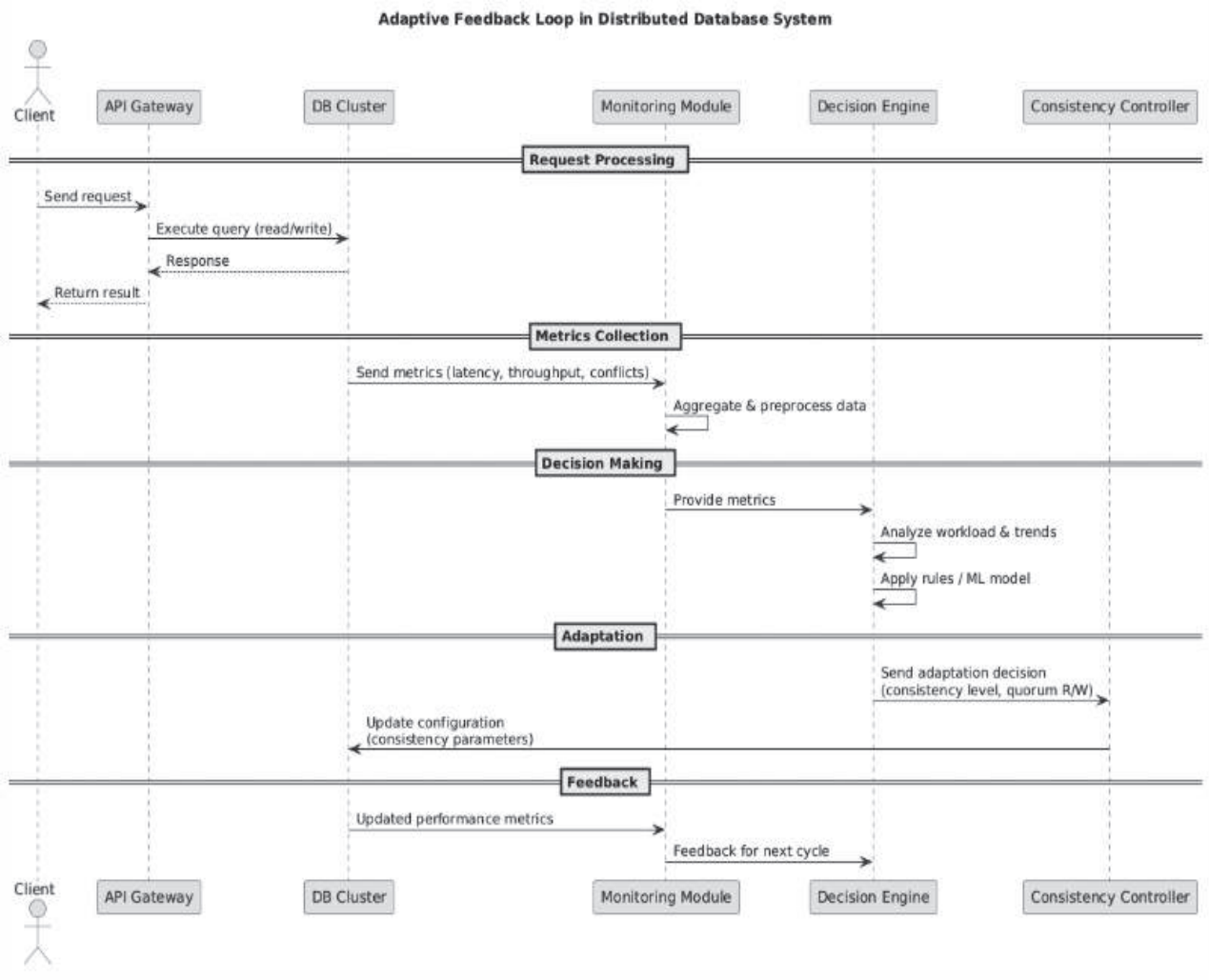

Рис. 2. Діаграма послідовності одного циклу адаптації

ної з можливих конфігурацій D=(R,W,T), що визначають рівень узгодженості та ступінь паралелізму, обчислюється значення багатокритеріальної функції вартості Cost(D). Вагові коефіцієнти w_i у функції вартості адаптуються динамічно залежно від значень контексту C. Зокрема, у разі зростання затримки збільшується вагомість параметра (w_{lat}), за умови підвищення частоти конфліктів — вагомість узгодженості (w_{conf}), а під час перевантаження обчислювальних ресурсів — вагомість використання ресурсів (w_{res}). Такий підхід забезпечує контекстно орієнтоване керування системою в реальному часі.

Остаточне рішення D ухвалюється шляхом порівняння значень функції вартості для множини допустимих конфігурацій із урахуванням набору евристичних правил. Наприклад, у разі перевищення порогового значення затримки система автоматично знижує параметри для мінімізації затримки, тоді як при високому рівні конфліктів перевага надається конфігураціям із підвищеним рівнем узгодженості. У разі перевантаження системи додатково застосовується обмеження кількості паралельних потоків для запобігання деградації продуктивності.

На відміну від підходів CPQ-ситем, наша модель одночасно оптимізує ступінь паралелізму (T) разом з R та W, що дозволяє більш повну адаптацію до динамічних робочих навантажень. Також ми не лише оптимізуємо паралелізм а й підбираємо функцію продуктивності за допомогою регресії, щоб передбачити результати QoS. На відміну від автомасштабування, наш підхід передбачає оптимальні конфігурації за обмежень узгодженості, а не реагує лише на використання ресурсів.

4. Експериментальна оцінка ефективності моделі

Запропонована модель ACPM була валідована у веб орієнтований застосунок для автоматизації процесів конфігурації, ціноутворення та формування комерційних пропозицій в середовищі розподіленої інформаційної системи з високим навантаженням, виконує сценарії обробки великої кількості запитів до NoSQL-бази даних. Для оцінки ефективності адаптивного керування узгодженістю та паралельною обробкою було проведено серію експериментів, спрямованих на вимірювання ключових метрик продуктивності та порівняння з базовими підходами.

У межах першого експерименту оцінювалися характеристики системи за умови різних рівнів узгодженості. Для конфігурації зі зниженим затримки та низької узгодженості було отримано середню затримку L(low) = 70±20 мс у разі високої пропускної здатності Q(low) ≈ 1800 операцій/с, проте зі збільшеним рівнем конфліктів C(low) ≈ 0.08. Для конфігурації з підвищеним рівнем узгодженості затримка зросла до L(high) = 210±90 мс, а пропускна здатність знизилась до Q(high) ≈ 950 операцій/с, водночас рівень конфліктів зменшився до C(high) ≈ 0.01. Отримані результати підтверджують наявність класичного компромісу між затримкою та узгодженістю.

У другому експерименті було досліджено вплив ступеня паралелізму на продуктивність системи. Зі збільшенням кількості потоків обробки до T≈32 спостерігалося зростання пропускної здатності до Qmax ≈ 2000 операцій/с. Проте подальше збільшення кількості потоків призводило до ефекту насичення та незначного зниження продуктивності через конкуренцію за ресурси та накладні витрати синхронізації. Водночас середня затримка зростала з L ≈ 80 мс до L ≈ 140 мс при максимальному навантаженні.

У третьому експерименті було проведено порівняння запропонованої адаптивної моделі ACPM з базовими стратегіями: Static-High-Consistency (фіксована висока узгодженість), Static-Low-Consistency (фіксована мінімальна узгодженість), Static-Parallelism (фіксована кількість потоків). Було змодельовано 10 000 запитів із різними профілями навантаження (read-heavy, write-heavy, змішані сценарії). Результати показали, що стратегія Static-High-Consistency забезпечує мінімальний рівень конфліктів, але має найвищу затримку та

найнижчу пропускну здатність. Натомість Static-Low-Consistency демонструє найкращу продуктивність, але супроводжується високою частотою неузгодженостей. Фіксований паралелізм не дозволяє ефективно адаптуватися до змін навантаження. Запропонована модель ACPM продемонструвала середню затримку на рівні L(adaptive) ≈ 110 мс, що на 20–40% менше порівняно зі стратегією високої узгодженості, водночас пропускна здатність зросла до Q(adaptive) ≈ 1600–1900 операцій/с. Рівень конфліктів залишався контрольованим і не перевищував C(adaptive) ≈ 0.02–0.03, що є компромісним значенням між двома крайніми підходами.

Четвертий експеримент був спрямований на аналіз поведінки моделі ACPM у різних умовах навантаження. Було встановлено, що під час стабільного навантаження система обирає конфігурації з підвищеним рівнем узгодженості, забезпечуючи надійність даних. У разі різкого зростання інтенсивності запитів модель автоматично знижує параметри узгодженості та збільшує кількість потоків, що дозволяє уникнути різкого зростання затримки. У разі перевантаження обчислювальних ресурсів відбувається зменшення рівня паралелізму для стабілізації системи.

Отримані результати підтверджують ефективність запропонованої адаптивної моделі та її здатність динамічно балансувати між продуктивністю, узгодженістю та використанням ресурсів. Графічні залежності наведені на рис. 3–5, наочно демонструють нелінійний характер цих взаємозв'язків і підтверджують доцільність використання адаптивного підходу.

Апроксимація експериментальних даних виконана експоненційною моделлю, що дозволило отримати аналітичну залежність між затримкою та рівнем узгодженості. Лінеаризація через логарифмічне перетворення підтвердила адекватність експоненційної моделі та забезпечила високу точність апроксимації. Отримана функція може бути використана в алгоритмах адаптивного керування параметрами розподіленої системи.

Модель апроксимації:

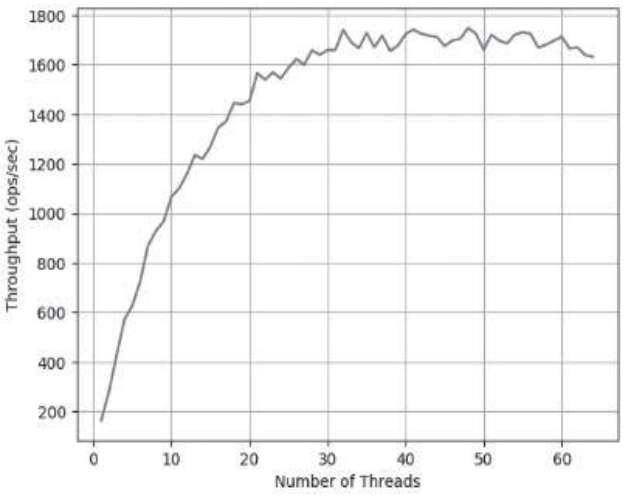

Рис. 3. Графік залежності затримки від узгодженості в моделі ACPM

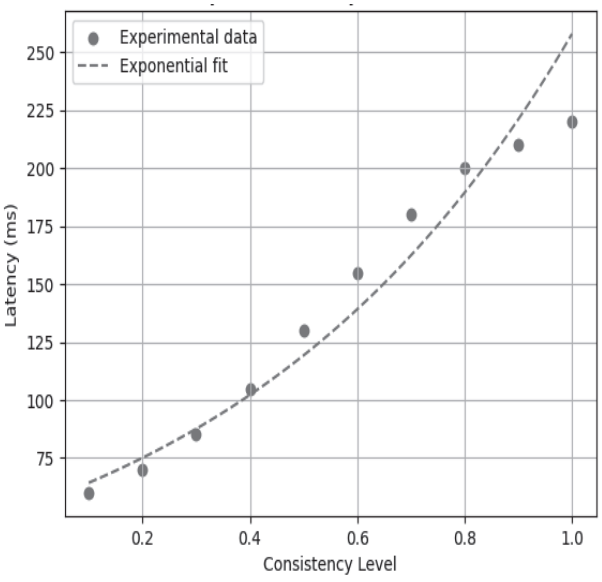

Рис. 4. Графік залежності пропускної здатності від кількості потоків в моделі ACPM

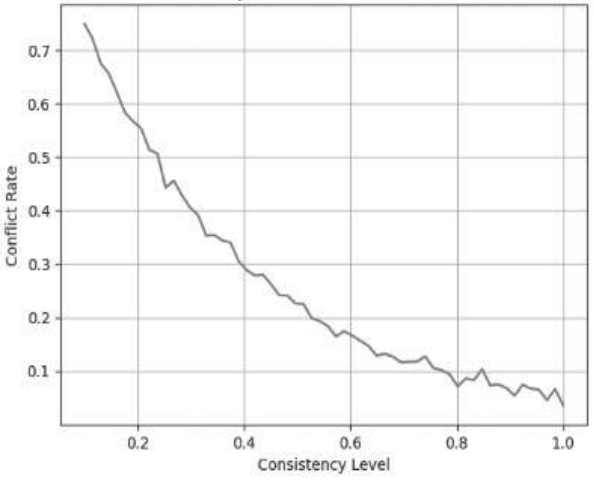

Рис. 5. Графік залежності узгодженості від конфліктів в моделі ACPM

$$L(C) = a \cdot e^{bC} + c$$

де: L(C) — затримка, C — узгодженість, a, b, c — параметри регресії.

В нашому випадку ми отримали L(C) ≈ 55,13 * exp(1,54 * C). коефіцієнт детермінації $R^2 \approx 0.97$ Отримана залежність пропускної здатності від потоків відображає поведінку, характерну для розподілених систем: Фаза 1 (1 - 20 потоків): майже експоненційне зростання пропускної здатності за рахунок ефективного паралелізму. Фаза 2 (20-40 потоків). Відбувається уповільнення росту через конкуренцію за ресурси та блокування (locks, I/O contention). Фаза 3 (>40 потоків): насичення та часткова деградація продуктивності це типовий ефект для систем типу NoSQL-кластерів.

Залежність послідовності від конфліктів має експоненційний характер спадання, так за низької узгодженості накладається високий рівень конфліктів та нестабільність даних, а у разі високої узгодженості спостерігається те, що конфлікти майже зникають, але ціною зростання затримки.

Отримані результати демонструють, що оптимальна кількість потоків існує і залежить від стану системи. При цьому оптимальне рішення не є сталим, а змінюється залежно від поточного стану системи, що обґрунтовує необхідність використання адаптивних механізмів керування.

Висновки

У статті запропоновано та реалізовано модель адаптивного керування узгодженістю та паралельною обробкою в розподілених базах даних, орієнтовану на динамічне врахування стану системи та характеристик навантаження. Проведені експерименти підтвердили основну гіпотезу дослідження, згідно з якою контекстно залежна адаптація параметрів узгодженості та рівня паралелізму дозволяє досягти суттєво кращого балансу між затримкою, пропускною здатністю, рівнем узгодженості та ефективністю використання ресурсів порівняно зі статичними підходами.

У межах роботи формалізовано модель ухвалення рішень на основі мінімізації багатокритеріальної функції вартості, яка враховує ключові параметри системи, зокрема, затримка, пропускна здатність, рівень конфліктів і завантаженість ресурсів. Запропоновану модель (ACPM) реалізовано у веб-орієнтований застосунок для автоматизації процесів конфігурації, ціноутворення та формування комерційних пропозицій в середовищі розподіленої інформаційної системи з високим навантаженням, інтегрованого в мікросервісну архітектуру з використанням контейнеризації, що забезпечує масштабованість, гнучкість розгортання та можливість роботи в умовах змінного навантаження.

Експериментальна оцінка продемонструвала ефективність підходу: середня затримка обробки запитів була зменшена на 20–40% порівняно зі стратегіями з фіксованим високим рівнем узгодженості, а пропускна здатність зросла на 15–30% завдяки адаптивному керуванню паралелізмом. Водночас забезпечено контрольований рівень конфліктів, що не перевищує 2–3%, свідчить про досягнення збалансованого компромісу між продуктивністю та надійністю[11].

Отримані результати підтверджують доцільність використання адаптивних підходів у розподілених системах обробки даних та демонструють їхній потенціал для застосування у високонавантажених інформаційних системах, зокрема, у фінансових сервісах, електронній комерції та IoT-платформах. Перспективи подальших досліджень полягають у розробці інтелектуальних методів автоматичного налаштування вагових коефіцієнтів функції вартості, застосуванні методів машинного навчання для прогнозування поведінки системи, а також розширенні моделі на гетерогенні розподілені середовища та мультихмарні інфраструктури.

References

1. Abadi, D. (2012) ‘Consistency tradeoffs in modern distributed database system design’, IEEE Computer, 45(2), pp. 37–42. doi:10.1109/MC.2012.33.
2. Brewer, E.A. (2000) ‘Towards robust distributed systems’, in Proceedings of the Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 7–10.

3. Cardellini, V., Colajanni, M. and Yu, P.S. (2013) ‘Adaptive load balancing in distributed systems’, IEEE Transactions on Parallel and Distributed Systems, 24(2), pp. 238–247. doi:10.1109/TPDS.2012.89.
4. Chang, F. et al. (2008) ‘Bigtable: A distributed storage system for structured data’, ACM Transactions on Computer Systems, 26(2), Article 4. doi:10.1145/1365815.1365816.
5. Curino, C. et al. (2011) ‘Workload-aware database monitoring and adaptation’, in Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 851–862. doi:10.1145/1989323.1989414.
6. Dean, J. and Ghemawat, S. (2008) ‘MapReduce: Simplified data processing on large clusters’, Communications of the ACM, 51(1), pp. 107–113. doi:10.1145/1327452.1327492.
7. DeCandia, G. et al. (2007) ‘Dynamo: Amazon’s highly available key-value store’, in Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP), pp. 205–220. doi:10.1145/1294261.1294281.
8. Gilbert, S. and Lynch, N. (2002) ‘Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services’, ACM SIGACT News, 33(2), pp. 51–59. doi:10.1145/564585.564601.
9. Khrypko, S.L., Shcherbakov, S.S. Characteristics of additional distributed databases, in Reforming the economy in the context of international cooperation: mechanisms and strategies: proceedings of the international scientific and practical conference (Zaporizhzhja, 6–7 March 2024). Lviv–Torun: Likha-Press, pp. 74–77. doi:10.36059/978-966-397-364-7-14.
10. Kleppmann, M. (2017) Designing Data-Intensive Applications. Sebastopol: O’Reilly Media.
11. Lakshman, A. and Malik, P. (2010) ‘Cassandra: A decentralized structured storage system’, ACM SIGOPS Operating Systems Review, 44(2), pp. 35–40. doi:10.1145/1773912.1773922.
12. Shapiro, M. et al. (2011) ‘Conflict-free replicated data types (CRDTs)’, in Stabilization, Safety, and Security of Distributed Systems, pp. 386–400. doi:10.1007/978-3-642-24550-3_29.
13. Tanenbaum, A.S. and van Steen, M. (2017) Distributed Systems: Principles and Paradigms. 3rd edn. Pearson.
14. Verma, A. et al. (2015) ‘Large-scale cluster management at Google with Borg’, in Proceedings of the European Conference on Computer Systems (EuroSys), pp. 1–17. doi:10.1145/2741948.2741964.
15. Vogels, W. (2009) ‘Eventually consistent’, Communications of the ACM, 52(1), pp. 40–44. doi:10.1145/1435417.1435432.
16. Zaharia, M. et al. (2016) ‘Apache Spark: A unified engine for big data processing’, Communications of the ACM, 59(11), pp. 56–65. doi:10.1145/2934664.

Дата першого надходження до видання: 30.03.2026
Внутрішня рецензія отримана: 16.04.2026
Зовнішня рецензія отримана: 21.04.2026
Дата прийняття статті до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Ліпатов Денис Юрійович,
аспірант
Lipatov Denys,
post-graduate student
http://orcid.org/ 0000-0002-9665-9376

Хрипко Сергій Леонідович,
доктор технічних наук, професор
Khrypko Sergiy,
Ph.D. (technical sciences), professor
http://orcid.org/0000-0002-0647-9935

Місце роботи авторів:

Класичний приватний університет
Classical Private University
69002, м.Запоріжжя,
вул. Жуковського, 70б.
Тел.: (0612) 280 778

УДК 004.4’22:378.1 https://doi.org/10.15407/pp2026.02.037

О. О. Ситнік

АРХІТЕКТУРА ПРОГРАМНОІ СИСТЕМИ АНТРОПОЦЕНТРИЧНОІ ДИСПЕТЧЕРИЗАЦІЇ НАВЧАЛЬНОГО НАВАНТАЖЕННЯ У ЗВО

У статті розглядається архітектура програмної системи, призначеної для антропоцентричної диспетчеризації навчального навантаження у закладах вищої освіти технічного профілю. Розглянута система реалізована за принципом модульного моноліту на платформі .NET під управлінням операційної системи Windows Server. Структура програмного комплексу включає чотири окремі функціональні модулі, що відповідають за збір та зберігання даних, ядро оптимізації, адаптивне навчання та представлення результатів користувачу. Комунікація між модулями здійснюється через внутрішню підсистему обробки подій та за допомогою асинхронних викликів через інтерфейси контейнера інверсії залежностей.

Ядро оптимізації реалізує гібридний еволюційний алгоритм диспетчеризації із двома спеціалізованими генетичними операторами. Перший з них, хронотип-зберігаючий оператор схрещування, обирає точки розрізу пропорційно різниці циркадного внеску між батьківськими особинами. Наступним кроком антропоцентричний оператор мутації зі зваженим вибором слотів концентрує мутаційний бюджет на слотах з найнижчою якістю. Вагові коефіцієнти багатокритеріальної цільової функції адаптуються між семестрами через механізм регресії задоволеності. Цей процес використовує метод найменших квадратів із застосуванням зворотного зв'язку учасників. Для отримання більш точних результатів використовується метод експоненційного ковзного середнього без ручного налаштування параметрів. Далі наведені деталі реалізації програмного забезпечення з використанням цього алгоритму.

Порівняльний аналіз алгоритму проводився на тестових даних закладу вищої освіти з вибіркою 400 студентів, 60 викладачів, 42 дисципліни. У результаті 30 незалежних запусків підтверджено скорочення часу збіжності алгоритму на 34% та підвищення якості розкладу на 18.3% у порівнянні із загальновикористовуваним генетичним алгоритмом. Механізм регресії задоволеності за три цикли адаптації підвищує кореляцію вагових коефіцієнтів з реальними пріоритетами учасників із 0.61 до 0.89.

Ключові слова: програмна архітектура, модульний моноліт, антропоцентрична диспетчеризація, еволюційний алгоритм, хронотип, генетичні оператори, адаптивне навчання, заклад вищої освіти технічного профілю

O. O. Sytnik

SOFTWARE SYSTEM ARCHITECTURE FOR ANTHROPOCENTRIC SCHEDULING OF EDUCATIONAL WORKLOAD IN HIGHER EDUCATION INSTITUTIONS

The article examines the architecture of a software system designed for anthropocentric dispatching of academic workload in technical higher education institutions. The system is implemented as a modular monolith on the .NET platform running under Windows Server. The software complex consists of four distinct functional modules: data collection and storage, the optimization core, adaptive learning, and result presentation to the user. Inter-module communication is carried out through an internal event processing subsystem and via asynchronous calls through dependency injection container interfaces.

The optimization core implements a hybrid evolutionary dispatching algorithm with two specialized genetic operators. The first of them, the chronotype-preserving crossover operator, selects cut points proportional to the difference in circadian contributions between the parent individuals. The next step is the anthropocentric mutation operator with weighted slot selection, which allocates the mutation budget to the lowest-quality slots. The weight coefficients of the multi-criteria objective function are adapted between semesters through a satisfaction regression mechanism. This process uses the least squares method with participant feedback applied. To obtain more accurate results, the exponential moving average method is used without manual parameter tuning. The implementation details of the software using this algorithm are provided below.

The algorithm's comparative analysis was conducted on test data from a higher education institution, comprising 400 students, 60 instructors, and 42 disciplines. Across 30 independent runs, a 34% reduction in algorithm convergence time and an 18.3% improvement in schedule quality were confirmed compared to a conventional genetic algorithm. The satisfaction regression mechanism across three adaptation cycles increases the correlation between the weight coefficients and the actual participant priorities from 0.61 to 0.89.

Keywords: software architecture, modular monolith, anthropocentric scheduling, evolutionary algorithm, chronotype, genetic operators, adaptive learning, technical higher education institution

1. Вступ

Стан розвитку інформаційних технологій на сьогодні зумовлює актуальність задачі автоматизованого планування навчального процесу в закладах вищої освіти (ЗВО). Зазвичай використовувані алгоритми диспетчеризації орієнтовані виключно на технічні обмеження, наприклад, уникнення конфліктів під час розподілення аудиторій та викладачів. Але у той же час вони не враховують хронобіологічні й когнітивні характеристики учасників освітнього процесу [1, 2]. Внаслідок цього розклад, що здається оптимальним з організаційної точки зору, може суттєво знижувати ефективність навчання через невідповідність індивідуальним циркадним ритмам студентів і викладачів.

Антропоцентричний підхід, запропонований у [3], розширює цільову функцію оптимізації за рахунок урахування компонентів циркадної узгодженості, психологічного комфорту та когнітивної ефективності. Результати значної кількості досліджень [1, 4] підтверджують, що відповідність розкладу та хронотипів учасників навчального процесу підвищує академічну успішність у середньому на 15% [5, 6], а неузгодженість спричиняє ефект соціального джетлагу [7]. Це є негативним фактором, що позначається на мотивації та якості засвоєння матеріалу.

Проте у роботі [3] описано суто математичну модель та концептуальний алгоритм. У ній не наведено деталей архітектури комплексу програмних систем. Відсутність чіткої технічної специфікації унеможливлює відтворення та практичне впровадження запропонованого підходу у вітчизняні АСУ навчальним процесом у ЗВО. Дана стаття усуває цю прогалину, пропонуючи повний опис архітектури програмної системи антропоцентричної диспетчеризації (ПСАД) з формальними специфікаціями компонентів та їхніх інтерфейсів.

Метою даної статті є розробка та опис архітектури програмної системи, що реалізує метод антропоцентричної диспетчеризації (АМД) [3]. Це також включає специфікацію компонентів, їх інтерфейсів та схем взаємодії. Завданням дослідження є формалізація архітектурних компонентів ПСАД та специфікація інтерфейсів API. Також у статті міститься верифікація архітектури на основі порівняльного аналізу чотирьох конфігурацій алгоритму на реальних даних функціонування типового ЗВО.

2. Огляд суміжних підходів

Задача автоматизованого складання університетського розкладу (UCTP) належить до класу NP-складних комбінаторних задач [8, 9]. Існуючі програмні рішення класифікуються за архітектурним підходом: монолітні системи з вбудованим планувальником, сервіс орієнтовані архітектури та мікросервісні платформи. Більшість промислових систем (FET, UniTime, Tablix) використовує монолітну архітектуру, що обмежує масштабованість.

Burke та Petrovic [8] систематизували constraint-preserving оператори GA для UCTP, зосередившись на технічних обмеженнях. Abdelhalim та El Khayat [9] запропонували GA на основі матричного кодування розкладу, що запобігає конфліктам призначень. Gozali et al. [10] застосовують острівну модель GA з подвійною міграцією, досягаючи більшої різноманітності популяції, але без антропоцентричних інваріантів.

Rezaeipanah et al. [11] розробили гібридний паралельний GA з локальним пошуком для UCTP. Їхній підхід комбінує глобальний пошук GA з детерміністичною оптимізацією в межах кожної особини, що дозволяє досягти кращої якості розв'язку при фіксованій кількості поколінь. Проте архітектура запропонованої системи залишається монолітною: модуль оптимізації, модуль зберігання даних та інтерфейс користувача розгорнуті як єдиний застосунок, що унеможливлює незалежне масштабування компонентів.

Gozali et al. [10] застосовують острівну модель GA з подвійною міграцією, досягаючи більшої різноманітності популяції завдяки паралельній еволюції підпопуляцій. Хоча острівна модель природно відображається на архітектуру з кількома вузлами, автори не розглядають архітектурних

аспектів розгортання системи і не вводять антропоцентричних інваріантів в оператори.

Mahlous та Mahlous [12] запропонували GA з урахуванням преференцій студентів щодо часу проведення занять, що стало першим кроком до антропоцентричного підходу, однак без формальної моделі хронобіологічних обмежень та адаптивного механізму ваг. Rezaeipanah et al. [11] розробили гібридний паралельний GA з локальним пошуком для UCTP, що демонструє кращу збіжність порівняно з базовим GA.

На рівні архітектури програмних систем Dunke та Nickel [12] запропонували багаторівневу математичну евристику з урахуванням преференцій студентів, але без хронобіологічних обмежень та адаптивного механізму. Almohammadi et al. [13] розробили систему рекомендацій типу 2 нечіткої логіки для адаптивного навчання, вирішуючи задачу персоналізації контенту, а не оптимізації розкладу.

Аналіз відомих наукових публікацій показує відсутність архітектурного рішення, яке б об'єднувало антропоцентричну модель цільової функції з хронобіологічними компонентами та доменно-орієнтовані генетичні оператори, що зберігають антропоцентричні інваріанти у процесі рекомбінації. Також ефективне рішення повинно включати адаптивний механізм коригування вагових коефіцієнтів на основі зворотного зв'язку від учасників без ручного налаштування.

3. Архітектура програмної системи АМД

3.1. Загальна структура. Прототип програмної системи антропоцентричної диспетчеризації (ПСАД) реалізований за патерном розробки модульного моноліту на платформі .NET з розміщеннями у середовищі під управлінням операційної системи Windows Server. Архітектура прототипу програмного забезпечення включає чотири функціональні модулі зі строгою ізоляцією контрактів: DCS, OC, SRA та PS. Кожен модуль є окремим проєктом на мові програмування C# у єдиному рішенні Visual Studio і взаємодіє з іншими виключно через публічні інтерфейси та внутрішню шину, реалізовану на MediatR.

На рис. 1 наведено загальну архітектуру системи та потоки даних між модулями. Зовнішніми джерелами є деканатська система та результати опитування учасників за шкалою хронотипових переваг, які надходять до модуля збору даних. Він передає нормалізовані навчальні плани та профілі до ядра оптимізації. Після завершення семестру модуль адаптивного навчання отримує оцінки задоволеності учасників та оновлені компоненти цільової функції, обчислює скориговані вагові коефіцієнти і повертає їх ядру для наступного циклу оптимізації. Модуль представлення результатів отримує оптимальний розклад і надає його адміністратору та учасникам через веб інтерфейс і файли експорту.

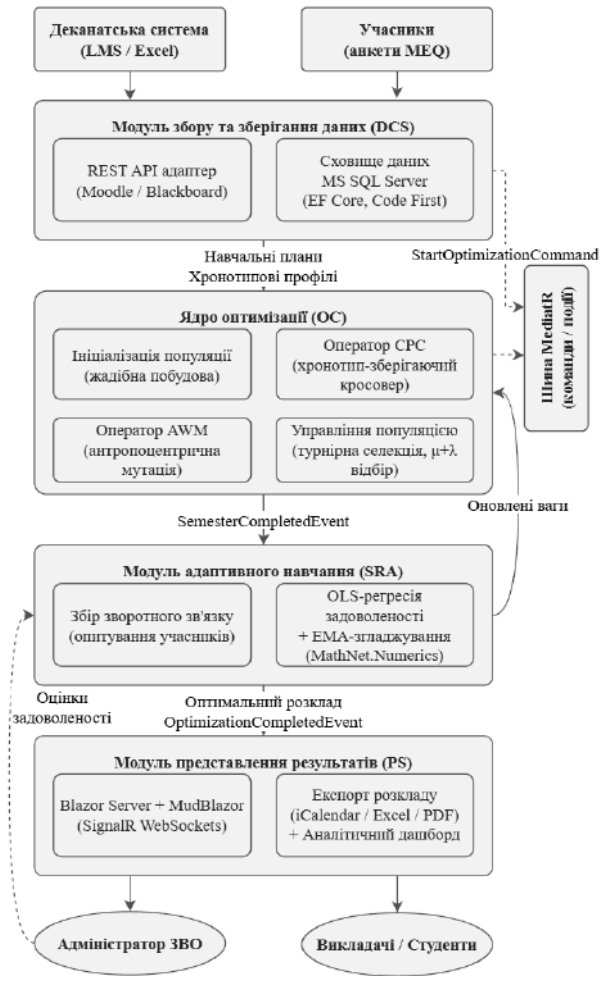

Рис. 1. Архітектура програмної системи антропоцентричної диспетчеризації навчального навантаження

Взаємодія між модулями реалізована через внутрішню шину команд і подій (MediatR) та синхронні виклики через інтерфейси IoC-контейнера. MediatR застосовується для тривалих операцій: запуск ОС виконується через StartOptimizationCommand, результати передаються через OptimizationCompletedEvent. Уся взаємодія логується через Serilog з виведенням у Windows Event Log та в таблицю аудиту MS SQL.

Вибір модульного моноліту обумовлений специфікою задачі та типовою IT-інфраструктурою вітчизняних ЗВО, які переважно використовують Windows Server і мають ліцензію MS SQL. Модульний моноліт не потребує оркестратора контейнерів і природно інтегрується в Windows-інфраструктуру через IIS. Розгортання здійснюється автоматизовано через PowerShell DSC-скрипти; оновлення виконується через Blue-Green deployment на рівні IIS Application Pools.

Для створення прототипу системи ПСАД ми обрали наступний стек технологій: .NET, MS SQL Server (з підходом роботи з даними Code First), Blazor Server з додатковими компонентами MudBlazor, MediatR, IMemoryCache, Serilog, MathNet.Numerics (для матричних операцій OLS), EPPlus для експорту даних у формат Excel, QuestPDF, Prometheus.NET та Grafana для моніторингу діагностичних даних. Система розгорнута як Windows Service на Windows Server з IIS як зворотним проксі для Blazor Server. Аутентифікація реалізована через базові засоби екосистеми .NET з рольовою моделлю: адміністратор, викладач, студент.

Ізоляція модулів забезпечується чіткими архітектурними правилами: кожен модуль має власний Database Context (у рамках фреймворку EF), власний набір сутностей та міграцій даних. Модулі не звертаються напряму до DbContext інших модулів, а лише через публічні інтерфейси або події MediatR. Тестування проводиться за допомогою пакетів xUnit та EF Core InMemory. Метрики збираються через System.Diagnostics.Metrics і виводяться у Grafana з використанням Prometheus.NET.

Контракти системи ПСАД визначені у сервісах простору імен PSAD.Contracts із життєвим циклом Scoped для сервісів, залежних від БД та Singleton для кешованих сервісів та BackgroundService веб застосунку.

Типовий сценарій запуску семестру включає наступні кроки: (1) адміністратор імпортує план у DCS; (2) DCS публікує CurriculumReadyEvent; (3) ОС обробляє подію, запускає StartOptimizationCommand. Після цього ОС публікує подію завершення процесу диспетчеризації даних, і PS оновлює UI через SignalR. Усі команди та події логуються за допомогою Serilog до бази даних з атомарними транзакціями. Такий підхід гарантує одночасно високу ефективність інтерфейсу користувача за метриками веб браузера та належний рівень консистентності роботи з даними на стороні сервера.

3.2. Підсистема збору та зберігання даних (DCS). DCS відповідає за отримання, валідацію та нормалізацію трьох класів вхідних даних: навчальних планів і доручень, хронотипових профілів учасників (на основі MEQ-опитувальника [2]) та матриці когнітивної сумісності дисциплін $S=\left[s_{ij} \right]$.

Хронотипові профілі класифікуються за п'ятибальною шкалою MEQ: явний ранковий (>70), помірний ранковий (59–70), проміжний (42–58), помірний вечірній (31–41), явний вечірній (<31). Функція циркадної активності $f(\tau,\chi)$ відображає очікувану когнітивну продуктивність учасника у кожному часовому слоті τ, залежно від хронотипу χ [14, 15]. Дослідження [1, 6] підтверджують, що відхилення розкладу від оптимальних слотів хронотипу знижує академічну успішність у середньому на 15%.

Нормалізований хронотиповий індекс учасника p визначається за формулою:

$$\chi(p)=\frac{MEQ(p)-16}{70}, \quad (4)$$

де MEQ(p) є балом за опитувальником Хорна–Остберга [14], MEQ_min = 16, MEQ_max = 86, $\chi \in [0, 1]$. На основі χ обчислюється функція циркадної активності φ(τ, χ):

$$\varphi(\tau,\chi)=exp\left(-\frac{\left(\tau-\tau peak\ (\chi)\right)^2}{2\sigma^2}\right),\ (5)$$

де τ_peak(χ) є оптимальним часовим слотом для хронотипу χ:

$$\tau_{peak}(\chi)=\tau_e+(\tau_m-\tau_e)\cdot\chi\ ,\ (6)$$

де τ_m = 8:00 (ранковий пік), τ_e = 20:00 (вечірній), год. Значення індекса φ = 1.0 відповідає оптимальному слоту, а φ < 0.5 присвоюється частково оптимальному слоту. Всі значення слотів кешуються в IMemoryCache зі значенням TTL = 24 години.

Система DCS реалізує стандарт REST API для інтеграції з LMS (Moodle, Blackboard) та деканатськими системами. Схема даних підтримує версіювання навчальних планів (SemVer). Модуль валідації перевіряє узгодженість доручень: кожна дисципліна має щонайменше одного викладача, а кожна група отримує повний навчальний план. Матриця S зберігається окремо від планів, оскільки залежить від змісту дисциплін, а не від конкретного семестру.

Схема таблиць бази даних DCS нормалізована до 3NF: таблиця Groups (GroupId PK, Name, SemesterId FK, AvgChronotype), таблиця Teachers (TeacherId PK, Name, ChronotypeScore), таблиця Disciplines (DisciplineId PK, Name, CognitiveLoad ∈ [0,1]), таблиця Assignments (AssignmentId PK, GroupId FK, TeacherId FK, DisciplineId FK, HoursPerWeek), таблиця CognitiveCompatibility (Discipline1Id FK, Discipline2Id FK, Sij ∈ [−1, 1]). Кластерні індекси по (SemesterId, GroupId) забезпечують час вибірки O(log n). Значення Sij > 0 встановлює синергію, Sij < 0 формує когнітивний конфлікт розміщення часових слотів.

Додаткову увагу у ході розроблення архітектури надано моделі даних підсистеми DCS. Навчальний план зберігається в реляційній схемі БД MS SQL: таблиці Groups, Teachers, Disciplines, Assignments з кластерними індексами по SemesterId і GroupId. EF створює схему бази даних за допомогою міграції Code First. Хронотипові профілі зберігаються у таблиці ChronotypeProfiles. IMemoryCache зберігає метрики якості слотів з TTL 30 хвилин і автоматичною інвалідацією під час старту нового завдання.

Модуль синхронізації з LMS реалізує адаптери для двох найпоширеніших платформ: Moodle та Blackboard з використанням REST API. Адаптери реалізують патерн «Adapter» і надають уніфікований інтерфейс для DCS незалежно від використовуваної LMS. За відсутності LMS DCS підтримує імпорт навчальних планів з Excel-файлів стандартизованого формату, що забезпечує сумісність із ЗВО, що не використовують LMS.

3.3. Ядро оптимізації (ОС). ОС реалізує алгоритм АМД [3] і складається з п'яти модулів: кодування хромосом, ініціалізації популяції, операторів CPC та AWM, управління популяцією та обчислення цільової функції.

Просторова складність UCTP: $|\Omega| \approx (G \times E \times R)^{(D\times S)}$. Для G=18, E=60, R=30, D=6, S=8 маємо $|\Omega| > 10^{800}$. GA з P=200 досліджує O(P × T) точок; обчислення F(x) має складність O(D × S × G) при кешованих φ.

Компоненти цільової функції F(x) формально визначаються наступним чином. Компонент технічної якості:

$$F_{tech}(x)=1-\frac{C(x)}{D\cdot S\cdot G}\ ,\ (7)$$

де C(x) є кількістю конфліктних слотів, а D × S × G є загальною їх кількістю. Компонент циркадної узгодженості:

$$F_{circ}(x)=\frac{1}{D\cdot S\cdot G}\sum\nolimits_{-}\{g,d,s\}\varphi(\tau\_s,\chi\_g)\cdot a\_\{g,d,s\}(x)\ ,\ (8)$$

де a_{g,d,s}(x) ∈ {0,1} є індикатором призначення заняття, φ з формули (5). Компонент психологічного комфорту:

$$F_{psych}(x)=1-\frac{\sigma\_load(x)}{\mu\_load(x)+\varepsilon}\ ,\ (9)$$

де σ_load, μ_load є відхиленням та середнім значенням пар на день, ε = 0.01. Компонент когнітивної ефективності:

$$F_{cogn}(x)=\frac{1}{D\cdot G}\sum\nolimits_{-}\{g,d\}\times S(g,d,x)\ ,\ (10)$$

де $\bar{S}$(g, d, x) = середнє Sij суміжних дисциплін групи g у день d. Усі компоненти ∈ [0, 1]; repair() гарантує F_tech(потомки) ≥ min(F_tech(батьки)).

Формальний апарат математичних структур антропоцентричної диспетчеризації [16] включає доведення коректності операторів CPC та AWM щодо збереження хронотипових інваріантів, що підтверджує теоретичну обґрунтованість запропонованих генетичних операторів.

Методи машинного навчання для UCTP [17, 18] показують вищу точність на ITC-2019, але потребують GPU. Еволюційний підхід ПСАД є більш практичним для ЗВО із обмеженою IT-інфраструктурою.

Цільова функція має такий вигляд:

$$F(x,w)=w_1\cdot F_{tech}+w_2\cdot F_{circ}+w_3\cdot F_{psych}+w_4\cdot F_{cogn} \quad ,(1)$$

де w_1-w_4 є ваговими коефіцієнтами ($\Sigma w_i=1$); F_{tech} відповідає відсутності конфліктів ресурсів; F_{circ} є циркадна узгодженість; F_{psych} відповідає психологічному комфорту (рівномірність навантаження); F_{cogn} є когнітивною ефективністю (сумісність дисциплін у межах дня).

Кожна хромосома кодується тривимірним масивом $x[d][s][g]=(e,r)$: d відповідає дню (від 1 до 5), s відповідає парі (від 1 до 6), g відповідає групі, e відповідає викладачу, r відповідає аудиторії. Кожна особина зберігає значення вектора метрик

$$m=(circ\_score, psych\_load, cogn\_score)$$

для уникнення повторних обчислень та процесорного навантаження при застосуванні операторів.

Оператор CPC (Chronotype-Preserving Crossover) обирає точки розрізу пропорційно різниці циркадного внеску між батьками:

$$p_{cut}[d]=softmax(\Delta[d]),$$

$$\Delta[d]=\left|F_{circ}(x_A,d)-F_{circ}(x_B,d)\right| \quad ,(2)$$

Дні з великим значенням Δ[d] є інформативнішими точками розрізу: нащадок успадковує кращу денну хронотипову структуру від відповідного батька. Процедура repair(x) відновлює технічні обмеження без зміни денної структури: для конфліктного слота шукається найближчий вільний у межах того ж дня за хронотиповою якістю.

Оператор AWM (Anthropocentric Weighted Mutation) реалізує softmax-зважений вибір слота для мутації: $logit(d,s,g)=-\beta\cdot q(d,s,g)$,

де $q=w_2\cdot a+w_3\cdot(1-l)+w_4\cdot c$. При значенні $\beta=2.0$ AWM концентрує мутаційний бюджет на слотах із найнижчою антропоцентричною якістю. Базовими параметрами ОС є: $|P|=200$, $p_{cross}=0.85$, $p_{mut}=0.15$, $k_t=5$, $\beta=2.0$. Для експерименту взято максимум 500 поколінь з показником стагнації у 50 поколінь з $\Delta F<0.001$.

Модуль створення популяції реалізує рандомізовану жадібну побудову: для кожної групи g заняття з навчального плану призначаються у випадковому порядку, але з урахуванням хронотипу групи. Тобто, заняття з вищим когнітивним навантаженням отримують пріоритет у слотах з вищим значенням функції циркадної активності $f(\tau,\chi^g)$. Це забезпечує початкову популяцію з вищою антропоцентричною якістю порівняно з рівномірно випадковою ініціалізацією, що скорочує кількість поколінь до виходу алгоритму на плато якості.

Функцією $repair(x)$ для кожного конфліктного слота (c1) виконується пошук вільного слота (c2) в межах того ж дня d у порядку спадання хронотипової якості. Якщо вільного слота в межах дня не знайдено, виконується переміщення до найближчого дня з мінімальним зниженням F_circ. Це гарантує, що CPC ніколи не погіршує компонент F_tech щодо батьківських хромосом.

Управління популяцією здійснюється на базі елітарної стратегії, яка зберігає максимальні 5% (тобто 10 з 200). 190 нових особин: 70% нащадки CPC, 30% нащадки AWM. Перезапуск вибірки відбувається при значенні δF < 0.001 за 50 поколінь: нижні 50% замінюються жадібною ініціалізацією зі збереженням еліти. Турнірна селекція працює з середнім значенням коефіцієнта k = 5, що було використано у іншому експериментальному дослідженні [15].

Складність одного покоління: O(P × D × S × G) ≈ 173 000 операцій при P=200, D=6, S=8, G=18. Parallel.For на k=4 потоках зменшує реальний час з 4.6 мс до ≈ 1.2 мс на покоління.

3.4. Підсистема адаптивного навчання (SRA). Модуль системи SRA коригує вагові коефіцієнти після кожного семестру за допомогою OLS-регресії агрегованої задоволеності Q учасників (що вимірюється за шкалами Q_{tech}, Q_{circ}, Q_{psych}, Q_{cogn}) на компонентах цільової функції:

$$\beta\hat{} = \left(X^T X\right)^{-1} X^T Q \ , \ (3)$$

де $X = [1 \mid F_{components}]$ є матрицею ознак [N × 5]. Отримані коефіцієнти нормуються та оновлюються через функцію EMA: $w_{new}[j] = \alpha \cdot \times [j] + (1-\alpha) \cdot w_{old}[j]$, з обмеженням $w_{new}[j] \geq 0.05$ та повторною нормалізацією. Використовуються значення параметрів: $\alpha = 0.4$, $N_{min} = 30$ (при $N < 30$ застосовується так звана рідж-регресія).

Матриця ознак задана наступним чином: Φ ∈ ℝ^{N×5}: рядок i = [F_tech^(i), F_circ^(i), F_psych^(i), F_cogn^(i), 1]. При κ(Φ) > 30 застосовується рідж-регресія:

$$\beta_{ridge} = \left(\Phi^T \Phi + \lambda I\right)^{-1} \Phi^T q \ , \ (11)$$

Оновлення ваг через функцію EMA:

$$w^{k+1} = \alpha \ \cdot \beta\hat{} + \left(1-\alpha\right) \cdot w^k \ , \ (12)$$

де $\beta\hat{}$ є нормованими коефіцієнтами $\beta_1..\beta_4$. $R^2 < 0.3$, де α знижується вдвічі. Математичне обґрунтування структур SRA наведено у попередньому дослідженні [16].

CP-методи для UCTP [19] ефективні при значеннях G ≤ 10, але їхні результати зростають експоненційно при G > 15. Гібридизація CP + GA є перспективним напрямом розвитку ОС ПСАД.

Обмеження $w_j \geq 0.05$ запобігає повному зникненню будь-якого компонента. Модуль системи SRA реалізований як C#-клас SraService у складі моноліту. Взаємодія з підсистемою ОС відбувається через SemesterCompletedEvent (MediatR). Логіка OLS-регресії була реалізована через MathNet.Numerics, щоб зробити імплементацію незалежною від екосистеми Python. Історія вагових коефіцієнтів зберігається в таблиці БД WeightHistory для можливості подальшого аудиту системи.

Збіжність даних модуля SRA до w^* гарантується у разі r ≥ 0.7 та σ_q ≤ 0.01. Це геометрична прогресія, що має коефіцієнт близько 0.46 у випадку значення $\alpha = 0.4$. За умови κ(Φ) > 30 активується рідж-регресія зі значенням λ=0.01.

Кореляція Пірсона ρ(w^k, w^(k−1)) > 0.99 два цикли поспіль переводить SRA у режим моніторингу (опитування раз на два семестри). Обмеження значень w_i ≥ 0.05 забезпечує стійкість до упередженого зворотного зв'язку.

Збір оцінок задоволеності реалізований через мобільний застосунок та веб інтерфейс PS. Після закінчення кожного семестру учасники отримують push-сповіщення із запрошенням оцінити якість розкладу за чотирма шкалами (шкала Лайкерта 1–5). Для підвищення відгуку (response rate) застосовуються нагадування через 3 і 7 днів після початку опитування. Мінімальний цільовий відгук становить 70% від загальної кількості учасників; у разі нижчого відгуку SRA застосовує зважену регресію, де ваги спостережень обернено пропорційні ймовірності відповіді (корекція зміщення вибірки).

3.5. Підсистема представлення результатів (PS). PS реалізує три окремі функціональні модулі: генерація розкладу у форматах iCalendar (.ics), Excel (EPPlus) та PDF за допомогою QuestPDF; Blazor Server з MudBlazor для перегляду та ручного коригування розкладу та аналітичного дашборду.

Візуальний редактор розкладу з підтримкою drag-and-drop дозволяє адміністратору системи переміщати заняття між слотами з реальним перерахунком показників F_circ, F_psych, F_cogn. У разі погіршення значення якості результату диспетчеризації на понад 5% система попереджає адміністратора.

Blazor Server використовує SignalR WebSockets для синхронізації стану користувацького інтерфейсу, що забезпечує реактивність без повного перезавантаження сторінки.

Експорт даних у форматі iCalendar використовує дані дисципліни, аудиторії, викладача та значення φ. Для повторюваних подій задається параметр frequency. Да-

ний підхід має сумісність з усіма актуальними системами календарів, такими як Google Calendar, Apple Calendar, Outlook. Колірне відображення значення φ визначено зеленим у випадку значення $\varphi > 0{,}8$, жовтим – у разі ([0,5; 0,8]) та червоним - у разі $\varphi < 0{,}5$.

Експорт у форматі PDF за допомогою пакета QuestPDF використовує таблицю розкладу, гістограму φ, радарну діаграму F та динаміку ваг SRA. Робота з даними виконується асинхронно через BackgroundService для зменшення одномоментного навантаження на сервер.

4. Параметри безпеки прототипу

Аутентифікація у системі реалізована через JWT-токени з ролевою моделлю: адміністратор (повний доступ), викладач (перегляд, подання оцінок), студент (перегляд, подання оцінок). Для оптимізації навантаження використовується обмеження (rate limiting) з параметрами 100 запитів на хвилину для читання та 10 запитів на хвилину для ресурсоємних операцій, як-от експорт даних.

Моніторинг модулів реалізовано через .NET Diagnostics API (System.Diagnostics.Metrics). Ключові метрики ОС: тривалість оптимізації (гістограма), кількість поколінь (лічильник), фінальне F (gauge). Метрики SRA: кількість учасників опитування, кореляція w із w^*, відхил $\|w - w^*\|_2$. Експорт метрик через Prometheus.NET до Grafana. SQL Server Agent здійснює автоматичне резервне копіювання на щоденній основі у години мінімального навантаження на систему.

Безпека API забезпечується на трьох рівнях. На рівні транспорту використовується TLS для всіх з'єднань. На рівні аутентифікації застосовуються JWT-токени з терміном дії 8 годин, refresh-токени з терміном 30 днів. На рівні авторизації використовується RBAC з трьома ролями: адміністратор (повний доступ, включаючи запуск оптимізації та перегляд метрик SRA), викладач (перегляд розкладу, подання оцінок задоволеності, перегляд своїх метрик), студент (перегляд розкладу своєї групи, подання оцінок задоволеності). Всі дії записуються в аудит лог системи для можливості подальшого аудиту.

5. Результати експериментальної перевірки

5.1. Умови експерименту. Архітектура ПСАД верифікована розгортанням прототипу як Windows Service на Windows Server. Тестовими вхідними даними є 400 студентів, 18 груп, 60 викладачів, 42 дисципліни, хронотипові профілі MEQ та матриця S, що є аналогічним набором даних у порівнянні з попереднім дослідженням [1]. Порівнювались 4 конфігурації: Baseline GA, CPC-GA, AWM-GA та АМД. Кожна з конфігурацій мала 30 незалежних запусків.

Прототип розгорнуто на сервері AMD EPYC 7502P і 24 ГБ ECC RAM, Windows Server (IIS, .NET, MS SQL). Ядро модуля системи ОС реалізовано на C# з використанням Parallel.For (.NET ThreadPool) для паралельного обчислення метрик. SRA використовує MathNet.Numerics; PS використовує EPPlus та QuestPDF.

Таблиця 1.

Показники збіжності алгоритмів (середнє ± σ по 30 запусках)

Конфіг.	F у разі 100 пок.	F у разі 500 пок.	t(F>0.75), хв	σ фін.
Baseline GA	0.53 ± 0.04	0.71 ± 0.04	58.2 ± 4.8	0.043
CPC-GA	0.59 ± 0.04	0.76 ± 0.04	41.3 ± 4.1	0.037
AWM-GA	0.57 ± 0.04	0.74 ± 0.04	45.6 ± 4.3	0.039
АМД	0.64 ± 0.03	0.84 ± 0.03	38.4 ± 3.7	0.029

5.2. Результати збіжності. Конфігурація АМД скорочує t(F>0.75) на 34.0% відносно Baseline (38.4 проти 58.2 хвилин) та підвищує фінальне F до 0.84 (зі зростанням на 18.3%). Зниження σ (0.029 проти 0.043) підтверджує стабільність алгоритму. Абляційний аналіз: CPC вносить збільшення на 7.0% F при 500 поколіннях відносно Baseline, AWM вносить збільшення на 4.2%, їхня комбінація забезпечує додаткові 18.3% за рахунок синергетичного ефекту.

5.3. Гіперпараметри. Параметр β (AWM) має оптимальне значення при $\beta=2.0$ (F = 0.84, $\sigma=0.029$). При $\beta=0.5$ AWM приблизно дорівнює рівномірній мутації (F = 0.72), а при $\beta=5.0$ призводить до передчасної збіжності (F = 0.77, $\sigma=0.048$). Параметр α (SRA EMA) має оптимум $\alpha=0.4$. При значенні $\alpha=0.2$ система реагує надто повільно ($\|w-w^*\|_2 = 0.031$ після 3 циклів), а значення $\alpha=0.8$ спричиняє нестабільні коливання ($\sigma(w) = 0.04$).

Таблиця 2.

Динаміка адаптації ваг через SRA (еталон $w^* = (0.15; 0.30; 0.35; 0.20)$)

Цикл SRA	w_1	w_2	w_3	w_4	$\|w - w^*\|_2$
Еталон w^*	0.150	0.300	0.350	0.200	–
w^0 (поч.)	0.250	0.250	0.250	0.250	0.152
w^1 (1 сем.)	0.192	0.278	0.311	0.219	0.089
w^2 (2 сем.)	0.167	0.292	0.336	0.205	0.039
w^3 (3 сем.)	0.156	0.298	0.347	0.199	0.011

SRA за 3 цикли скорочує $\|w-w^*\|_2$ з 0.152 до 0.011 (зі зменшенням значення на 92.8%), кореляція зростає з 0.61 до 0.89. Домінуючим напрямком є: $\uparrow w_3$ (psych), $\uparrow w_2$ (circ), $\downarrow w_1$ (tech). Тобто учасники навчального процесу цінують психологічний комфорт і хронотипову відповідність вище, ніж формальну відсутність конфліктів.

5.4. Обчислювальна вартість. Збільшення часу покоління на $11.8 \pm 0.9\%$ було досягнуто на конфігурації AMD EPYC 7502P, 1 потік. Загальна вартість досягнення F > 0.75 є меншою, ніж у Baseline, через скорочення поколінь на 34%. Тобто для 18 груп значення часу складає 38.4 ± 3.7 хвилин на 1 потоці та у випадку 4 потоків (Parallel.For) це 12 ± 1.8 хв. Фактично, у даному випадку паралелізація обчислень надає прямо пропорційне збільшення ККД усього процесу.

Під час розрахунку використовуваного обсягу оперативної пам'яті (RAM) було отримано наступні значення. Кожна хромосома займає 6×8×18×2×4 = 13 824 байти (int32 для e та r). Таким чином, популяція з 200 особин потребує приблизно 2.7 МБ оперативної пам'яті. Кешовані метрики m(x) додають 200×3×8 = 4 800 байт, що є незначним фактором. У разі горизонтального масштабування на k вузлів загальний обсяг пам'яті зростає лінійно, тобто k×2.7 МБ для популяцій з накладними витратами на серіалізацію у випадку міграції у JSON з 15 КБ на учасника навчального процесу. Для k=4 загальний обсяг оперативної пам'яті складає до 11 МБ, що є не критичним навантаженням для сучасних серверів.

6. Обговорення

Архітектура модульного моноліту ПСАД на .NET та Windows Server реалізує чотири модулі (DCS, OC, SRA, PS) з ізоляцією через MediatR і EF та публічні інтерфейси. Ізоляція модулів забезпечує незалежне тестування та інтеграцію в існуючу інфраструктуру ЗВО, що використовує ОС Windows без додаткової інфраструктури.

Обмеження: SRA потребує 3 семестри для стабілізації ваг. Перші 2 семестри система функціонує з наближеними вагами ($\|w-w^*\|_2 > 0.05$). Для нових ЗВО рекомендується прискорена адаптація ($\alpha=0.6$) у першому семестрі. Лінійна модель SRA не відображає нелінійні залежності задоволеності у разі різких змін контингенту; у таких випадках рекомендується скидання ваг.

Порівняно з монолітними системами, як наприклад FET та UniTime, ПСАД не потребує додаткової інфраструктури поза Windows Server та MS SQL, яка вже розгорнута у більшості Українських ЗВО. Модульний моноліт на .NET розгортається як служба ОС Windows за допомогою PowerShell DSC без додаткових інструментів. Для малих ЗВО, у яких навчається до 20

груп студентів, система працює на MS SQL Express із мінімальними вимогами до операційної системи.

Жодна з відомих систем, таких як Mimosa, aSc, FET або UniTime, не поєднує адаптивні ваги, хронобіологічну модель та зворотний зв'язок з усіма учасниками навчального процесу. ПСАД є першою реалізацією, що об'єднує всі три в єдиній .NET архітектурі [3, 16].

Рефакторинг програмного коду у мікросервіси (gRPC замість MediatR) потребує орієнтовно до 60 людино-годин за умови роботи спеціалістів середнього рівня. Поточний вибір монолітної архітектури обумовлений інфраструктурними обмеженнями українських ЗВО [20].

Процес розроблення даного програмного забезпечення має наступні перспективи: (1) інтеграція з моніторингом успішності для збагачення Ф; (2) DL для прогнозу оптимальних слотів; (3) мобільний застосунок з рекомендаціями щодо часу самопідготовки [6].

Порівняння ПСАД із комерційними системами планування демонструє принципову відмінність у підході. Системи, як-от FET та UniTime, орієнтовані на ручне налаштування вагових параметрів адміністратором з подальшою фіксацією конфігурації на весь термін використання.

Практичне впровадження ПСАД у ЗВО вимагає виконання трьох підготовчих кроків. Перший передбачає збір хронотипових профілів через MEQ-опитувальник на початку академічного року (оцінений час: 20 хвилин на учасника, загалом до 2 тижнів для всього контингенту). Другий полягає у формуванні матриці когнітивної сумісності дисциплін S за участю методистів кафедр (оцінений час: 3 робочі дні). Третій охоплює інтеграцію з існуючою деканатською системою через REST API DCS (оцінений час: 2 тижні розробки адаптера). Після цих базових кроків система готова до повністю автоматичного складання та адаптації розкладу навчальних занять.

7. Висновки

– Запропонована архітектура модульного моноліту ПСАД на .NET реалізує чотири модулі (DCS, OC, SRA, PS) з ізоляцією через MediatR, EF Core та публічні інтерфейси. Ізоляція модулів системи надає можливість незалежного тестування та інтеграції в існуючі АСУ без модифікації програмного коду ядра системи.

– Ядро підсистеми ОС реалізують оператори CPC та AWM, що переносять антропоцентричні обмеження на рівень структури генетичних операцій. Верифікація підтверджує скорочення часу збіжності на 34% та підвищення якості розв'язку на 18.3% порівняно з Baseline GA при збільшенні часу покоління лише на 11.8%.

– Підсистема SRA автоматично коригує вагові коефіцієнти через OLS-регресію зворотного зв'язку з EMA ($\alpha = 0.4$) та нижнім обмеженням ($w_i \geq 0.05$). За 3 цикли роботи алгоритму кореляція з еталонними вагами зростає з 0.61 до 0.89 у разі стійкості до шуму $\sigma \leq 0.01$.

– Перспективами розвитку системи є реалізація паралельної острівної версії підсистеми ОС (до 16 вузлів з оцінкою масштабування), заміна лінійної регресії SRA на RL-агент для нелінійних залежностей задоволеності та розробка модуля динамічного перепланування розкладу впродовж семестру.

Література

1. *Goldin A. P. et al.* Interplay of chronotype and school timing predicts school performance. Nature Human Behaviour, 2020. Vol. 4. No. 4. P. 387–396.
2. *Jankowski K. S., Díaz-Morales J. F., Vollmer C.* Chronotype, time of day, and performance on intelligence tests in the school setting. Journal of Intelligence, 2023. Vol. 11. No. 1. Art. 13. DOI: https://doi.org/10.3390/jintelligence11010013
3. *Ситнік О. О. Антропоцентрична модель синтезу розкладу занять у ЗВО авіакосмічного профілю.* Aerospace Technic and Technology, 2025. № 6 (215). С. 1–12. DOI: https://doi.org/10.32620/aktt.2025.6.07
4. *Rodríguez Ferrante G. et al.* A better alignment between chronotype and school timing is associated with lower grade retention. npj Science of Learning, 2023. Vol. 8. Article 21.
5. *Al-Rfooh O. F., Khater W.* The impact of chronotype on physical health, psychological health, and job performance among health care providers in acute care settings. International Journal of Healthcare Management, 2023. Vol. 16. No. 4. P.

581–589. DOI: https://doi.org/10.1080/20479700.2023.2177665
6. *Maučec K., Štukovnik V.* The relationship between chronotype and academic achievement among Slovene university students: The mediating role of trait self-control and sleep quality. Center for Educational Policy Studies Journal, 2024. DOI: https://doi.org/10.26529/cepsj.1790
7. *Smarr B. L., Schirmer A. E.* 3.4 million real-world learning management system logins reveal the majority of students experience social jet lag. Scientific Reports, 2018. Article 4793.
8. *Abdipoor S., Yaakob R., Goh S. L., Abdullah S.* Meta-heuristic approaches for the university course timetabling problem. Intelligent Systems with Applications, 2023. Vol. 19. Art. 200253. DOI: https://doi.org/10.1016/j.iswa.2023.200253
9. *Bashab A., Ibrahim A. O., Tarigo Hashem I. A. et al.* Optimization techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues. Computers, Materials & Continua, 2023. Vol. 74. No. 3. P. 6461–6484. DOI: https://doi.org/10.32604/cmc.2023.034051
10. *Rezaeipanah A. et al.* A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Applied Intelligence, 2021. Vol. 51. P. 467–492. DOI: https://doi.org/10.1007/s10489-020-01833-x
11. *Mahlous A. R., Mahlous H.* Student timetabling genetic algorithm accounting for student preferences. PeerJ Computer Science, 2023. Vol. 9. Article e1200. DOI: https://doi.org/10.7717/peerj-cs.1200
12. *Davison M., Kheiri A., Zografos K. G.* Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling, 2024. Vol. 28. P. 195–215. DOI: https://doi.org/10.1007/s10951-024-00817-w
13. *Zanevych O., Kukharskyy V.* Overview of machine learning methods for academic scheduling. Electronics and Information Technologies, 2024. Vol. 27. DOI: https://doi.org/10.30970/eli.27.8
14. *Horne J. A., Ostberg O.* A self-assessment questionnaire to determine morningness-eveningness in human circadian rhythms. International Journal of Chronobiology, 1976. Vol. 4. No. 2. P. 97–110.
15. Eiben A. E., Smith J. E. *Introduction to Evolutionary Computing*, 2nd ed. Springer, Berlin, 2015. 302 p.
16. *Ситнік О., Вдовітченко О. Математичні структури та засоби антропоцентричної диспетчеризації у закладах вищої освіти.* Open Information and Computer Integrated Technologies, 2025. № 105. С. 212–226. DOI: https://doi.org/10.32620/oikit.2025.105.17
17. *Chen M. C. et al.* A survey of university course timetabling problem: Perspectives, trends and opportunities. IEEE Access, 2021. Vol. 9. P. 106515–106529. DOI: https://doi.org/10.1109/ACCESS.2021.3100613
18. *Gu X., Krish M., Sohail S. et al.* From integer programming to machine learning: A technical review on solving university timetabling problems. Computation, 2025. Vol. 13. No. 1. Art. 10. DOI: https://doi.org/10.3390/computation13010010
19. *Zanevych O. B., Kukharskyy V. M.* Solving university timetabling problems using constraint programming with adaptive local search and elite solution memory. Visnyk of the Lviv University. Series Applied Mathematics and Computer Science, 2025. Vol. 34. DOI: https://doi.org/10.30970/vam.2025.34.13638
20. *Носиков О., Ситнік О. Формування єдиного інформаційного простору в закладах вищої освіти: виклики, рішення та перспективи розвитку.* Open Information and Computer Integrated Technologies, 2025. № 104. С. 200–213. DOI: https://doi.org/10.32620/oikit.2025.104.13

References

1. Goldin, A. P. et al. (2020), Interplay of chronotype and school timing, *Nature Human Behaviour,* Vol. 4, No. 4, P. 387–396.
2. Jankowski, K. S., Díaz-Morales, J. F., Vollmer, C. (2023), Chronotype, time of day, and performance on intelligence tests in the school setting, *Journal of Intelligence,* Vol. 11, No. 1, Art. 13. DOI: https://doi.org/10.3390/jintelligence11010013
3. Sytnik, O. O. (2025), Anthropocentric model of class scheduling in HEI of aerospace profile, *Aerospace Technic and Technology,* No. 6 (215), P. 1–12. DOI: https://doi.org/10.32620/aktt.2025.6.07
4. Rodríguez Ferrante, G. et al. (2023), A better alignment between chronotype and school timing, *npj Science of Learning,* Vol. 8, Article 21.
5. Al-Rfooh, O. F., Khater, W. (2023), The impact of chronotype on physical health, psychological health, and job performance among health care providers, *International Journal of Healthcare Management,* Vol. 16, No. 4, P. 581–589. DOI: https://doi.org/10.1080/20479700.2023.2177665

6. Maučec, K., Štukovnik, V. (2024), The relationship between chronotype and academic achievement among Slovene university students, *Center for Educational Policy Studies Journal.* DOI: https://doi.org/10.26529/cepsj.1790
7. Smarr, B. L., Schirmer, A. E. (2018), 3.4 million real-world LMS logins reveal social jet lag, *Scientific Reports,* Article 4793.
8. Abdipoor, S., Yaakob, R., Goh, S. L., Abdullah, S. (2023), Meta-heuristic approaches for the university course timetabling problem, *Intelligent Systems with Applications,* Vol. 19, Art. 200253. DOI: https://doi.org/10.1016/j.iswa.2023.200253
9. Bashab, A., Ibrahim, A. O., Tarigo Hashem, I. A. et al. (2023), Optimization techniques in university timetabling problem, *Computers, Materials & Continua,* Vol. 74, No. 3, P. 6461–6484. DOI: https://doi.org/10.32604/cmc.2023.034051
10. Rezaeipanah, A. et al. (2021), A hybrid algorithm for the university course timetabling problem, *Applied Intelligence,* Vol. 51, P. 467–492. DOI: https://doi.org/10.1007/s10489-020-01833-x
11. Mahlous, A. R., Mahlous, H. (2023), Student timetabling genetic algorithm accounting for preferences, *PeerJ Computer Science,* Vol. 9, Article e1200. DOI: https://doi.org/10.7717/peerj-cs.1200
12. Davison, M., Kheiri, A., Zografos, K. G. (2024), Modelling and solving the university course timetabling problem with hybrid teaching considerations, *Journal of Scheduling,* Vol. 28, P. 195–215. DOI: https://doi.org/10.1007/s10951-024-00817-w
13. Zanevych, O., Kukharskyy, V. (2024), Overview of machine learning methods for academic scheduling, *Electronics and Information Technologies,* Vol. 27. DOI: https://doi.org/10.30970/eli.27.8
14. Horne, J. A., Ostberg, O. (1976), A self-assessment questionnaire to determine morningness-eveningness, *International Journal of Chronobiology,* Vol. 4, No. 2, P. 97–110.
15. Eiben, A. E., Smith, J. E. (2015), *Introduction to Evolutionary Computing,* 2nd ed., Springer, Berlin, 302 p.
16. Sytnik, O., Vdovitchenko, O. (2025), Mathematical structures and tools for anthropocentric dispatching in higher education institutions, *Open Information and Computer Integrated Technologies,* No. 105, P. 212–226. DOI: https://doi.org/10.32620/oikit.2025.105.17
17. Chen, M. C. et al. (2021), A survey of university course timetabling problem: Perspectives, trends and opportunities, *IEEE Access,* Vol. 9, P. 106515–106529. DOI: https://doi.org/10.1109/ACCESS.2021.3100613
18. Gu, X., Krish, M., Sohail, S. et al. (2025), From integer programming to machine learning: A technical review on solving university timetabling problems, *Computation,* Vol. 13, No. 1, Art. 10. DOI: https://doi.org/10.3390/computation13010010
19. Zanevych, O. B., Kukharskyy, V. M. (2025), Solving university timetabling problems using constraint programming with adaptive local search and elite solution memory, *Visnyk of the Lviv University. Series Applied Mathematics and Computer Science,* Vol. 34. DOI: https://doi.org/10.30970/vam.2025.34.13638
20. Nosikov, O., Sytnik, O. (2025), Formation of a unified information space in higher education institutions, *Open Information and Computer Integrated Technologies,* No. 104, P. 200–213. DOI: https://doi.org/10.32620/oikit.2025.104.13

Дата першого надходження до видання: 06.04.2026
Внутрішня рецензія отримана: 23.04.2026
Зовнішня рецензія отримана: 29.04.2026
Дата прийняття статті до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Ситнік Олег Олександрович,
аспірант кафедри інженерії
програмного забезпечення
Sytnik Oleg,
Post-graduate student
ORCID: 0009-0009-4504-3489.

Місце роботи авторів:

Національний аерокосмічний університет
«Харківський авіаційний інститут»,
National Aerospace University
"Kharkiv Aviation Institute"
вул. Чкалова, 17, Харків, Україна, 61070.
E-mail: o.sytnik@khai.edu

УДК 517.958:57 +519.711.3 + 612.51.001 https://doi.org/10.15407/pp2026.02.049

R.D. Grygoryan, I.P. Sinitsyn, A.G. Degoda, T.V. Lyudovyk, O.I. Yurchak, N.A. Strutynska

MODELING OF PANCREAS-LIVER INTERACTION UNDER CONTROLED HEMODYNAMICS

Autonomous software (AS) was created to simulate the dynamics of the glucose-insulin-glycogen-glucagon relationship in a healthy person. Our AS is based on a quantitative mathematical model consisting of three components: a model describing the pancreas-liver and the pancreas-skeletal muscles relationships; a model describing blood circulation in the branched cardiovascular system, taking into account neurohumoral regulators of cardiac function, vascular tone and total blood volume; and a model describing blood filtration in the renal glomeruli and tubular reabsorption. A glucose tolerance test (GTT) was also programmed. Test simulations demonstrated adequate model responses. The program is integrated into a specialized computer simulator (SCS). It allows studying mechanisms that, depending on the dynamics of exogenous and endogenous physicochemical variables, dynamically form multidimensional health landscape of biometric indicators. The effect of extreme blood flow increase on the dynamics of the main variables of the model was also simulated without additional carbohydrate intake. AS is created in C#, and can be delivered as an Exe-module for IBM-compatible computers. Medical students can be additional users of the AP as an additional didactic tool. The AS ensures the preservation of all simulation data for future reviews and publications. The AS can be used by future endocrinologists in their training. Physiologists interested in the integrative physiology of cellular life support are recommended to use the SCS.
Keywords: glucose homeostasis, physiological systems, mathematical model, visualization, students

Р.Д. Григорян, І.П. Сініцин, А.Г. Дегода, Т.В. Людовик, О.І. Юрчак, Н.А. Струтинська

МОДЕЛЮВАННЯ ВЗАЄМОДІЇ ПІДШЛУНКОВОЇ ЗАЛОЗИ ТА ПЕЧІНКИ В УМОВАХ КОНТРОЛЬОВАНОЇ ГЕМОДИНАМІКИ

Автономне програмне забезпечення (АПЗ) створено для моделювання динаміки взаємозв'язку ендогенних агентів глюкоза-інсулін-глікоген-глюкагон у здорової людини. АПЗ базується на кількісній математичній моделі, що складається з трьох компонентів: моделі, що описує взаємозв'язок підшлункова залоза-печінка; моделі, що описує кровообіг у розгалуженій серцево-судинній системі, враховуючи нейрогуморальні регулятори серцевої функції, судинного тонусу та загального об'єму крові; та моделі, що описує фільтрацію крові в ниркових клубочках та канальцеву реабсорбцію. Також було запрограмовано тест на толерантність до глюкози (ГТТ). Тестові симуляції продемонстрували адекватні реакції моделі. Програма інтегрована у спеціалізований комп'ютерний симулятор (СКС). Він дозволяє вивчати механізми, які, залежно від динаміки екзогенних та ендогенних фізико-хімічних змінних, динамічно формують багатовимірній ландшафт здоров'я у просторі біометричних показників. Вплив екстремального збільшення кровообігу на динаміку основних змінних моделі також було змодельовано без додаткового притока вуглеводів. АПЗ — це автономне програмне забезпечення на C#, що постачається як Exe-модуль для IBM-сумісних комп'ютерів. Студенти-медики можуть користуватися АПЗ як додатковим дидактичним засобом. АПЗ, що забезпечує збереження всіх даних моделювання для майбутніх розглядів та публікацій, може бути використане ендокринологами у дослідженнях. Фізіологам, які цікавляться інтегративною фізіологією клітинного життєзабезпечення, рекомендується використовувати СКС.
Ключові слова: гомеостаз глюкози, фізіологічні системи, математична модель, візуалізація, студенти

Introduction

Irregular intake of carbohydrates can create energy problems in human sensitive cells like neurons, kidney cells, hepatocytes, and myocytes. Special mechanisms smoothing out the flow of glucose into the blood and providing glucose homeostasis evolved. Important roles in glucose homeostasis play the pancreas producing insulin, hepatocytes, and myocytes that

transform the exec amount of blood glucose into liver and muscle glycogen. However, namely the liver, accumulating up to 120 g glycogen and capable of its reverse transformation to blood glucose when its concentration essentially drops, is the main organ dynamically reacting to blood glucose lack. Normally, the urine does not contain essential concentration of glucose. At the same time, the mechanism responsible for blood glucose homeostasis has of limited power. Therefore, even under physiological conditions, extreme glucose intakes lead to elevated concentrations of glucose in the urine which finally removes serious volumes of glucose. In endocrinology, special glucose tolerance test (GTT) is applied to assess the efficiency of mechanisms providing glucose homeostasis [1].

The main product of the pancreas is the hormone insulin. It performs two functions: first, it promotes the entry of glucose into the cell; second, it activates the transformation of excess glucose into glycogen and its accumulation in the liver and muscles. An additional product of the pancreas is the hormone glucagon. Its production is activated when there is not enough glucose in the blood.

It is under such conditions that the reverse transformation of glycogen into glucose occurs in hepatocytes. Glucagon enhances the action of the heart pump and affects blood pressure. Therefore, we model these physiological processes, since they are an important link in the energy supply of cell metabolism.

The model of glucose homeostasis

Various models have been proposed for an in-depth study of the glucose homeostasis mechanism and its possible disturbances (for examples, [2-8]). The models [5,6] help understand the mechanisms of type 2 diabetes by demonstrating how disruptions at the cellular level lead to impaired glucose regulation throughout the body.

The model we proposed describes dynamic interactions of blood glucose ($G(t)$), insulin ($I(t)$), liver glycogen ($g_L(t)$), muscle glycogen ($g_M(t)$), and glucagon ($g_G(t)$) depending on velocities of glucose incomes ($v_{G+}(t)$) and consumption ($v_{G-}(t)$). The model takes into account that in certain cells (hepatocytes, fats, and skeletal myocytes) glucose consumption is associated with insulin concentration while other specialized cells directly consume glucose.

$$T_G \frac{dG(t)}{dt} = G(t) + \alpha \cdot I(t) - \beta \cdot W(t), \quad (1)$$

In (1), T_G is the time constant of glucose dynamics, $W(t)$ - presents the power of general biological work, coefficient $\alpha \neq 0$ only for hepatocytes, fats, and skeletal myocytes. Virtually, by altering the value of coefficient β, the user can simulate nuances of glucose consumption dynamics in chosen cell types.

The dynamics of insulin is described by following differential equation:

$$T_I \frac{dI(t)}{dt} = \gamma \cdot G(t) - \chi \cdot I(t), \quad (2)$$

In (2), γ and χ are approximation constants.

Special mechanism providing blood glucose homeostasis is modeled using the differential equation describing excess glucose transformation into $g_L(t)$ and $g_M(t)$, and reverse transformation of $g_L(t)$ to $G(t)$.

$$T_{gL} \frac{dg_L(t)}{dt} = \begin{cases} k_G^g \cdot (G(t) - G_{cr}) - g_L(t), & G(t) \geq G_{cr} \\ 0, & G(t) < G_{cr} \end{cases}, \quad (3)$$

$$T_{gM} \frac{dg_M(t)}{dt} = k_G^{g_M} \cdot (G(t) - G_{cr} - g_M(t)) - g_M(t), \quad (4)$$

$$G(t) \geq G_{cr}; g_M(t) < g_M^{\max}$$

In (4), T_{gM}, $k_G^{g_M}$, and $g_M^{\max}$ are approximation constants.

The next equation describes the dynamics of glucagon:

$$T_{gG} \frac{dg_G(t)}{dt} = \begin{cases} \delta \cdot (G(t) - G_{cr}) - g_G(t) - g_u, & G(t) > G_{cr} \\ \delta \cdot (G_{cr} - G(t)) - g_G(t), & G(t) \leq G_{cr} \end{cases}, \quad (5)$$

In (5), T_{gG}, δ, and g_u are approximation constants.

So, the equation system (1)-(5) describes the dynamics of the glucose-insulin-glycogen-glucagon relationships in a healthy individual.

Autonomic software (AS) was created for the numerical solution of this system of equations on a computer. The primary user of this program was intended to be a

specialist (endocrinologist), so a problem-oriented, user-friendly interface (UI) was developed. In parallel, a simulator was developed that reproduces the integrative responses of human internal organs to a wide range of endogenous and exogenous dynamic factors [9,10]. They are subdivided into eight clusters. The appearance of a special screen form that allows the user to operate with the simulation results is shown in Figure 1. The user can arbitrarily generate the desired set of characteristics for graphical visualization. Initially, these characteristics are grouped by related features into eight clusters on the left side of the window. On the right side of the window are two sub-windows where the variables to be analyzed are collected. These sub-windows allow the model's input and output variables to be separated. By opening each cluster one by one, the user selects a variable in it and moves it to one of the sub-windows on the right side. This quickly generates a set of analyzed variables in the form of graphs.

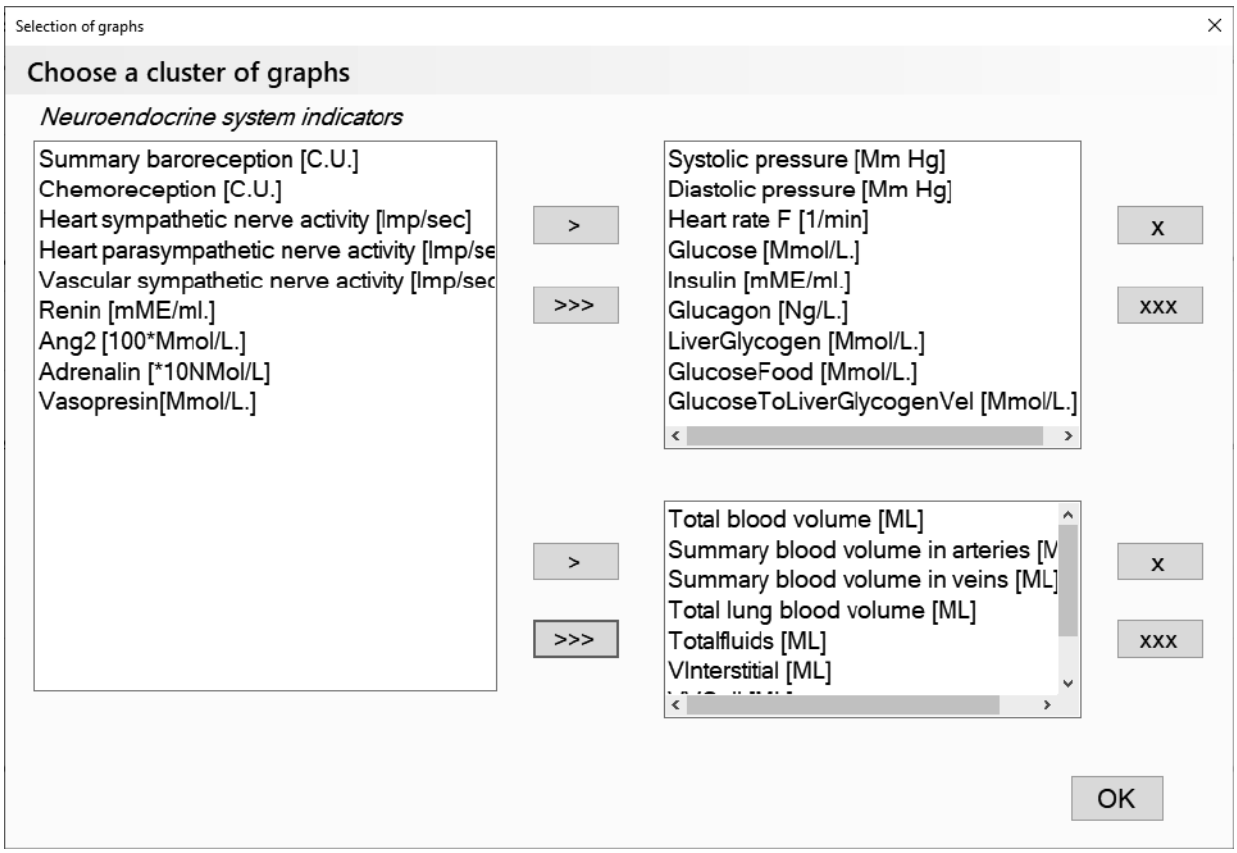

Fig. 1. The screen form of UI for actualizing sets of input-output variables

In particular, our complex model that also describes the overall circulation includes special equations concerning hemodynamic effects of glucagon through its dilating influence on coronary arteries and increase of coronary circulation. The latter effect elevates the myocardium power. Besides, assistant equations describing velocities of glucose incomes $v_{G+}(t)$ and consumption $v_{G-}(t)$ create an opportunity to simulate different scenarios of glucose dynamics depending on given $v_{G+}(t)$ and $v_{G-}(t)$.

Naturally, current rates of glucose incomes $v_{G+}(t)$ and consumption $v_{G-}(t)$ are unpredictable variables. Our software provides with special options to simulate both these variables using a list of analytical functions and screen forms for modifying values of coefficients. Two examples below illustrate this modification by actualizing values of constants $v(0), \eta_1, \eta_2, \eta_3, \varphi, and\, v_2$.

$$v_{G+}(t) = \begin{cases} v(0) + \eta_1 \cdot t, & v_{G+}(t) \le v_1 \\ v(0) - \eta_2 \cdot t, & v_{G+}(t) \ge v_2 \end{cases},$$

$$v_{G+}(t) = v(0) + \eta_3 \cdot \sin(\varphi \cdot t), \quad 0 < (t) \le t_1.$$

In contrast, $v_{G-}(t)$ depends on organs' activities generally correlating with regional blood flows. Therefore, the equation describing this dependency looks like (6):

$$\frac{dv_{G-}}{dt} = a_1 \cdot q_b(t) + a_2 \cdot q_h(t) + a_3 \cdot q_k(t) \cdot (1 + a_4 \cdot q_{kr}(t)) + a_5 \cdot I \cdot (c \cdot q_h(t) + a_6 \cdot q_{lm}(t)), \quad (6)$$

Here, $q_b(t)$ is the summary brain flow, $q_h(t)$ is the coronary blood flow, $q_k(t)$ is the kidney glomeruli blood flow, $q_{kr}(t)$ reabsorption fraction, $q_h(t)$ and $q_{lm}(t)$ represent blood flows in insulin-depended organs, while $a_1 - a_6$ are approximation constants.

$$\frac{dv_L}{dt} = \partial \cdot C_L(t) + a_{11} \cdot W(t) \cdot I(t) \cdot (c \cdot q_h(t) + a_{12} \cdot q_{lm}(t)) + a_{13} \cdot q_h(t) + a_{14} \cdot q_k(t) \cdot (1 + a_{15} \cdot q_{kr}(t)) + a_{16} \cdot I(t) \cdot (c \cdot q_h(t) + a_{17} \cdot q_{lm}(t)), \quad (7)$$

In (7), ∂, a_{11} - a_{17}, and c are approximation constants.

Food glucose velocity $v_{FG+}(t)$ is proportional to the volume F_{G+} of glucose intake, so $v_{FG+}(t) = k_g \cdot F_{G+} / t$, where k_g characterizes the average intensity of glucose entry from the gastrointestinal tract into the blood.

Glucagon altering the lumen of coronary arteries modulates their resistance $R_C(t)$ relatively to initial value of $R_C(0)$ characteristic for basal coronary flow $q_C(0)$:

$$R_C(t) = R_C(0) \cdot (1 - \varpi_6 \cdot q_C(0) / q_C(t)), \quad (6)$$

This elevates the ventricle contractility $k(t)$

$$k(t) = k(0) + \varpi_4 \cdot (q_C(t) - q_C(0)). \quad (7)$$

In (6) and (7), $\varpi_4, \varpi_6, and\ k(0)$ are constants.

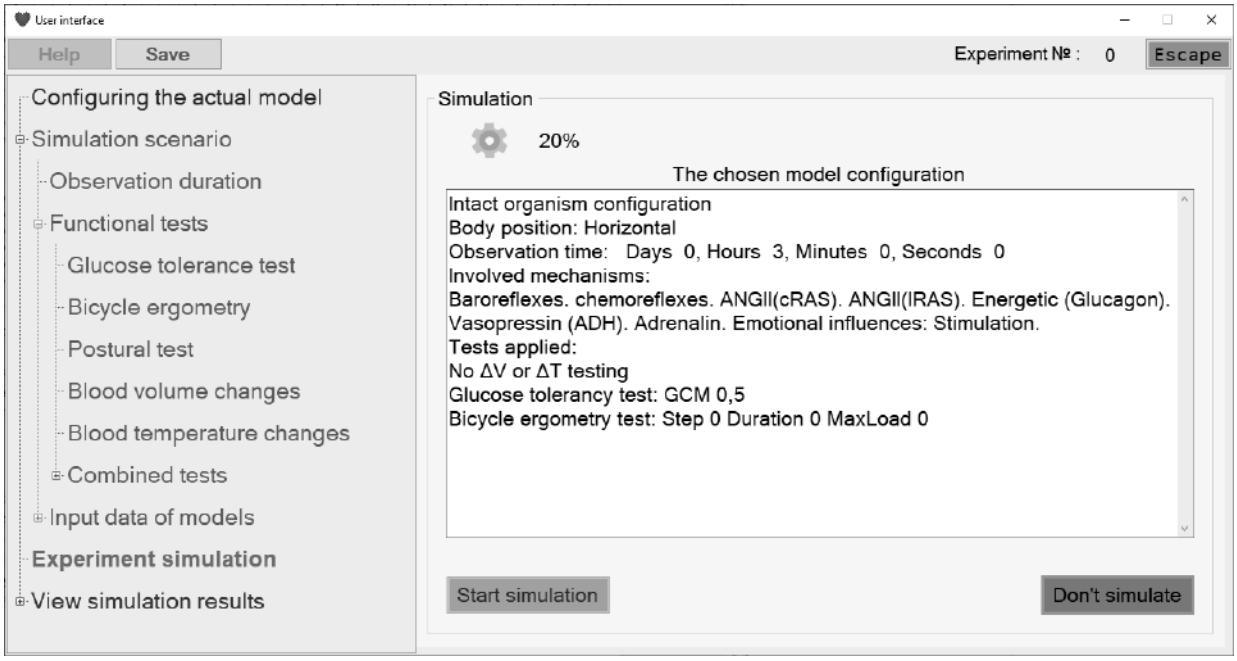

Fig. 2. The screen-form indicating a part of options for configuring the actual model and constructing the simulation scenario (left) and information concerning the chosen model configuration (right)

Examples of simulations

Our simulator has two versions: AS and extended simulator (ES). The latter includes models of the functionally integrated internal organs that optimize cell life support (see [9,10]).

Simulations presented further are addressed both to programmers and experts modeling human physiology. At the same time, the reader's belief in the simulator depends in no small part on the adequacy of the simulations.

ES can simulate effects caused by other organs and physiological systems. Therefore, before to consider simulations provided AS, it is useful to look at Figure 3. It presents main hemodynamic characteristics simulated ES. They show that transient processes in the system resolve quickly: a steady-state circulation regime is observed for almost the entire ten-minute exposure period. This guarantees that the dynamics of all variables in the glucose homeostasis model provided by AS are determined by the dynamics of variables specific to the glucose homeostasis model. Naturally, if the characteristics of the coupled models change, the dynamics of the glucose homeostasis model variables will also change.

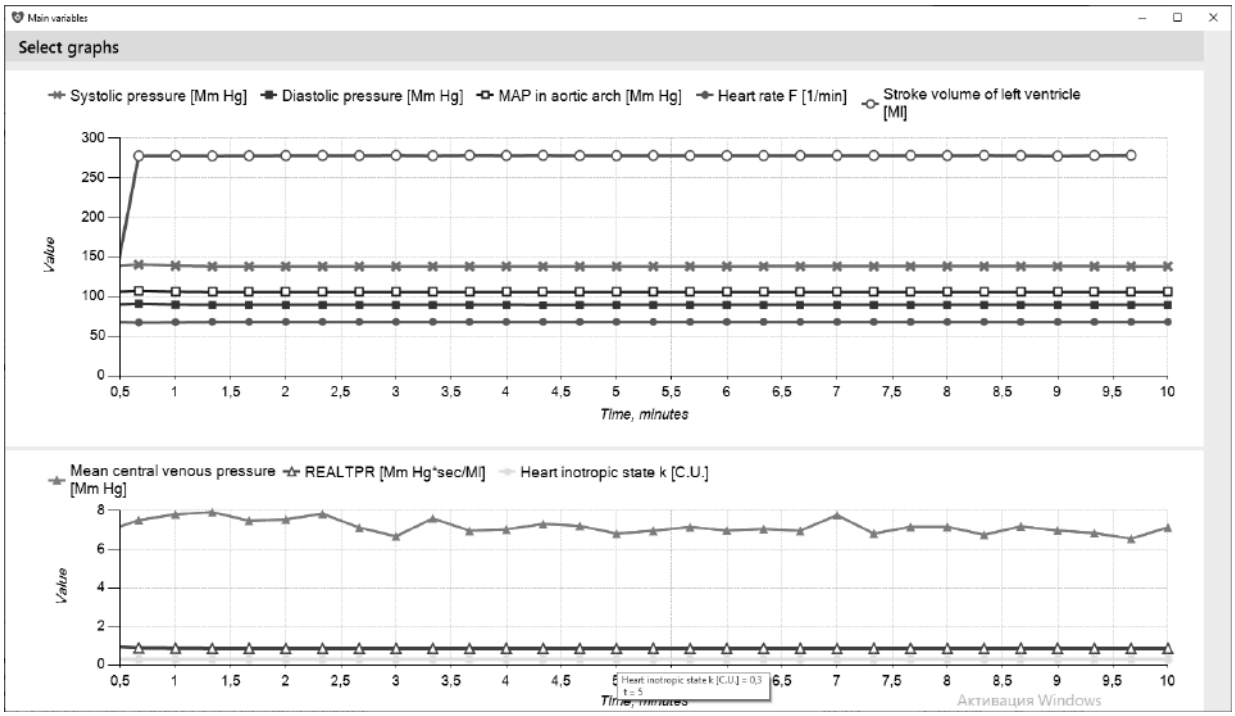

Fig. 3. Hemodynamics in control simulation

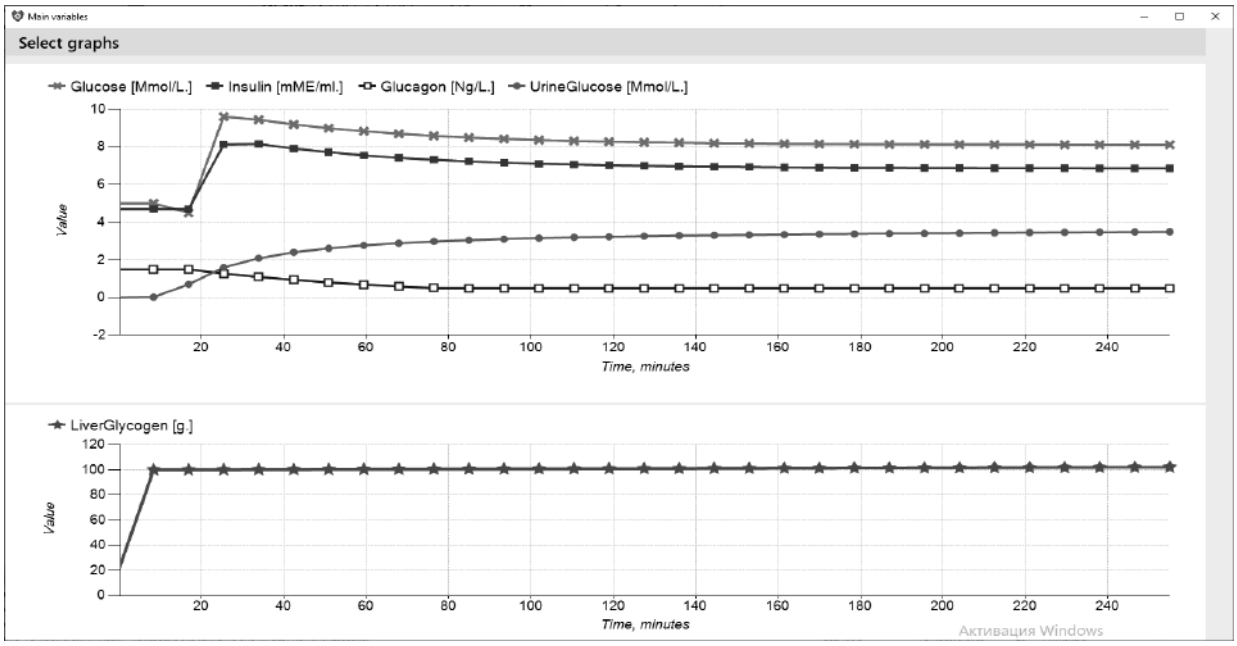

Fig. 4. Simulated dynamics of glucose, insulin, glucagon, and urine glucose (upper curves) and liver glycogen (bottom) in a healthy person model under stable glucose production (0.07 mmol/min) and consumption (0.05 mmol. /min) rates for 4.15 hours real time exposure

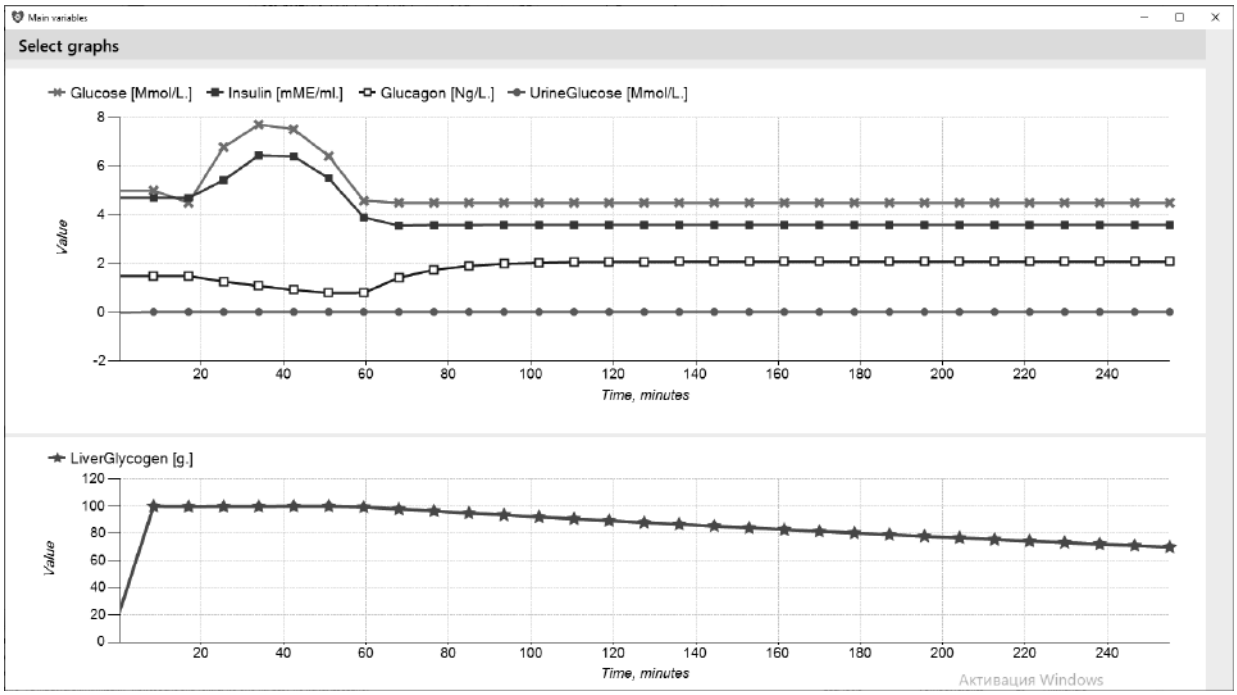

Fig. 5. Simulated dynamics of glucose, insulin, glucagon, and urine glucose (upper curves) and liver glycogen (bottom curve) in a healthy person model under stable glucose production (0.07 mmol/min) and consumption (0.075 mmol. /min) rates for 4.15 hours real time exposure

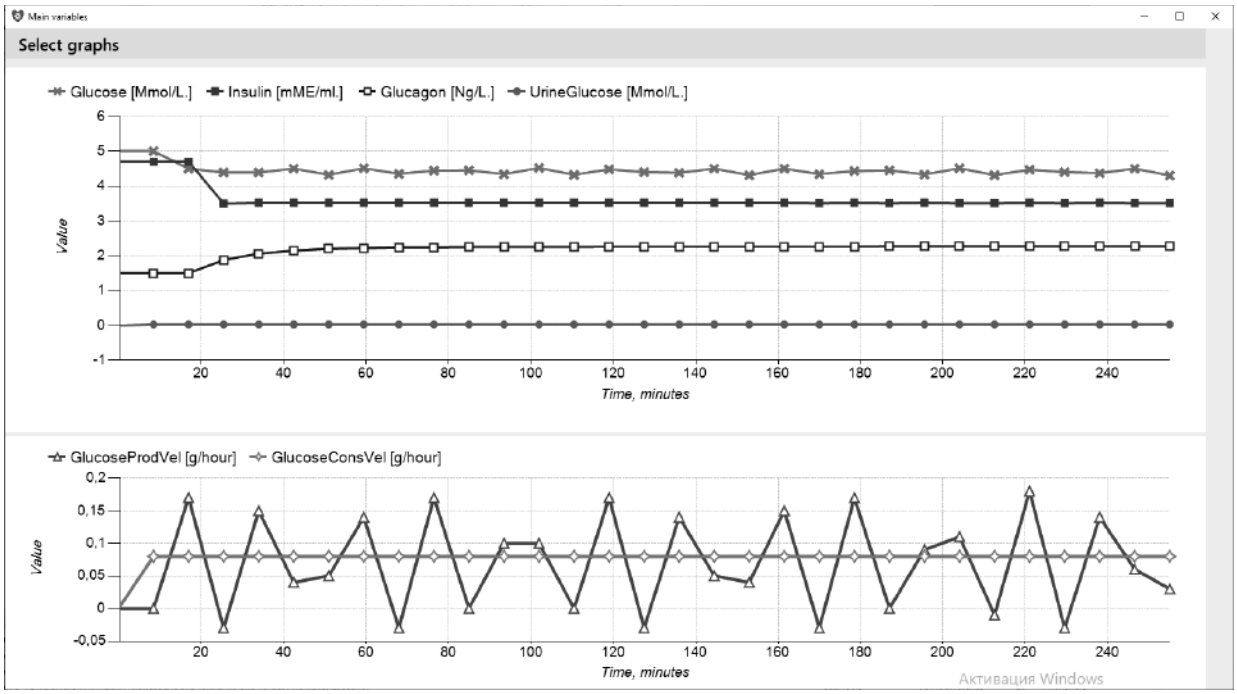

Fig. 6. Simulated dynamics of glucose, insulin, glucagon, and urine glucose (upper curves) and liver glycogen (bottom curve) in a healthy person model under periodic changes in the rate of glucose production according to a sinusoidal law and stable consumption (0.075 mmol. /min) rate for 4.15 hours real time exposure

Figures 3-6 demonstrate that our simulator is mainly adequate at least in the time intervals considered. This gives us the opportunity to simulate specific medical GTT-test.

Simulating glucose tolerance test - GTT

Pictures 7 and 8 illustrate simulations of standard GTT for different healthy persons and persons diseased with type 1 diabetes.

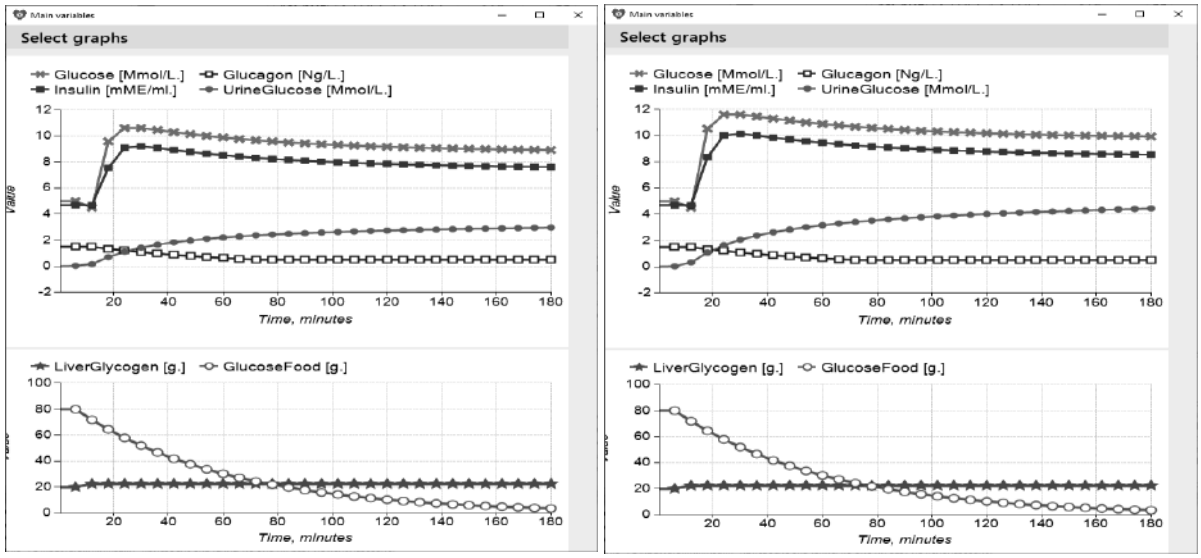

Fig. 7. Two simulated GTT (glucose tolerance tests) on models of the same healthy person. The left picture displays dynamics of glucose, insulin, glucagon, and urine glucose (upper curves) and liver glycogen and test glucose income (bottom curve) under stable glucose consumption rate of 0.06 mmol. /min but for glucose production rate of 0.085 mmol. /min. The right picture displays these same curves for the case of 0.095 mmol. /min glucose production rate and stable consumption rate of 0.06 mmol. /min.

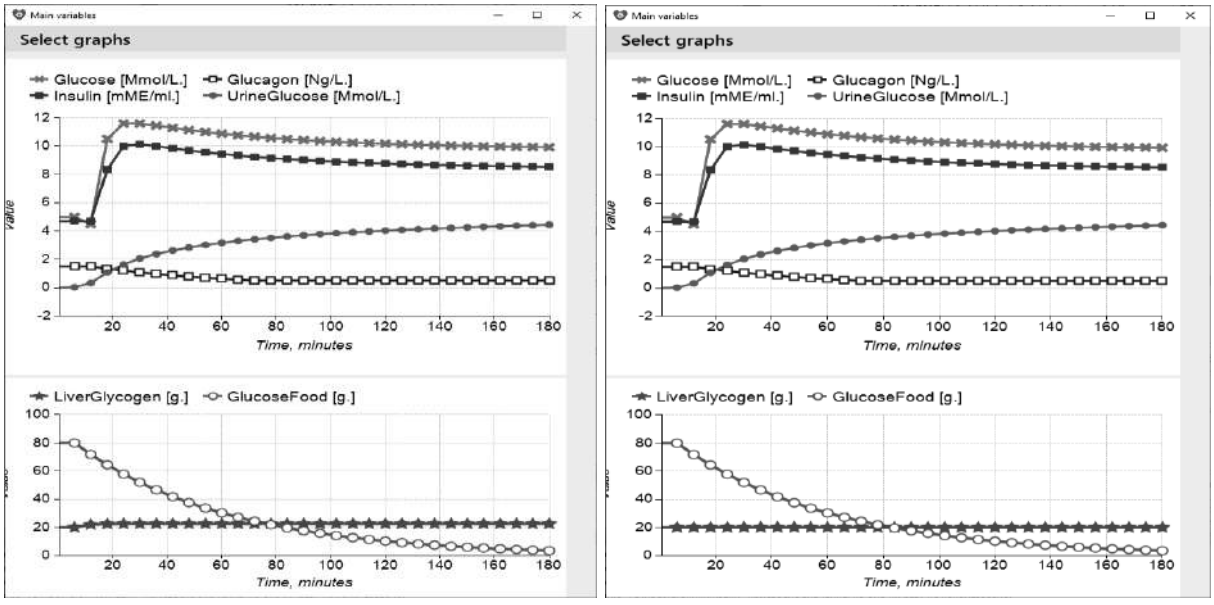

Fig. 8. Two simulated GTT (glucose tolerance tests) on models of persons having problems with glucose homeostasis. The left picture displays dynamics of glucose, insulin, glucagon, and urine glucose (upper curves) and liver glycogen and test glucose income (bottom curve) under stable glucose consumption rate of 0.06 mmol. /min. but for glucose production rate of 0.085 mmol. /min but with two-times weak homeostasis while the right picture displays these same curves for the uncontrolled blood glucose.

The graphs in the last figure should be compared with those on the left side of Figure 7. The most significant difference is evident when comparing the dynamics of urine glucose: as blood glucose homeostasis weakens, the amount of glucose in the urine increases. Under providing of real GTT, this is an objective indicator for the diagnosis of type 1 diabetes.

Discussion

In a real organism, most of biological variables are also influenced by other endogenous factors not accounted for in our model or in simulations demonstrated. Therefore, the proposed model is limited in its simulation capabilities. At the same time, an analysis of available clinical data shows that our model is generally adequate. This conclusion is also true during comparison of simulation results provided by our model with similar models by other authors (e.g., [1-7]), Moreover, our model offers a number of new possibilities for theoretical research aimed at understanding of fundamental physiological mechanisms and practical consideration of their nuances. For example, the inclusion of a muscle-mediated mechanism of glycogen storage, control of excess urinary glucose excretion, and the ability to model different pathways that deplete liver glycogen make our model a useful tool capable of providing insight into the uncontrolled underlying mechanisms that drive individual differences during GTT.

When performing a glucose tolerance test, the physician has no information on the levels of residual glycogen in the liver or muscles. The rate of accumulation of one of these products influences the rate of accumulation of the other. However, we were unable to find quantitative data on these residual products in the literature, and therefore cannot confirm the accuracy of the modeled rate dynamics. However, we note that incorporation of this model into an extended model describing the interaction of human internal organs has provided the first tool for theoretically studying the role of neuroendocrine modulators in glucose homeostasis.

To improve the effectiveness of our simulator, we plan to model the effects of lactate, adrenaline, and fat concentrations. In particular, it is well known that under low insulin, the body breaks down fats, creating ketone bodies (acetoacetate and β-hydroxybutyrate) for energy. Blood ketones normally less than 0.6 mmol/L, under Type 1 diabetes can elevate up to danger level of 3 mmol/L with symptoms extreme thirst, frequent urination, nausea, vomiting, abdominal pain, and fruity-smelling breath, requiring emergence care. As our complex model describes cardiovascular and kidney-urinary systems, its proper advancement could make the simulator usable for deeper study of pathological scenarios too.

Conclusion

A quantitative mathematical model of human glucose homeostasis has been developed. The model is implemented as a C# program. The program can function both as a standalone executable module and as part of a newly proposed integrated program that simulates the fundamental physiological patterns of the dynamic interactions of human internal organs. The program can serve as an additional didactic tool for visualization of the glucose-insulin-glycogen-glucagon dynamic relationships in a healthy individual.

References

1. Jagannathan R., Neves J.S., Dorcely B., Chung S.T., Tamura K., Rhee M., Bergman M. The Oral Glucose Tolerance Test: 100 Years Later. Diabetes Metab Syndr Obes. 2020 Oct 19;13:3787-3805. doi: 10.2147/DMSO.S246062.
2. Palumbo P., Ditlevsen S., Bertuzzi A., De Gaetano A. Mathematical modeling of the glucose-insulin system: a review. Math Biosci. 2013;244(2):69-81. doi: 10.1016/j.mbs.2013.05.006.
3. Ashoka H., Pradeep S., Nair K., Venkatesh K. Modeling glucose–insulin dynamics to evaluate healthy and diseased population characteristics in type 2 diabetes mellitus. Mathematics in Medical and Life Sciences, 2024.1(1). https://doi.org/10.1080/29937574.2024.2431821.
4. Bachar M. Sensitivity analysis for a delay mathematical model: the glucose-insulin model. Front. Appl. Math. Stat. 2025,11:1562636. doi: 10.3389/fams.2025.1562636.
5. Han Kyungreem, Kang Hyuk, Choi M. Y.,Kim Jinwoong, Lee Myung-Shik. Mathematical model of the glucose-insulin regulatory system:

From the bursting electrical activity in pancreatic β-cells to the glucose dynamics in the whole body. Physics Letters.A;2012,376; 45; https://doi.org/10.1016/J.PHYSLETA.2012.08.006
6. López-Palau, N.E., Olais-Govea, J.M. Mathematical model of blood glucose dynamics by emulating the pathophysiology of glucose metabolism in type 2 diabetes mellitus. Sci Rep. 2020. 10,12697. https://doi.org/10.1038/s41598-020-69629-0.
7. Omwenga, V.O.; Madhumati, V.; Vinay, K.; Srikanta, S.; Bhat, N. Mathematical Modelling of Combined Intervention Strategies for the Management and Control of Plasma Glucose of a Diabetes Mellitus Patient: A System Dynamic Modelling Approach. Mathematics 2023, 11, 306. https://doi.org/10.3390/math11020306
8. Falkenhain K. Ketones and Insulin: A Paradoxical Interplay With Implications for Glucose Metabolism, Journal of the Endocrine Society, 2025, 9, Issue 8, bvaf101, https://doi.org/10.1210/jendso/bvaf101.
9. Grygoryan R.D., Degoda A.G., Lyudovyk T.V., Yurchak O.I. Simulating of human physiological supersystems: integrative function of organs supporting cell life. Problems in programing,2024,4,77-88. DOI: 10.15407/pp2024.04.077.
10. Grygoryan R.D., Sinitsin I.P., Degoda A.G., Lyudovyk T.V., Yurchak O.I. A simulator providing theoretical research of human integrative physiology. Problems in programing, 2025,4,77-88. DOI: 10.15407/pp2025.04.077.

Дата першого надходження до видання: 05.03.2026
Внутрішня рецензія отримана: 23.03.2026
Зовнішня рецензія отримана: 02.04.2026
Дата рекомендації до друку: 05.06.2026
Дата публікації: 29.06.2026

About authors:

Grygoryan Rafik,
Ph.D., Doctor (biology),
Department chief
Григорян Рафік Давидович,
доктор біологічних наук,
завідувач відділу
http://orcid.org/0000-0001-8762-733X

Sinitsyn Igor,
Ph.D., Doctor (technical sciences),
Professor, Director
Сініцин Ігор Петрович,
доктор технічних наук,
професор, директор
http://orcid.org/0000-0002-4120-0784

Degoda Anna,
Senior scientist, Ph.D.
Дегода Анна,
доктор наук,
старший науковий співробітник
http://orcid.org/0000-0001-6364-5568.

Lyudovyk Tetyana,
Senior scientist, Ph.D.
Людовик Тетяна,
доктор наук,
старший науковий співробітник
https://orcid.org/0000-0003-0209-2001

Yurchak Oksana,
Leading software engineer
Юрчак Оксана,
Провідний інженер з програмування
https://orcid.org/0000-0003-3941-1555

Strutynska Natalia,
Leading scientist, Ph.D.
Струтинська Наталія,
доктор наук,
провідний науковий співробітник
http://orcid.org/0000-0001-7993-2705

Place of work:

Institute of Software Systems
of the National Academy of
Sciences of Ukraine
Інститут програмних систем
НАН України
03187, Куїv, Acad. Glushkov avenue, 40,
E-mail:
rgrygoryan@gmail.com,
ips2014@ukr.net,
anna@silverlinecrm.com,
tetyana.lyudovyk@gmail.com,
daravatan@gmail.com.

УДК 681.3 https://doi.org/10.15407/pp2026.02.058

Д.В. Рагозін, В.Є. Смірнов

SIMULATION AND ANALYSIS OF PEER-TO-PEER ROBOT SWARM NETWORK

In the paper we describe the full cycle development of TDMA-based communication protocol for a swarm of robots, including simulation and hard-ware implementation. The protocol is targeted to have a robust, interference-immune transport in the network. First, we developed a simulator, based on SimPy simulation package, which helps us to run thousands of simulations for proving the concept of TDMA-based communications for robotic swarm. Second, using the simulator we developed a bunch of techniques for the TDMA transport to improve network robustness and simulations allowed to gather statistics and choose the better algorithm. Third, we employed so-called AI tools to implement parts of simulator and a helper technique to convert simulation code to embedded code, approaching "digital twin" paradigm. Finally, the simulated protocols are successfully ported to hardware, which supports LoRa protocol, but not limited to LoRa physical layer. The resulting embedded code works accordingly to simulation results and gathered statistics. The developed simulation environment and modern so-called AI tools allowed to shorten dramatically the embedded software development cycle and evaluate algorithm efficiency information from the simulation results before applying on real hardware.
Keywords: swarm simulation, wireless network, digital twin, TDMA-based communication

D.V. Rahozin, V.Ye. Smirnov

МОДЕЛЮВАННЯ І АНАЛІЗ РІВНОПРАВНИХ РОЇВ РОБОТІВ

У статті розглядається повний цикл розробки комунікаційного протоколу на основі TDMA для рою роботів, включно з моделюванням та апаратною реалізацією. Протокол розроблений для забезпечення надійного, захищеного від перешкод передавання даних у мережі. 1) Було розроблено симулятор на основі пакета моделювання SimPy, який дозволяє проводити тисячі симуляцій для підтвердження концепції комунікацій на основі TDMA для рою роботів. 2) За допомогою симулятора розроблено низку методів для передавання даних в рамках фреймів TDMA з метою покращення стійкості мережі. Ці симуляції дозволили зібрати статистику, проаналізувати та вибрати кращий алгоритм. 3) Використано інструменти так званого генеративного штучного інтелекту для створення частин симулятора та додаткові техніки для перетворення коду моделювання у вбудований код з метою наближення до парадигми «цифрового двійника». 4) Змодельовані протоколи успішно перенесені на обладнання, яке підтримує протокол LoRa, але не обмежується фізичним рівнем LoRa. Отриманий вбудований код працює відповідно до результатів моделювання та зібраної статистики на основі моделі. Розроблене середовище моделювання та сучасні інструменти штучного інтелекту дозволили значно скоротити цикл розробки вбудованого програмного забезпечення та оцінити інформацію про ефективність алгоритму з результатів моделювання перед застосуванням на реальному обладнанні.
Ключові слова: моделювання рою, бездротова мережа, цифровий двійник, зв'язок на основі технології TDMA

1. Introduction

Today the swarm of robots or drones is an in-demand technology for field application, as it provides control of multiple robotic devices using only one control center or even allow them to act autonomously following some scenario. For various industrial purposes the basic scenario, when an operator controls only one drone, looks obsolete, as this case requires a dedicated operator for each active drone. The multiple drones use may improve the execution efficiency of many process types, but it requires: 1) big enough number of operators; 2) dedicated control channels which should be separated one from another; 3) maybe the strongest issue - synchronization and controlling operators to execute meaningful tasks over some area without interfering one another. Anyway, the human operator use now is the simplest and the cheapest solution for many cases, as the operator job can be done remotely for

low price. But this benefit cannot be projected for near future. The main goal of our research is to discover the possibilities of building semi-autonomous robot swarms, which act over defined area in peer-to-peer network and may exchange roles under changing environmental conditions and under limitation of robotic resources until the mission is completed. One of the main limitations is the definition of communication protocol, which further should support efficient swarm control protocol.

There are many aspects that restrict the data exchange paradigm in a robotic network, but the first step is the building of inter-robotics communication. We are not going to write down a long list or a large classification of factors that affect the network protocol, but we highlight the aspects most important for our case. One aspect is the limited bandwidth, at least for the autonomous drone scenario usually. We have no requirement for continuous video streaming delivered for human operator, so usually the geographical coordinates, velocity and RSSI are required data to exchange. Another aspect is the maintaining robot network integrity in case of obstacles, jamming and noise in communication channel. The data bandwidth aspect should be considered for establishing tradeoff between being narrow enough against the increased number of drones in the swarm, up to hundreds. All the aspects restrict the tradeoffs in definition of efficient control protocol, which allows the robotic network to reach mission goals with or without operator control. Possibly, some attention should be paid for limiting power consumption – to extend the life of battery-operated drones in "suspend mode" or temporary inactive robot mode. This enables long missions; even weekly mission becomes possible. The proper communication protocol gives a good basis for building a network - with wide variety of underlaying physical and transport layers. Robust protocol is a good base for implementing particular algorithms which automatize robot mission planning and execution.

In chapter 2 we are defining the task and goals for the robotic network concept and discussing its most important properties. In chapter 3 we researching the possible ways to build a model, and describe the model of communication protocol. In chapter 4 we describe simulation results, including gathered metrics for model efficiency and corresponding hardware implementation.

2. Robotic Network Concept

2.1. Protocol concept

Our goal is to define the communication model of robotic network with respect to modern hardware we use further for its implementation. The model is used to evaluate important metrics: 1) swarm recovery time during mission execution in case of jamming/bad link cases; 2) drone swarm reaction time for rebuilding a swarm control software for a new mission; 3) minimum required bandwidth for tasks; 4) much simpler characteristics of the swarm behavior in case of different physical radio transceivers and power supply characteristics. On the next step the developed model allows to evaluate high-level swarm mission scenarios.

It should be noted, that our research echoes the method applicable for ad-hoc networks [1], however the modern robots have less limitations – more energy, less operating time, less restrictions for transmission channels and speeds, less number of operating devices. Still, the robots can move across the network area, compared to practically non-moving sensor. And the main difference – if the earlier sensors devices have quite simple onboard sensors, modern robots have RGB and thermal cameras, radars, lidars, ultrasonic sensors and many other devices. Such device set give the overwhelming information about environment, but still the scenarios of the robot swarm use are quite basic, usually restricting useful scenarios for basic operations, for example, in agricultural sector. We are hopeful that introducing even the basic swarm usage scenarios into industry will help to employ more and more use cases over near time.

For the definition of the protocol concept, we are reviewing several most often used scenarios for the swarm: 1) surveillance scenario, where the swarm is constantly looking for some anomalies over an area; 2) continuous execution of basic tasks for robots – e. g. spraying some agriculture; 3) delivering packages over routes – in case of emergency situation – with possible mission changes "on-the-fly".

All enlisted tasks require communication protocol for swarm orchestration, when all swarm devices share the same quite narrow bandwidth re-source; and a protocol for mission control. For this paper we concentrate on communication protocol: for OSI network layers, we are considering levels 2-3 – Dala Link and Network layers, and partially level 4 – Transport layer. The physical layer for practical evaluation is fixed and the most valuable for today are LoRa standard physical layer.

Practical considerations for scenarios show us, that the communication layer should concentrate at least on the following tasks: 1) providing stable communication between the operator and the swarm; 2) stable communication in case if operator is off-line and the network is autonomous; 3) robustness in case if some nodes fall off-line for short or moderate time; 4) adding new nodes for swarm; 5) joining two separate swarms. Looking back to previously elaborated metrics: coverage, fault tolerance, response time, scalability, throughput [2], robustness, task completion rate [3], precision, success rate, adaptability to scale [4] and so on – we are setting narrower but more complex metrics and goal to simulate, as our scenario set and selected physical link layer mark several metrics as more important. So, we define a set of application-level metrics of our interest, which reflect the components of swarm mission success.

2.2. Physical layer

The most common off-the-shelf communication solution is based on LoRa [5], which looks to be well-known low-power solution for environmental sensors, designed for the long range – up to 15 km distance – applications. On the other side the communication speed looks to be quite low, 0.3 – 50 kBit/sec, but this does not look as a big issue. Earlier we have discussed, that low automation degree of drones requires high bandwidth, as the human operator requires good quality video stream for effective operation, for example current FPV drones` infrastructure is built exactly such way. If we drop video stream, we also drop high bandwidth requirements, moving exact object recognition operations to drone side. This allows to fit communication requirements into strict LoRa band-width, limiting communication to simple exchange of sensor values. The types of sensor samples are: current GPS position, velocity vector, power level, payload weights or value. Basically, the required sensor value types depend on control protocols, which are defined on higher protocol levels, and this requires some iterative process of defining over-the-network control algorithms. We are going to cover several control protocols samples in the next articles, now we concentrate on employing off-the-shelf communication solutions into our swarm.

2.3. Common considerations for communication

Generally, the control protocol does not depend on particular physical layer, but the practical considerations usually point to the cheapest "industry standard" radio transceivers available on market. For the time of writing this article different LoRa devices and modules, working in 433MHz non-licensed range, are the most suitable type of devices available on market. This does not mean that TDMA-based protocol requires LoRa, it can be implemented on any type of radio transmitting devices, where the user can directly control the transmission speed and operating modes of transmitter. For communication protocol planning we should know basic timing delays for switching the transmitter between generally idle mode, receiver mode and transmitter mode. The maximum time necessary for changing transmitter operating mode (e.g. from receive mode to transmission) specifies the time gap between TDMA slots. Other point is the accuracy of local clock, which also affects time gaps between TDMA slots in protocol. The theoretical and practical considerations for time synchronization between connected devices in our network were earlier made in [6], including practical results and hardware implementation. The useful feature is RSSI, which allows to evaluate the received signal strength, so we are able to evaluate the distance between devices and possible device movement.

It should be noted, that robotic communication networks are developed having in mind the target structure of the network and use scenarios. The initial use scenarios for swarm define the complexity of the network

physical layer protocol. Basically, we can separate the network types into the following large groups: 1) permanent configuration, where robots practically are not moved; 2) permanent configuration where robots can move within the limits of their defined areas, but leaving the communications between neighbors practically unchanged – with minor distance change; 3) configuration where robots can freely move across area. The corner cases of the last type is the subdivision of the swarm into several swarm with communication configuration rebuild or joining several swarms into one swarm. All the network types also are challenged the problem of network nodes, that can temporary be offline, so leaving the network for short time and joining it after the leave. The effective solution of corner cases greatly improves the overall performance of the network, and one of goals of our study is the modeling of these corner cases scenario.

For our study we chose the scenario, where the robots located in the geographical center of the swarm can communicate to all the robots in the swarm. For moderate number of robots (80-200) we are employing TDMA-based techniques, where we can effectively divide time resource into the defined number of time slots and give fixed amount of outcoming traffic for each robot per time slot. The first useful work, describing TDMA protocol for mobile devices was described in [7], where multi-hop mobile devices network concept for low-speed communications was described. Also, it was proved [6] that off-the-shelf and low cost quartz resonators can provide time-synchronization for robotic network. These techniques enable to design various types (multi-hop, one-hop) of networks, based on TDMA communication layer techniques which can employ LoRa physical layer.

2.4. LoRa communications considerations

We start from considerations, that in order to simplify TDMA network, we can employ peer-to-peer network concept over a comparatively large geographical area, as maximum distance for LoRa communication is up to 15 km. One of the algorithmic improvements we can employ for this case – the selection of central zone (fig. 1), where the devices can reach all the devices in the network, but other devices possibly cannot communicate directly to peers located too far from them. Fig. 1 shows the robots located on the semi-rectangular field, the central zone devices (black squares) can communicate to all the devices in the network and, at least, one of them has a kind of bridge to Internet network for reporting swarm statistics and receiving control commands. Fig. 1 shows additionally 2 devices, marked with 1 and 2, and border limits – dashed lines, 1-limit and 2-limit, which shows the communication range limits for devices, marked 1 and 2. So, comparing to TDMA-based peer-to-peer communication networks, the central zone controls the TDMA protocol and slots. Other devices, competing for slots in TDMA network, does not "hear" all the devices.

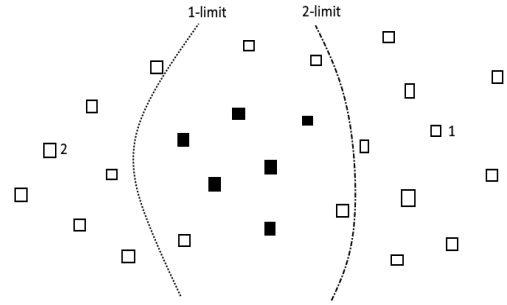

Fig. 1. Two-zones robotic network structure

Consider the TDMA frame structure for IoT devices, shown at fig. 2. The reader may refer to [5] for comparing to simpler implementation.

Slot X	Slot T_0	…	Slot T_K	Slot 0	Slot 1	…
	Slot N-1	Slot N	Slot R_0	…	Slot R_M	

Fig. 2. TDMA frame structure

The total length T_{frame} of the TMDA frame at fig. 2 is fixed, and this time may vary from 0.5 sec to 15 sec, which depends on necessary communication characteristics of the network. T_{frame} shows the typical time of spreading information in the network; for the

network with central zone at fig. 1 this time grows to 2*T_{frame}. For peer-to-peer network this time looks very flexible in terms of control protocol.

The TDMA frame may include: start slot X, which is employed for adding an additional robot or device to the network. Slots T_0-T_K may be used for optimizing of the adding new devices into the network – they extend slot X. Slots 0-*N* are used for communicating between devices, so basically each device owns one slot, optimized scenarios employ multiple slots for one device to increase bandwidth. For our case the value of N starts from 30-40 and finishes near value 200. Some scenarios introduce device priorities, so that high-prioritized devices use several slots for communication. Slots R_0-R_M are used for additional traffic optimizations.

As LoRa transmitters supports RSSI value, which renders approximate distance between receiver and transmitter, the robot approximately can have information about distance between devices, GPS information allows to track device movement and control the so-called density of the devices over a field. This also can be used for specifying communication speed into separate slots and optimizing the traffic inside network. For the simplest applications, only slot X and slots 0-N are used. Slots Ti and Ri can be allocated inside slots 0-N, employing different optimization techniques, which are out of discussion scope now.

3. Swarm simulation concept

3.1. Metrics discussion

The proposed TDMA technique looks basic, but there are several algorithm parts, which are considered quite complex and need to be observed and proven inside simulation. The usual communication procedure looks simple, but the most interesting issues are forming the network and reentering the network after communication signal jams. Communication jams is a short-range or long-range signal obstacles which prohibits communication for the part of the network. In case if the jam is quite long in time, there is a possibility that after such a jam the network need to build the network from scratch. The scenarios of joining two swarms into one or separating one swarm are also targets for our simulation.

3.2. Simulation toolkit

The main goals of the simulation not only for our study are at least 1) to have an environment suitable for observing and analyzing the algorithms; 2) provide a cheap alternative for direct use of hardware; 3) provide a bridge between simulated environment and scenarios – the code in usual case – and the real hardware. Additional point is to have ability to switch between hardware platforms without conceptual coder changes. It should be noted that we are not considering computing hardware of the robotic system, as communication protocol usually requires less than 1% of overall system computing power.

The first common choice for such kind of simulator is NS-3[8], or OM-NeT++ [9]. These simulators and a bunch of other simulators, functionally close to it, are event-driven simulators, which have a rich number of extensions, able to simulate practically every communication protocols. Still the older our work [6] clearly shows, that for our study much lighter tools can be involved. Our final choice was LoRaSim[10] from Lancaster University. Its functionality looks quite basic, but our study showed us that this choice was right: the simulator is lightweight and cheap in term of resources necessary to learn it and use it. As we use LoRa hardware as a tool, we do not need to track much of LoRa protocol internals. All TDMA protocols can be based on transmitters, which are able to receive a packet of defined length and provide a packet of defined length. For robustness we need the ability to change underlaying protocol without redoing the system. So, LoRa simulator [10] is used as an interchangeable component and a physical layer, which can be changed to another physical layer implementation, simply connecting channel layer in OSI model to the physical environment.

3.3. SimPy implementation benefits

The protocol simulation is based on the SimPy [11] – process-based discrete-event simulation framework based on standard Python. This choice is based on its simplicity, as

1) standard Python infrastructure is used; 2) simulation is supported by shared objects and synchronization component libraries in SimPy; 3) the simulation code flow in SimPy is a Python process without specific definitions and infrastructure, which makes the SimPy use also cheap and simple in learning how to use and apply.

The use of common programming language as Python showed us many benefits for our case. First, the simulated process algorithmics, which implement various TDMA-network support parts, is implemented as a well-structured programming code. We even may utilize the concept of "digital twins", directly translating the simulation code into the robotic code. Despite the fact, that Python interpreter makes the Python code slower 10-20x times that C++ code, the modern AI tools – such as commonly used ChatGPT or Gemini tools - allow seamlessly convert Python code to C++ code. The underlying protocol concept employs statically allocated components – data buffers, arrays, variables and so one, as the simulated process is targeted for embedded platforms. The AI tools use allows to overcome the barrier, which was introduced long time ago by using different programming languages for different kind of simulations – as all fast simulations employ C++ optimized code, for example ROS-2 [12]. Its simulation concept is an exact "digital twin" system, versus more easily written but more slow systems in Python, such as SimPy. Now AI tools allow to rewrite still with some limitations the simulation code from one language to another, simplifying and extending the "digital twin" concept use. Sure, that AI tool now cannot deal with optimized driver code for LoRa, but the underlying low level code base is not the simulator part. It is separated from our simulated algorithms with abstraction layer, so we are able to convert simply between languages even using AI tools. For developers, who are still worried with automated code conversions – it is quite possible to get to mature Python-based model, convert it to C/C++, verify differences between original and converted code flow, and in case of success – move further to embedded code.

Another important point for the SimPy use is that the Python-based development and process-based simulator helps to use AI code generation tools efficiently, as it can provide the whole algorithmic base for Python effectively even if the developer of the simulation algorithm is not a proficient Python programmer. Although the "digital twin" concept cannot be used directly for our model, the common code structure for object (robot) behavior is a common control code with time-synchronization primitives and yield instructions, which allows to synchronize the object behavior with model time. During the conversion of the model code into the hardware-side code, only these synchronization primitives need to be changed into hardware-related code. However, the underlying real-time support library can provide the corresponding compatibility layer with SimPy synchronization primitives. All the other control code can be built using AI tools and we have widely practiced to use AI tools while prototyping the simulation code.

4. Simulation details and results

We have implemented our simulation ecosystem on the top of LoRaSim [10] and SimPy [11] software packages, and implemented our TDMA algorithmics using this simulation engine. Our implementation is well aligned with concept used in ROS and ROS-2 [12] simulation, where the robotic control software is build using "digital twin" concept. This concept suppose that the simulation process is identical to control program for the real robot or drone, so the ROS simulator provides all necessary functionality as system libraries, including alternate versions of flight controller.

To be complete with moving the simulation results to the real hardware, Raspberry Pi boards with LoRa transceiver E32-433T30D was used, which is connected by UART link. As expected, real hardware elaborates simulated algorithms as expected.

4.1. Simulation engine

Despite of increased computational complexity - as proper physics simulation for drone engine should be computed during simulation cycle – the "digital twin" approach works well for the modern computing hardware. Our solution provides real time simulation for dozens of drones, including the render-

ing of drone operations using 3D-engine via Unity, and this even emulates video stream for opera-tor reference. We successfully approached the "digital twin" concept, except we have the main loop implemented in Python. The structure of SimPy-based code reflects the structure of real-time control program for drones, so the dropped "digital twin" paradigm component – the same programming language – is covered with AI tools which make translation between Python and C language. We introduce some internal limitations for simulation code – such as static memory allocation and simpler code structure – and current AI tools work well enough during program mapping to C language. We are not going to discuss exact over-head introduced by language translation, this may be a separate investigation mainly in the area of program code quality and code base management.

Due to nature of the model, we have clear layers of the code, which simplify object synchronization in model and coding practices. The bottom layer includes LoRaSim simulator, which is used mainly for physical layer of data transmission, so the practical synchronization of robot's work is done by LoRaSim layer. As discussed in previous chapter, we can change LoRaSim layer to any other wireless simulation layer, as basically we need the following functionality: 1) packet receive; 2) packet transmission; 3) RSSI value as a part of packet receive; 4) setting speed and transmission power values. Any wireless protocol simulator that provides this functionality can be used instead LoRaSim. TDMA frame forming is based on SimPy timer functionality, which can be easily mapped on any hardware. The TDMA frame formation principle is inspired by ideas from old work [6], lowering the complexity of multi-hop system into two-hop network or star-based network. Additionally, LoRaSim is slightly modified to provide tranmission jamming and simulation programmable packet loss.

4.2. Simulation goals and metrics

On the top of TDMA frame formation we simulate a control program, which includes the following main parts:

1. Initial forming of TDMA network, when the robotic swarm should define priorities, example metric – the number of visible devices or RSSI of root node which has link to operator. Operator link is low-speed and transfers basic swarm statistics, goal reaching results and passes operator commands to swarm devices. We are not considering radio transmitter power consumption as it usually less than 1% of overall power consumption on any flying drone. One of modeling goals was to observe different scenarios of network forming, allowing to improve the speed of robot joining the swarm by extending the functionality of slot X (fig. 2). Our model allows to improve speed of network forming 4x-8x times, depending on slot X length and configuration.

2. Link loss correction, which has two cases: short time link loss for several frames, where the robot should not leave the network and save his slot(s) active; and long-time link loss, when the algorithm is similar to initial network forming. This is the most interesting part, as the metric of time, necessary for forming the TDMA-based network, is the main metric for cases, when swarm works around concrete buildings which shield the signal from central drones, so the drone may accomplish some mission in offline, join the network again and send gathered data to robots, which have connection to main network.

The most interesting metric we have analyzed was the time of the network reconstruction, when the network structure is collapsed after a long signal jam. We have applied different optimization techniques for minimization of the number of TMDA frames necessary for network reconstruction, which are not the topic of our paper, and our simulation engine is able to run thousands of network reconstruction scenarios with the results in table 1.

Table 1.

Network reconstruction times for basic and optimized TDMA protocol.

Number of nodes in network	Construction frames for the basic period	Construction frames for the optimized period
5	7	3
12	16	6
20	25	7
40	45	13
80	87	25

So, the numbers in table 1 clearly shows that our simulator allows us to construct and debug algorithms quickly, as the brute force approach to develop some algorithms just on hardware simply not working. Also we are not using any techniques of formal verification due to its complexity, so we need the big number of simulations with randomizing conditions in start and randomizing jamming for each network configuration. Our simulation allows us to have thousands of runs for the network with some constant number of nodes.

4.3. Hardware platform notes

Sure, that the first move of the simulation to hardware platform introduces some problems, related to first improper assumptions on timings for radio transceiver. However, after initial timings correction, the simulation code runs at the platform as we expected. We use frequency band 433MHz, available for radio amateurs, and the limit to 0.5W output power allows to debug the protocol without specific restrictions.

For hard debugging cases we may use a "sniffer" hardware, which is a unit which always receives data from our wireless network, and logs the TDMA frame structure, frame/packets timestamps and data, so it may be compared with simulation log.

It should be noted, that the simulation of protocol and its different optimized version is the only way to make an embedded system in appropriate time. Even if hardware platform Raspberry Pi 4 looks developer friendly, the protocol debugging is a nutshell, as requires elaboration of results and its analysis after each run. Also, it is practically impossible to provide algorithmic debugging via multiple runs, as the communication speed in LoRa protocol makes scenarios run time extremely long. Also, it is quite hard to provide jamming for LoRa in real life.

We are not mention real hardware tests at large distances. Our previous experience with transceivers shows that in real life the transmission distances, signal power and speeds closely resemble the hardware manuals, so the operational range of our transceivers reaches the maximum, described in manuals. Our simulation engine uses RSSI information and introduces random errors due to communication range, so the simulation results are similar to the real-life scenarios.

5. Conclusion

Our research clearly shows that quite simple simulation system can greatly speed up the development of robotic networks with wireless communication abilities. We developed a SimPy-based simulation, which allows to develop and debug TDMA-based protocol for self-organized robotic swarm based on LoRa wireless hardware, and the protocol has advanced abilities to reconstruct network in case of signal jams. The developed algorithms can be comparatively easily transformed, sometimes with helps of current AI tools, into embedded computers, equipped with LoRa hardware, but not limited to LoRa. We have not noticed important differences in functioning of the simulator-based system and its hardware counterpart, so we are close to name these systems as digital twins. The most important is the shortening the development cycle of communication system several times, up to our experience in embedded systems development.

References

1. Agrawal R., Faujdar N., Romero C.A.T., Sharma O., Abdulsahib G.M., Khalaf O.I, Mansoor R.F., Ghoneim O.A. Classification and comparison of ad hoc networks: A review. // Egyptian Informatics Journal, Vol. 24, Issue 1, 2023, p. 1-25, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2022.10.004.
2. Kimon P. Valavanis, George J. Vachtsevanos. Handbook of Unmanned Aerial Vehicles. Springer, 2014. 3022 p. https://doi.org/10.1007/978-90-481-9707-1
3. Mohsan, S.A.H., Othman, N.Q.H., Li, Y. et al. Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends. Intel Serv Robotics 16, 109–137 (2023). https://doi.org/10.1007/s11370-022-00452-4
4. Mueller, M., Smith, N., Ghanem, B. (2016). A Benchmark and Simulator for UAV Tracking. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. LNCS, vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_27

5. Paredes W.D., Kaushal H., Vakilinia I.; Prodanoff Z. LoRa Technology in Flying Ad Hoc Networks: A Survey of Challenges and Open Issues. Sensors 2023, 23, 2403. https://doi.org/10.3390/s23052403
6. Rahozin D. (2008) Modeling synchronised sensor networks // Problems in Programming, special issue, 2008, #2-3. P. 721-729. https://nasplib.isofts.kiev.ua/handle/123456789/2155
7. Kanzaki A., Uemukai T., Hara T., Nishio S.. Dynamic TDMA Slot Assignment in Ad Hoc Networks // In Proc. 17th Int. Conf. on Advanced Information Networking and Applications (AINA'03), 2003. – P. 330–336.
8. Campanile L., Gribaudo M., Iacono M., Marulli F., Mastroianni, M. Computer Network Simulation with ns-3: A Systematic Literature Review. Electronics, 9(2), 2020, p. 272. https://doi.org/10.3390/electronics9020272
9. Virdis A., Kirsche M. Recent Advances in Network Simulation The OMNeT++ Environment and its Ecosystem: The OMNeT++ Environment and its Ecosystem, 2019. ISBN 978-3-030-12841-8. https://doi.org/10.1007/978-3-030-12842-5.
10. Voigt T., Bor M., Roedig U. Alonso, J. Mitigating Inter-Network Interference in LoRa Networks. In EWSN '17 Procs. of the 2017 Int. Conf. on Embedded Wireless Systems and Networks (pp. 323-328). ACM Press. http://dl.acm.org/citation.cfm?id=3105395
11. SimPy homepage, https://simpy.readthedocs.io/en/latest/, last accessed 2025/08/02.
12. Haridevan A., Kang J., Yuan M., Shan J. ROS2-Gazebo Simulator for Drone Applications. In Proc. of 2024 Intl. Conf. on Unmanned Aircraft Systems (ICUAS), p. 1232-1238. https://doi.org/10.1109/ICUAS60882.2024.10556903.

Дата першого надходження до видання: 16.04.2026
Внутрішня рецензія отримана:02.05.2026
Зовнішня рецензія отримана:08.05.2026
Дата рекомендації до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Рагозін Дмитро Васильович,
кандидат технічних наук,
старший науковий співробітник
Ragozin Dmytro,
Ph.D. (technical sciences),
senior researcher
http://orcid.org/0000-0002-8445-9921.

Смірнов Валентин Євгенович,
аспірант
Smirnov Valentyn,
post-graduate student
http://orcid.org/0009-0006-4022-5951.

Місце роботи авторів:

Інститут програмних систем
НАН України
Institute of Software Systems of the
National Academy of Sciences of Ukraine
тел. +38-044-522-62-42
E-mail: ukrprog@isofts.kiev.ua
www.iss.nas.gov.ua

УДК 004.056.53:004.89:621.391.82 https://doi.org/10.15407/pp2026.02.067

Г.В. Шуклін, О.В. Барабаш, А.Б. Гребенніков, І.Д. Данилов, Ю.В. Пепа

АНАЛІЗ СТІЙКОСТІ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ВИЯВЛЕННЯ ЗАСОБІВ НЕСАНКЦІОНОВАНОГО ОТРИМАННЯ ІНФОРМАЦІЇ В УМОВАХ НАВМИСНИХ ЗАВАД

У роботі запропоновано новий науковий підхід до визначення умов стійкості процесів виявлення технічних засобів несанкціонованого отримання інформації (ТЗНОІ) в умовах впливу навмисних завад. Показано, що стійкість інтелектуальної системи виявлення навмисних завад залежить від часу затримки, який враховується системою диференціальних рівнянь із запізненням, які описують інертність каналів моніторингу, затримки обробки сигналів та адаптивну динаміку параметрів, що характеризують наявність навмисних завад. Для аналізу стійкості системи застосовано овал Рихлінського як інструмент дослідження спектральних властивостей характеристичного квазіполінома систем диференціальних рівнянь із запізненням. Отримано нову умову стійкості процесів виявлення (СПВ) за наявності змінних затримок та активної протидії з боку порушника. Проведено аналітичний та кількісний аналіз динаміки системи, що демонструє ділянки стійкого та нестійкого функціонування інтелектуального модуля виявлення.
Ключові слова: захист інформації, навмисні завади, диференціальні рівняння із запізненням, овал Рихлінського, стійкість, інтелектуальний моніторинг, технічні канали витоку інформації

G.V. Shuklin, O.V. Barabash, A.B. Grebennikov, I.D. Danylov, Yu.V. Pepa

ANALYSIS OF THE RESILIENCE OF AN INTELLIGENT SYSTEM FOR DETECTING MEANS OF UNAUTHORIZED INFORMATION ACCESS IN THE FACE OF DELIBERATE INTERFERENCE

This paper proposes a new scientific approach to determining the stability conditions for processes aimed at detecting technical means of unauthorized information acquisition (TMAIA) under the influence of deliberate jamming. It is shown that the stability of an intelligent system for detecting intentional jamming depends on the delay time, which is accounted for by a system of delay differential equations describing the inertia of monitoring channels, signal processing delays, and the adaptive dynamics of parameters characterizing the presence of intentional jamming. To analyses the stability of the system, the Rychlinski oval is used as a tool for investigating the spectral properties of the characteristic quasi-polynomial of systems of differential equations with delay. A new stability condition for detection processes (SPV) is obtained in the presence of variable delays and active countermeasures by the attacker. An analytical and numerical analysis of the system's dynamics has been carried out, demonstrating the regions of stable and unstable operation of the intelligent detection module.
Keywords: information security, deliberate interference, differential equations with delay, Rychlinski's oval, stability, intelligent monitoring, technical information leakage channels

Вступ

Сучасні системи захисту інформації функціонують в умовах складної електромагнітної та інформаційної обстановки, що характеризується активним застосуванням засобів радіоелектронного придушення, генерацією маскувальних шумів, динамічними змінами параметрів каналів витоку та прихованим функціонуванням стійкості процесів виявлення (СПВ). Особливу небезпеку становлять технічні засоби несанкціонованого отримання інформації, здатні здійснювати приховану передачу даних, адаптуватися до умов середовища, змінювати параметри корисного сигналу.

Класичні моделі виявлення базуються переважно на статичних імовірнісних підходах і не враховують інертність процесів аналізу, тимчасові затримки обробки, накопичувальний ефект перешкод та нелінійну адаптацію порушника.

У зв'язку з цим виникає необхідність розробки динамічних моделей, що враховують запізнення та стійкість процесів виявлення.

Аналіз останніх досліджень

Проблематика виявлення ТЗНОІ в умовах навмисного деструктивного впливу є міждисциплінарною, оскільки поєднує питання технічного захисту інформації, радіомоніторингу, радіоелектронної протидії, статистичного аналізу сигналів, машинного навчання та теорії стійкості динамічних систем із запізненням. Сучасні дослідження в цій сфері свідчать, що класичні статичні підходи до виявлення каналів витоку інформації вже не повною мірою відповідають умовам функціонування сучасних об'єктів інформаційної діяльності, де порушник здатний адаптувати параметри сигналу, створювати імітаційні завади, змінювати спектральні характеристики випромінювання та впливати на роботу інтелектуальних засобів моніторингу.

У роботі [1] розглянуто моделювання процесів виявлення ТЗНОІ в умовах навмисного впливу завад. Автори акцентують увагу на тому, що процес виявлення ТЗНОІ повинен аналізуватися не лише з позиції ймовірності правильного виявлення, а й з урахуванням динаміки зміни ознак сигналу, впливу завад, часових затримок обробки та адаптивної поведінки порушника. В [2, 3] розглянуто адаптивні методи протидії активним шумовим завадам та методи протидії у радіонавігаційних конфліктах. У цих дослідженнях показано, що навмисні завади можуть мати не лише шумовий, а й адаптивний або імітаційний характер. Вони здатні змінювати частотні, часові та енергетичні параметри залежно від поведінки системи виявлення. Це особливо важливо для задач виявлення ТЗНОІ, оскільки порушник може не просто маскувати сигнал, а формувати такі завадові впливи, які спричиняють хибні спрацьовування або знижують достовірність ухвалення рішень інтелектуальним детектором. Автори в [4] досліджують ефективність функціонування систем передачі інформації з хаотичними сигналами та OFDM-модуляцією в умовах впливу навмисних завад. Значущість цього джерела полягає в тому, що автори розглядають вплив завад на складні сигнальні структури, які мають нелінійні та спектрально розподілені властивості. Для задач виявлення ТЗНОІ це має принципове значення, оскільки побічні електромагнітні випромінювання, приховані радіоканали або маскувальні сигнали можуть мати складну спектральну структуру, що ускладнює їх ідентифікацію класичними методами спектрального аналізу.

Окремий напрям досліджень пов'язаний із технічними каналами витоку інформації. Так у роботі [5] проаналізовано технічний канал витоку інформації через побічні електромагнітні пере- випромінювання допоміжних технічних засобів і систем. Це дослідження важливе для формування фізичної основи моделі виявлення, оскільки наявність ТЗНОІ або каналів витоку часто проявляється через амплітудні, частотні, просторові та часові ознаки електромагнітного випромінювання. Саме такі ознаки можуть бути використані як вхідні параметри для інтелектуального детектора. У роботі [6] запропоновано алгоритми вимірювання частоти кадрової розгортки моніторів для частотно-вибіркового придушення каналів витоку інформації. Це джерело демонструє практичний підхід до виявлення та придушення конкретного типу технічного каналу витоку. Його доцільно використати для обґрунтування того, що ефективне виявлення ТЗНОІ потребує не лише реєстрації факту випромінювання, а й аналізу його частотно-часових характеристик, які можуть бути змінені або замасковані під дією навмисних завад. Дослідження [7] присвячене аналізу та моделюванню джерел радіоелектронної боротьби з урахуванням просторово-частотного орієнтування. Значення цієї роботи полягає в тому, що вона враховує не лише спектральні, а й просторові характеристики джерел завад. Для інтелектуальної системи виявлення ТЗНОІ це є важливим, оскільки локалізація джерела сигналу або завади може бути одним із ключових критеріїв ухвалення рішення. Просторово-частотне моделювання дозволяє підвищити достовірність розмежування корисного сигналу, побічного випромінювання та навмисної завади. У роботі [8] розглянуто нейромережеву модель оцінювання рівня захищеності

складнозашумленої мовної інформації. Це дослідження є важливим з позиції використання інтелектуальних методів аналізу сигналів в умовах високого рівня шумів. Нейромережеві підходи дають можливість виявляти приховані закономірності в даних, які важко формалізувати класичними аналітичними методами. У контексті даної статті це підтверджує доцільність використання інтелектуального детектора, який поєднує спектральні, статистичні, кореляційні та ймовірнісні ознаки.

Серед іноземних досліджень важливе місце займає робота [9], в якій запропоновано методи виявлення jamming-атак у мережах IEEE 802.11 на основі машинного навчання. Автори показують, що ML-методи можуть ефективно розрізняти нормальний режим роботи мережі та режим навмисного радіоелектронного придушення. Це є важливим для обґрунтування інтелектуального компонента системи виявлення, оскільки воно демонструє практичну ефективність машинного навчання у задачах ідентифікації навмисних завад. У праці [10] досліджено фізичну автентифікацію супутникових передавачів із використанням глибинного навчання. Автори використовують ознаки фізичного рівня сигналу для ідентифікації джерел випромінювання. Отже, застосування глибинного навчання до фізичного рівня сигналу є перспективним напрямом для підвищення стійкості систем виявлення. У роботі [11] розглянуто захист когнітивних радіомереж від інтелектуального постановника завад, який адаптує свою поведінку залежно від реакції системи. Це дослідження є особливо важливим для даної статті, оскільки в ній також розглядається не звичайний випадковий шум, а саме інтелектуальна завада, яка аналізує роботу детектора та намагається знизити його ефективність. Такий підхід дозволяє розглядати протидію як динамічний процес взаємодії системи захисту та порушника. Дослідження [12] присвячене виявленню атак фальсифікації даних спектрального зондування у мобільних когнітивних радіомережах із використанням методів штучного інтелекту. Показано, що загроза може бути пов'язана не лише з фізичним придушенням сигналу, а й з навмисним викривленням даних, на основі яких система ухвалює рішення. У контексті виявлення ТЗНОІ це означає, що інтелектуальний детектор повинен бути стійким не тільки до шумових завад, а й до фальсифікації або імітації ознак витоку інформації.

У роботі [13] систематизовано методології та виклики використання машинного навчання для гарантування безпеки спільного використання спектра. Автори аналізують атаки на спектральне зондування, включаючи імітацію легітимного користувача та фальсифікацію результатів вимірювання. Зазначимо важливість захисту систем моніторингу від адаптивних і маскувальних впливів, які можуть порушувати достовірність виявлення. В [14] досліджено ефективність frequency hopping як методу протидії jamming-атакам у мережах IEEE 802.11. Це джерело є важливим для розуміння традиційних методів зниження впливу навмисних завад. Разом з тим, воно показує, що навіть ефективні методи частотної перебудови мають обмеження в умовах адаптивного порушника. Це підтверджує необхідність створення більш складних інтелектуальних систем виявлення та протидії, які враховують динаміку завадового впливу.

Математичну основу дослідження стійкості систем із часовими затримками розкрито у працях [15, 16]. В [15] запропоновано покращені критерії стійкості для нейронних мереж із запізненням на основі функціоналів Ляпунова–Красовського, а в роботі [16] досліджено стійкість мережевих систем керування з урахуванням затримок передавання та втрат пакетів. Це положення є близьким до задачі аналізу інтелектуального детектора, в якому затримки обробки, інерційність каналів моніторингу та несвоєчасне оновлення параметрів можуть призводити до втрати стійкості процесу виявлення.

Таким чином, аналіз наукових джерел показує, що на сьогодні достатньо ґрунтовно досліджено окремі аспекти проблеми: технічні канали витоку інформації, методи радіоелектронної протидії, виявлення jamming-атак, застосування машинного навчання до аналізу сигналів, а також математичні критерії стійкості систем із

запізненням. Водночас недостатньо досліджено питання комплексного поєднання цих напрямів у межах єдиної математичної моделі стійкості інтелектуальної системи виявлення ТЗНОІ в умовах навмисних адаптивних завад. Саме це визначає наукову доцільність розробки моделі на основі систем диференціальних рівнянь із запізненням та застосування геометричних критеріїв аналізу стійкості, зокрема овалу Рихлінського.

Мета статті

Метою даної статті є розробка та наукове обґрунтування математичної моделі стійкості процесів виявлення технічних засобів несанкціонованого отримання інформації (ТЗНОІ) в умовах навмисного деструктивного впливу з використанням систем диференціальних рівнянь із запізненням та апарату овалу Рихлінського для аналізу стійкості інтелектуальної системи ухвалення рішень щодо виявлення витоку інформації технічними каналами.

Постановка проблеми

Нехай система захисту здійснює моніторинг об'єкта інформаційної діяльності. Необхідно визначити:

1. Стійкість процесу виявлення;
2. Вплив затримок;
3. Вплив адаптивних перешкод;
4. Умови втрати виявлення.

Введемо наступні позначення:

$x_1(t)$ – інтегральна міра ознак, за якими система виявлення навмисних завад спроможна ідентифікувати наявність технічного каналу витоку інформації;

$x_2(t)$ – змінна, яка характеризує інтенсивність навмисного деструктивного впливу на систему виявлення;

$x_3(t)$ – інтегральна оцінка інтелектуального детектора, структурну схему якого представлено на рис. 1, і яка характеризує внутрішній стан інтелектуальної системи ухвалення рішень.

У табл. 1 представлено перелік ознак для технічних каналів витоку інформації.

Таблиця 1.

Ознаки витоку інформації по технічним каналам

Вид технічного каналу	Ознаки наявності витоку
1. Радіотехнічний канал	А) Амплітуда паразитного випромінювання Б) Стійкі спектральні піки В) Особливості модуляції Г) Гармонічні складові
2. Акустичні і віброакустичні канали	А) Кореляція вібрацій Б) Особливі частоти В) Часові шаблони
3. Побічне електромагнітне випромінювання	А) Рівень побічного електромагнітного випромінювання Б) Просторова локалізація джерела В) Синхронізація з обчислювальними процесами
4. Мережеві приховані канали	А) Аномалії трафіку Б) Нетипова періодичність В) Приховані часові кореляції

У запропонованій моделі розглядаються саме інтелектуальні перешкоди. На відміну від звичайного шуму, який є або стаціонарним, або випадковим і який не адаптується, інтелектуальна перешкода аналізує поведінку системи захисту, змінює свій спектр для імітації корисного сигналу, створює складні кореляції і намагається викликати хибні тривоги.

В табл. 2 представлено види інтелектуальних перешкод, які розглядаються в моделюванні процесів виявлення технічних засобів несанкціонованого отримання інформації в умовах навмисного впливу завад.

Таблиця 2.

Основні види інтелектуальних перешкод

Види інтелектуальних завад	Характеристики завад
1. Активні електромагнітні	А) Широкосмугове зашумлення Б) Імпульсні завади В) Адаптивна зміна частоти
2. Імітаційні	А) Генерація хибних спектральних ознак Б) Копіювання структури сигналу ТЗНОІ
3. Когнітивні	А) Аналіз роботи детектора Б) Динамічна адаптація В) Формування маскуючих послідовностей за допомогою нейронних мереж

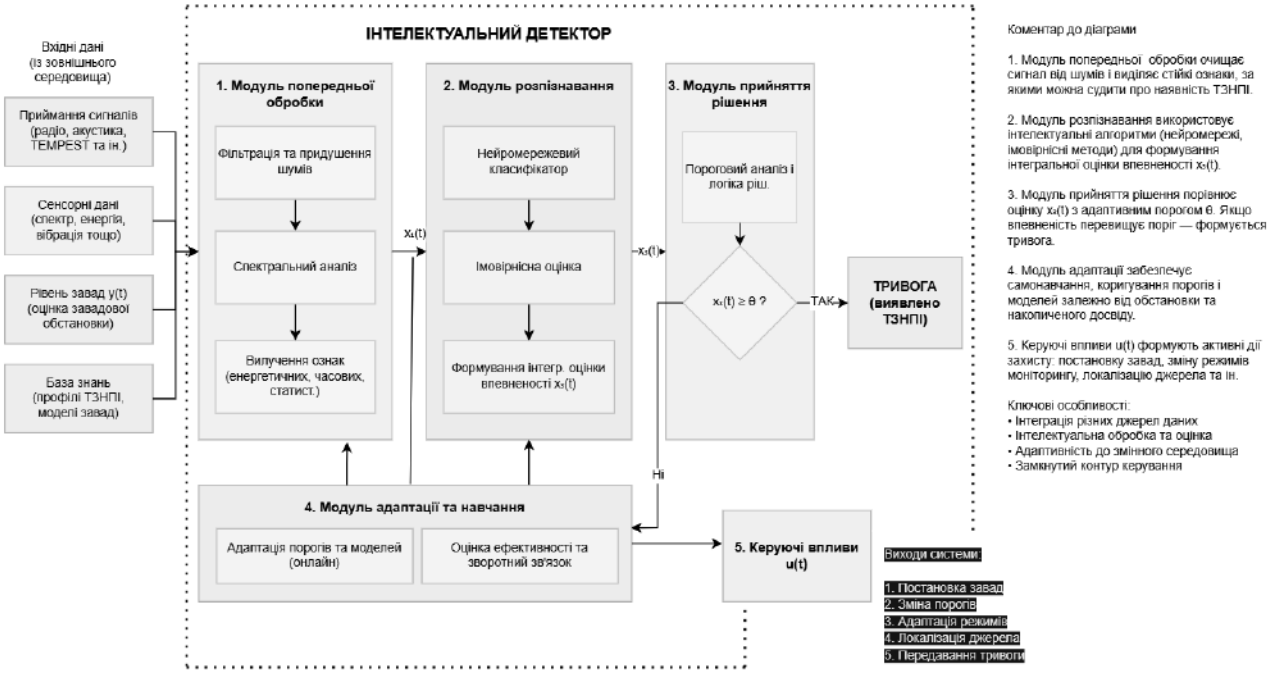

Рис. 1. Структурна схема інтелектуального декодера виявлення ТЗНОІ

За допомогою цієї оцінки рішення не є бінарним, тобто «виявлено/не виявлено». Ця оцінка характеризує впевненість системи в правильності ухвалення рішення в реальному режимі часу. Ця оцінка включає в себе спектральні ознаки, часові аномалії, статистичні відхилення, класифікатори, створені за допомогою нейронних мереж, імовірнісні оцінки і кореляційні показники.

τ – затримка обробки (час запізнення). Ця величина є одним з небезпечних факторів, позаяк в реальних системах миттєвого аналізу не існує.

Основна задача полягає в створені взаємозв'язку динамічних змінних $x_1(t)$, $x_2(t)$, $x_3(t)$ з урахуванням часу запізнення τ за допомогою якої інтелектуальний детектор прийняття рішення виявляє ТЗНОІ.

Викладення основного матеріалу

Математична модель, яка зазначена в меті даної статті, формується системою трьох диференціальних рівнянь з запізненням.

Перше диференціальне рівняння описує динаміку корисного сигналу і має наступний вид

$$\frac{dx_1(t)}{dt} = a \cdot x_1(t) - c \cdot x_2(t) \cdot x_1(t) - b \cdot x_1(t-\tau) + u(t) \ , \tag{1}$$

де a – коефіцієнт підсилення ознак; b – коефіцієнт деградації внаслідок часу затримки τ;
c – коефіцієнт придушення навмисних завад.

Друге диференційне рівняння описує динаміку інтелектуальної (навмисної) завади і має вид:

$$\frac{dx_2(t)}{dt} = d \cdot x_2(t) + g \cdot x_3(t-\tau) + f \cdot \sin(\omega \cdot t) \ , \tag{2}$$

де d – коефіцієнт затухання; g – коефіцієнт адаптації завад; $f \cdot \sin(\omega \cdot t)$ – штучний шумовий вплив.

Третє рівняння моделює інтелектуальний модуль виявлення завад і має наступний вид

$$\frac{dx_3(t)}{dt} = -n \cdot x_2(t) + k \cdot x_3(t) + m \cdot x_1(t-\tau) + l \cdot \sigma(x_1(t)) \ , \tag{3}$$

де n – параметр придушення завадою, який характеризує ступінь руйнівного впливу завад на інтелектуальний детектор.

Перший доданок рівняння (3) визначає деградацію оцінку виявлення ТЗНОІ під впливом завад. Інакше кажучи, цей параметр визначає, наскільки система захисту вразлива до такої завади;

k – параметр затухання інтелектуальної оцінки, який характеризує швидкість втрати накопиченої впевненості інтелектуального детектора. Другий доданок правої частини рівняння (3) визначає істотне затухання внутрішнього стану системи. Інакше кажучи, інтелектуальна система не повинна нескінченно «пам'ятати» старі ознаки. Якщо підозрілі сигнали зникли, то впевненість має поступово зменшуватися, інакше інтелектуальний детектор почне генерувати помилкові сигнали тривоги, стане інертним і втратить здатність до адаптації;

m – параметр чутливості до істинних ознак ТЗНОІ, який визначає наскільки сильно інтелектуальний модуль реагує на ознаки ТЗНОІ. Третій доданок правої частини рівняння (3) описує посилення істотних ознак ТЗНОІ. Інакше кажучи, цей параметр є параметром «уваги» системи до виявленого сигналу. Чим більше значення цього параметра, тим сильніший вплив ознак ТЗНОІ, швидше зростає впевненість інтелектуального детектора і вища ймовірність виявлення ТЗНОІ;

l – параметр інтелектуального нелінійного підсилення, який керує інтелектуальним накопиченням впевненості інтелектуального детектора у виявленні істинних ознак ТЗНОІ:

$$\sigma(x_1(t)) = \frac{1}{1 + e^{-x_1(t)}} \ . \tag{4}$$

Сигмоїда (4) моделює нейронну активність інтелектуального детектора, а також імовірність ухвалення рішень і м'який поріг. У разі малого значення параметра l система диференціальних рівнянь, яка складається з рівнянь (1) – (3), майже лінійна, що свідчить про слабку інтелектуальність детектора і неякісне виявлення прихованих ТЗНОІ. У випадку достатньо великого значення параметра l, система має високий рівень інтелектуальності, що призводить до різкого виявлення аномалій і високої чутливості. Однак водночас є і недоліки, адже виникають нелінійні коливання змінної $x_1(t)$, хаотичні режими і складності стійкості інтелектуальної системи ухвалення рішень.

Розв'язок системи диференціальних рівнянь (1) – (3) має достатньо складне представлення, однак, якщо цю систему лінеаризувати, то можна дослідити стійкість цієї системи навколо її стаціонарної точки $\left(x_1^*, x_2^*, x_3^*\right)$. В інтерпретації математичного визначення стійкості системи виявлення ТЗНОІ введемо наступне означення.

Означення 1. Система диференціальних рівнянь (1) – (3), яка моделює процеси виявлення ТЗНОІ називається стійкою, якщо для довільного $\varepsilon > 0$ і для довільного $t > t - \tau$ виконується система нерівностей

$$\begin{cases} x_1(t) < x_1^* \\ x_2(t) < x_2^* \\ x_3(t) < x_3^* \end{cases}. \quad (5)$$

Після лінеаризації системи (1) – (3) вона матиме наступний вигляд:

$$\frac{dX(t)}{dt} = AX(t) + BX(t-\tau), \quad (6)$$

де $X(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix}$, $A = \begin{pmatrix} a & -c \cdot x_1^* & 0 \\ 0 & d & 0 \\ 0 & -n & k \end{pmatrix}$, $B = \begin{pmatrix} -b & 0 & 0 \\ 0 & 0 & g \\ m & 0 & 0 \end{pmatrix}$.

Для системи (5) характеристичне рівняння має наступний вигляд:

$$\det\left(\lambda I - A - B \cdot e^{-\lambda \cdot \tau}\right) = 0 , \quad (7)$$

де I – одинична матриця розміром 3х3.

Здійснивши відповідні перетворення, рівняння (6) являє собою наступний квазіполіном:

$$\lambda^3 + v_2\lambda^2 + v_1\lambda + v_0 + \left(b\lambda^2 - b(d+k)\lambda + dbk\right) \cdot e^{-\lambda\cdot\tau} + g \cdot \left(bn + mcx_1^*\right) \cdot e^{-2\lambda\tau} = 0 , \quad (8)$$

де $v_2 = -d - k - a$, $v_1 = d \cdot k + a \cdot d + a \cdot k + n \cdot g$, $v_0 = -a \cdot (d \cdot k) + n \cdot g$.

Рівняння (7) є трансцендентним і тому теорема Ляпунова, доведена для систем диференціальних рівнянь (5), у яких відсутній другий доданок правої частини рівняння (5), не є справедливою. Однак Український математик Рихлінський В.А. [1] зумів довести теорему щодо умов стійкості систем диференціальних рівнянь виду (5), яка отримала назву овал Рихлінського.

Суть теореми Рихлінського полягає у визначенні кола (околу) належності кожного спектра матриці

$$Q = A + B \cdot e^{-\lambda\cdot\tau} , \quad (9)$$

де кожному її власному числу відповідає область в комплексній площині за наступної умови

$$\begin{cases} |\lambda - a| \le -c \cdot x_1^* + d + k - n \\ |\lambda - d| \le a - c \cdot x_1^* - n + k \\ |\lambda - k| \le a - c \cdot x_1^* + d - n \end{cases}. \quad (10)$$

Система (9), яка визначає область, що носить назву овал Рихлінського, враховує взаємний вплив умов системи і дозволяє точніше оцінювати локальне розташування коренів рівняння (7) в комплексній площині. При цьому, область стійкості (10) повинна задовольняти умови, визначені наступною системою:

$$\begin{cases} |\lambda - a| \cdot |\lambda - d| \le d + k - c \cdot x_1^* - n \\ |\lambda - a| \cdot |\lambda - k| \le -c \cdot x_1^* + d + k - n \\ |\lambda - d| \cdot |\lambda - a| \le a - c \cdot x_1^* - n + k \\ |\lambda - d| \cdot |\lambda - k| \le a - c \cdot x_1^* + d - n \end{cases}. \quad (11)$$

Якщо позначити через $\mathrm{Re}(\lambda)$ дійсні частини власних чисел рівняння (6) і якщо для всіх λ, які належать овалам (10), виконується умова

$$\mathrm{Re}(\lambda) < 0, \quad (12)$$

то система є стійкою.

Продемонструємо вище викладене на конкретному прикладі. Нехай матриця A має наступний вигляд:

$$A = \begin{pmatrix} 1{,}4 & -0{,}35 & 0 \\ 0 & 1{,}1 & 0 \\ 0 & 0{,}3 & 1{,}6 \end{pmatrix}.$$

Після проведених розрахунків за представленням (7) і (10) було отримано наступні значення спектра матриці (9) з побудовою овалу Рихлінського, які представлені на рис. 2:

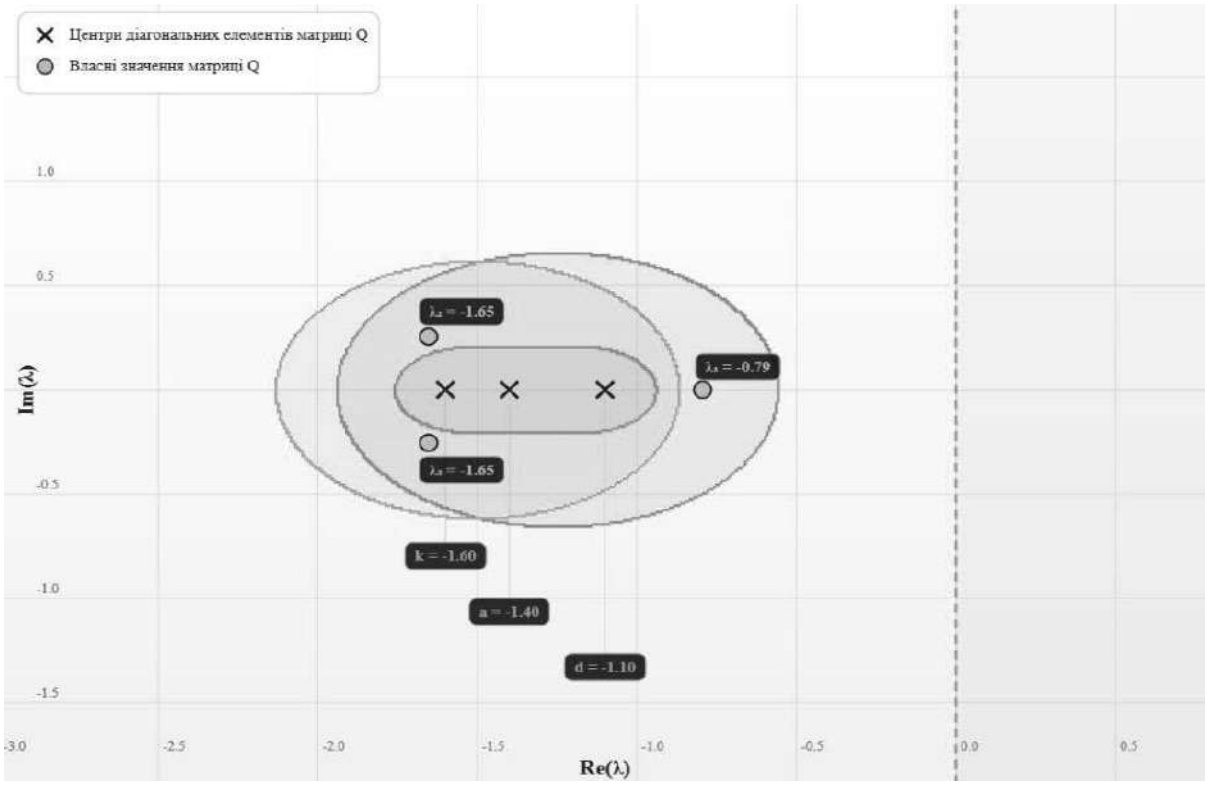

Рис. 2. Овали Рихлінського для лінеарізованої системи (6)

Овали Рихлінського показують, що стійкість диференціальних систем з запізненням визначається не тільки належністю дійсних частин розв'язків характеристичного рівняння (7), а і належністю їх певній області в цій комплексній півплощині. Практично це означає: інтелектуальний детектор здатний стабільно виявляти ТЗНОІ навіть за наявності помірних адаптивних перешкод.

Висновки

У даній роботі розроблено математичну модель процесів виявлення технічних засобів несанкціонованого отримання інформації в умовах навмисного впливу перешкод. Застосування систем диференціальних рівнянь із запізненням дозволило описати динаміку корисних ознак, адаптацію навмисних завад, описати структурну схему інтелектуального детектора виявлення ТЗНОІ ухвалення рішень. Уперше було застосовано геометричні представлення овалу Рихлінського, що уможливило забезпечення аналізу стійкості складної нелінійної системи. Отримані результати показують, що критичними факторами деградації виявлення ТЗНОІ є часові затримки, адаптивність перешкод, зростання нелінійності інтелектуального аналізу. Запропонований підхід може стати основою для створення інтелектуальних систем активної протидії ТЗНОІ нового покоління.

Література

1. Khusainov D.Ya., Ivanov A.F., Shuklin G,V.(2005). On a representation of solution of linear delay systems. Differential Equations. 2005. Volume 41. P. 1054-1058 https://link.springer.com/article/10.1007/s10625-005-0249-4
2. Шуклін Г.В., Наконечний В.С., Данилов І.Д., Пепа Ю.В. (2026) Моделювання процесів виявлення технічних засобів несанкціонованого отримання інформації в умовах навмисного впливу завад. Сучасний захист інформації. №1(65). 142–148 https://doi.org/10.31673/2409-7292.2026.011799
3. Крючкова Л., Ворохоб Н. (2025) Адаптивні методи протидії активним шумовим завадам. Кібербезпека: освіта, наука, техніка. Т. 2. № 30. 455–472. https://doi.org/10.28925/2663-4023.2025.30.987
4. Крючкова Л., Шандрук М. (2025) Методи протидії в радіонавігаційних конфліктах. Кібербезпека: освіта, наука, техніка. № 4(28). 766–780. https://doi.org/10.28925/2663-4023.2025.28.863
5. Васюта К., Збежховська У., Слободянюк В. (2022) Аналіз ефективності функціонування систем передачі інформації з хаотичними сигналами з OFDM-модуляцією в умовах впливу навмисних завад. Information Technology and Security. Vol. 10. Issue 2(19). 216–229. https://doi.org/10.20535/2411-1031.2022.10.2.270439
6. Заболотний В.І., Олейніков А.М., Заболотний Д.М., Кустов А.К. Технічний канал витоку інформації побічними електромагнітними перевипромінюваннями допоміжних технічних засобів і систем. Радіотехніка. 2024. № 218. DOI: 10.30837/rt.2024.3.218.04.
7. Євграфов Д.В., Яремчук Ю.Є. (2022). Алгоритми вимірювання частоти кадрової розгортки моніторів для частотно-вибіркового придушення каналів витоку інформації. Вісник Вінницького політехнічного інституту. 2022. № 4. С. 83–90. https://doi.org/10.31649/1997-9266-2022-163-4-83-90
8. Бибик Р., Опірський І. (2025). Дослідження методів аналізу та моделювання джерел РЕБ з урахуванням просторово-частотного орієнтування. Кібербезпека: освіта, наука, техніка. 2025. Том 2. №30. С.20 – 34 . https://doi.org/10.28925/2663-4023.2025.30.950.
9. Нужний С. (2025) Нейромережева модель оцінювання рівня захищеності складнозашумленої мовної інформації на основі структурної схеми RII. Кібербезпека: освіта, наука, техніка. 2025. №2(30). С. 645-661. https://doi.org/10.28925/2663-4023.2025.30.970
10. Puñal O., Aktas I., Schnelke C.-J., Abidin G., Wehrle K., Gross J. Machine Learning-Based Jamming Detection for IEEE 802.11: Design and Experimental Evaluation. IEEE WoWMoM, 2014. DOI: 10.1109/WoWMoM.2014.6918964.
11. Oligeri G., Sciancalepore S., Raponi S., Di Pietro R. PAST-AI: Physical-Layer Authentication of Satellite Transmitters via Deep Learning. IEEE Transactions on Information Forensics and Security. 2023. Vol. 18. P. 274–289. DOI: 10.1109/TIFS.2022.3219287.
12. Ibrahim K., Alnajim A.M., Naveed Malik A., Waseem A., Alyahya S., Islam M., Khan S. Entice to Trap: Enhanced Protection against a Rate-Aware Intelligent Jammer in Cognitive Radio Networks. Sustainability. 2022. Vol. 14, No. 5. Article 2957. DOI: 10.3390/su14052957.
13. Yara Cifuentes L.M., Cadena Muñoz E., Cubillos Sánchez R. Development of a Model for Detecting Spectrum Sensing Data Falsification Attack in Mobile Cognitive Radio Networks Integrating Artificial Intelligence Techniques. Algorithms. 2025. Vol. 18, No. 10. Article 596. DOI: 10.3390/a18100596.
14. Wang Q. et al. When Machine Learning Meets Spectrum Sharing Security: Methodologies and Challenges. 2022.
15. Pelechrinis K., Koufogiannakis C., Krishnamurthy S.V. On the Efficacy of Frequency Hopping in Coping with Jamming Attacks in 802.11 Networks. IEEE Transactions on Wireless Communications. 2010. Vol. 9, No. 10. P. 3258–3271. DOI: 10.1109/TWC.2010.082310.100113.
16. Wang S. etal. Improved Stability Criteria for Delayed Neural Networks via Lyapunov–Krasovskii Functional. Mathematics. 2022. Vol. 10, No. 15. Article 2768.
17. Shao H. A Lyapunov–Krasovskii Functional Plus Approach for Stability of Networked Control Systems with Transmission Delay and Packet Dropouts. 2023.

Дата першого надходження до видання: 04.05.2026
Внутрішня рецензія отримана: 22.05.2026
Зовнішня рецензія отримана: 27.05.2026
Дата прийняття статті до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

[1]Шуклін Герман Вікторович,
кандидат технічних наук, доцент
Shuklin German,
Ph.D. (technical sciences), associate professor
https://orcid.org/0000-0003-2507-384X
E-mail: mathacadem-kiev@ukr.net.

[1]Барабаш Олег Володимирович,
доктор технічних наук, професор
Barabash Oleg,
Ph.D. (doctor, technical sciences),
professor
https://orcid.org/0000-0003-1715-0761
E-mail: bar64@ukr.net.

[2]Гребенніков Асаді Болдхоягович,
заступник директора
Grebennikov Asadi,
deputy director
https://orcid.org/0000-0002-1207-7609
E-mail: g_as_b@ukr.net.

[3]Данілов Ігор Дмитрович,
аспірант
Danylov Igor,
post-graduate student
https://orcid.org/0009-0000-1426-6414
E-mail: danylovihor@gmail.com.

[3]Пепа Юрій Володимирович,
Доцент
Pepa Yuriy,
Ph.D., associate professor
https://orcid.org/0000-0003-2073-1364
E-mail: yurka14@ukr.net.

Місце роботи авторів:

[1]Національний технічний університет України «КПІ імені Ігоря Сікорського»
NTU of Ukraine “Igor Sikorski Kyiv Politechnical Institute”

[2]Інститут програмних систем НАН України
Institute of Software Systems of the NASU

[3]Державний університет інформаційно-комунікаційних технологій
State university of info-communication technology

УДК 004.43 https://doi.org/10.15407/pp2026.02.077

D. Shcherbak, K. Zhereb

IMPLEMENTATION OF HOT RELOADING IN COMPILED PROGRAMMING LANGUAGES

Hot reloading is a powerful software developing tool which allows the programmer to make changes to the codebase, and see those changes be applied to the program while it is running. This feature is naturally easy to implement in interpreted programming languages, such as Python, JavaScript and Ruby. This feature is highly demanded in compiled languages as well, because it allows to write and debug programs much faster and simpler, without the need to recompile the whole project to test out new functionality. It is especially useful when working on user interface or investigating problems that require the program in a specific state, which is hard or time-consuming to reproduce. This paper explores existing approaches to solving this problem in compiled languages. We take a look at approaches used in languages with additional runtime, such as Java and C#, and explore languages which are compiled to native binaries, such as C++, Rust and Zig. We also discuss current challenges that arise with this technology, and what solutions are possible in new generations of programming languages.
Keywords: hot reloading, hot swapping, dynamic software updates, dynamic library reload, runtime code patching, programming languages

Д.В. Щербак, К.А. Жереб

РЕАЛІЗАЦІЯ ГАРЯЧОЇ ЗАМІНИ КОДУ В КОМПІЛЬОВАНИХ МОВАХ ПРОГРАМУВАННЯ

Гаряча заміна коду – це потужний інструмент для розробки програмного забезпечення, який дозволяє програмісту вносити зміни коду і бачити їх вплив на програму під час того, як вона виконується. Цю функцію легко реалізувати в інтерпретованих мовах програмування, таких як Python, JavaScript та Ruby. Ця функція також дуже затребувана в компільованих мовах, оскільки вона дозволяє писати та відлагоджувати програми набагато швидше та простіше, без необхідності перекомпілювати весь проєкт для тестування змін. Це корисно під час роботи над інтерфейсом користувача або у процесі дослідження проблем, які вимагають фіксацію програми у певному стані, особливо, якщо відтворення такого стану займає багато часу. У цій статті досліджуються існуючі підходи до вирішення цієї проблеми в компільованих мовах. Ми розглядаємо підходи, що використовуються в мовах з додатковим середовищем виконання, таких як Java та C#, та досліджуємо мови, які компілюються безпосередньо в бінарні файли, такі як C++, Rust та Zig. Ми також обговорюємо поточні проблеми, що виникають з цією технологією, та рішення, можливі в нових поколіннях мов програмування.
Ключові слова: гаряча заміна коду, динамічні оновлення програмного забезпечення, динамічне перезавантаження бібліотек, виправлення коду під час виконання, мови програмування

Introduction

Hot reloading (or hot swapping) refers to the ability to modify the source code for existing programs and see the changes appear in the program without having to restart it [1]. This feature is native to interpreted and dynamic languages since they often run straight from the source code. Hence, changes to the source code will be instantly reflected in the program behavior. By contrast, compiled languages, which produce executables tailored to the target system, face some difficulties implementing this feature. This paper researches the history of the hot reloading feature, from its first implementations in dynamic languages to adoptions to current popular compiled systems languages like C++, Java and C#. We also discuss limitations that might be imposed on this feature in compiled languages, for instance immutable function definitions and class definitions, and explore how those limitations can be overcome in the future.

Hot reload in dynamic languages

Hot reloading has its roots in the earliest dynamic languages and environments. For example, Smalltalk, which was introduced in the 1970s, presented programmers with an environment which allowed them to make changes to the classes during the debugging session [2]. The simple approach of Smalltalk, which relies on objects, their methods and messages being sent between those objects, provides a uniform interface for all language features, and allows making changes in runtime easily.

Common Lisp also allows for reassigning symbols at runtime [3]. This functionality can be implemented thanks to the interpreter that runs the program and can incorporate the changes at runtime, pausing the execution if needed and providing safety checks to ensure validity of program state. Since code isn't converted to binaries, the runtime environment can manage updated definitions of methods and functions, calling the new version after any changes were made. Following Lisp and Smalltalk, Python also incorporates a similar approach: as an interpreted language, it allows for changes in functions and class definitions in a live developing environment, and it can also load changed versions of modules on the fly using importlib.reload [4]. This flexibility usually comes at the cost of efficiency and runtime execution speed, which are typically lower in interpreted programming languages. This paper addresses the challenge of bringing hot reloading capabilities to compiled languages without the need to sacrifice the efficiency of compiler optimizations.

Hot reload in compiled languages

By contrast, compiled languages achieve significantly better performance by producing machine-readable artefacts, which are tailored to the target operating platform. This approach allows to perform great performance optimizations – for example, unused variables might not be calculated at all [5].

The possibility of using hot reloading in compiled languages was intriguing. Java introduced HotSwap [6] in 2002, and C# added EditAndContinue [7] in 2004, introducing some limited hot swapping features, mainly in debugging mode, for the convenience of the developer. These languages run on virtual machines (JVM for Java) and managed runtimes (.NET CLR for C#), which allows for easier implementation of hot swapping, than in languages with fully native binaries. On the other hand, systems languages like C++ and Rust historically lacked built-in hot reloading, but various attempts of implementing it were made. In this section we discuss the most notable implementations of hot swapping in Java, C#, C/C++ and some other compiled languages, focusing on their capabilities and constraints.

Java and the JVM

Some hot reload functionality can be accessed in Java through the JVM HotSwap mechanism [6]. It was first introduced in Java 1.4, and gives the ability to change method bodies while in debug mode on a live program. This can be useful while fixing minor bugs, but it imposes major limitations: developers are not allowed to add or remove methods and class fields, change their type, name and access level, or mutate class hierarchy. The original design of HotSpotVM held an assumption of immutability of classes during runtime, hence all changes to class hierarchy still required full application restart. This limitation prompted further research, such as the Dynamic Code Evolution VM (DCEVM), which was a patch for HotSpotVM developed by Thomas Würthinger in the Institute for System Software in Linz [8]. It drastically increased the abilities of classic HotSwap, allowing for changing class fields, adding and removing methods, and changing class hierarchy. This is achieved using a special $transformer method, which is applied to every object on the heap of the JVM, which initializes new fields, and helps ensure the consistency of class invariants. This instrument is very powerful, allowing for many types of changes to the running program, with no ongoing runtime cost – benchmarks show [8] that the program

returns to original performance metrics after the changes are applied. The drawbacks of this approach might include the slightly longer reload times, additional complexity and the fact that this is JVM-specific: the same virtual machine that provides all the class information at runtime, which is not available in system languages like C++, introduces additional memory and performance overheads. However, with so many advantages for a considerably low price in terms of performance, the adoption of this instrument is low: the project repository wasn't updated for six years as of time of writing this paper. The fact that DCEVM is a separate tool could be a factor that limits the adoption. It needs a separate installation and users need to learn how to use it alongside usual Java tools. We believe that if such powerful hot reloading was a native language feature, with the main language features built around it, much more developers would be using it.

Other notable implementations include JDrums [9] and JVolve [10]. JDrums implements a dynamic JavaVM, which can allow for many runtime changes, but are limited in changes to class hierarchy. JVolve uses Jikes ResearchVM to implement dynamic code updates, but falls short of DCEVM as well, as it is not able to change running methods.

.NET and C#

In the ecosystem of .NET framework and the C# language, the ability to make changes while the program is running was represented since the early 2000s by the "Edit and Continue" feature [7]. This feature was available in the official .NET IDE, the Visual Studio, and allowed for minor changes to be done while the program was in Debug Mode. Those changes will then be applied to the program, without the need to restart the application. However, the number of changes that were available was quite limited, primarily changes inside method bodies were allowed, with no changes to class hierarchy or layout.

This feature was recently extended substantially by the new .Net Hot Reload feature, released first for .Net 6 in 2021 [11]. First major improvement was the ability to make changes without the need for explicit breakpoints, while the Edit-And-Continue feature required the program to be completely stopped before you could make any modifications to the running executable. The number of changes that are allowed was also significantly increased: the new .Net Hot Reload supports adding methods, fields, constructors, properties to classes, including async methods and changing method's return type [12]. However, some changes still remain unavailable, such as changes to the interfaces, adding a destructor to existing class, modifying a class parameter.

Akin to Java using its JVM to smooth out hot reload, .NET is using its Just-In-Time compiler to seamlessly introduce updated method bodies to the existing application. However, some systems are not suited to incorporate the additional overhead of Just-In-Time compilation and Garbage Collection, and instead rely on low-level languages, such as C and C++. In the next section we review the state of hot reloading in C++, as an example of a language that is compiled to a native executable, and how it handles hot reloading.

C and C++

Implementing hot reloading in languages that compile directly to machine code is a much more challenging task. It is further complicated by major compiler optimizations, which can drastically change the layout of instructions in the executable. There are two main approaches to this problem. The first approach is Dynamic Library Reload, described in [13]. This is achieved by splitting a program into a host subprogram, which starts the application and calls the main executing functions, and the rest of the logic, which is stored in Dynamic Link Libraries (DLL) or Shared Objects (SO). When the new version of a function is compiled, the host loads the new Shared Object instead of the old one and redirects function pointers, ensuring that all future calls will execute the new logic. The state of the app is stored by the host, so that changing the Shared Objects will not affect the application

data. This approach has some drawbacks: it requires ahead-of-time planning. Since changes cannot be applied in the host part of the application, signatures of the functions being called from Shared Objects cannot change, and need to be decided in advance. Changing object layouts in memory is also problematic with this approach, since state is most often managed by the immutable host, and if new SOs try to access objects with a different schema, they will encounter unexpected errors [14].

Another approach to hot reloading in C++ is Runtime Code Patching, described in [15]. This way, the changes are made directly to the executable code of the process in memory. This is a much harder task, since compilers are relying on the fact that the program code is static, and can use this information, as was mentioned, to perform optimizations. One of the tools that can be used to achieve this is Live++, a proprietary tool developed by Molecular Matters [16]. It requires the program to be compiled with special build flags that add a small patchable entry point at the beginning of each function. This way Live++ can easily insert a jump instruction which provides an indirection for the execution flow, allowing to delay execution before changes are made to the target function, and then return to normal execution, or if the size of the function is growing more than the allocated scope allows, the function instructions can be moved to a different slot in memory altogether. Another powerful feature is Hot Restart: if the changes made by the developer cannot be handled in runtime, they will be applied through the full program restart. However, since the app was already precompiled, the changes can be applied much quicker, skipping most of the warmup time. All in all, Live++ provides a great set of tools to perform hot reloading in C++, allowing for many types of changes during runtime with minimal overhead. A few of its drawbacks include some compile parameters limitations: the /FUNCTIONADMIN linking parameter is compulsory and some optimization flags, like Whole-Program Optimizations and Link-Time Optimizations are prohibited. It is prohibited to introduce new global or static variables into thread-local storages, and some behaviour changes might occur when debugging a patched program.

From the academic side, the most notable solutions for dynamic software updating in C and C++ included Ginseng, which implemented hot reloading for C in [17], and later improved it with an algorithm for a multi-threaded solution in [18]. However, its approach required automatically introducing additional variables to each struct and function to provide indirection, which could cause performance issues. A later project Kitsune [13] proved to outperform it in terms of runtime update speed, while maintaining the ability to make almost arbitrary changes to the program. This, however, was achieved by a requirement for the developer to explicitly mark code points, in which hot reload is possible. While such technique can work as a safety guarantee (or at least shift the blame for the inconsistencies on the developer, who didn't properly choose the update points), it makes the adoption process of Kitsune much harder for many projects, especially for those with an extensive code base.

Other notable implementations

Zig, as a low-level language that is compiled to a native executable, can easily implement the dynamic library reload, introduced in the C++ section of [19]. An implementation of runtime code patching was proposed in 2022 by one of the language maintainers [20]. The proposal outlined an incremental compilation mode, which left the compiler running as a background service. After receiving a command to update the live executable, the compiler determines which changes need to be made and updates the corresponding memory of the running process. This is only implemented as an experimental feature, and no handling of program state was described, but it shows potential for future implementations of hot reloading in Zig, leveraging its low level access to memory.

Erlang is a programming language built around the idea of scalable, fault-tolerant systems with long running time [21]. The main idea of Erlang is implementing multiple

concurrent processes, that don't share any memory or state, and communicate via asynchronous messages. Those processes are designed in a way that makes them easy to halt and start new instances, which provides a perfect environment for hot reloading. Erlang supports multiple versions of the same process running at the same time, with the main assumption that all old processes will eventually die out, and only the newest version processes are created as soon as it is compiled. Though Erlang is using a virtual machine, which restricts it from being used in performance-critical environments, its ability to change running systems, among other advantages, made it widely used in telecommunication systems, bankings and more.

Rust is another example of a low-level language, that is known for its long compile times, and, as such, would benefit greatly from hot-reloading capabilities. There are solutions for Dynamic Library Reload like dynamic_reload [22] or hot_lib_reloader [23], but no implementations of Runtime Code Patching were introduced. With **Go**, another popular compiled language, the most successful solutions like Air [24] provide the capability of Live Reloading, which essentially fully restarts the application in an automated way.

Mun [25] is a programming language in the early stages of development, which aspires to have a Rust-like syntax and feature set, combined with native capabilities of hot reloading. As of writing of this paper, it is too early to tell how it compares in terms of speed, and patches runtime overhead to other solutions in this paper. However, it shows promise, and has potential to become a great example of native hot-reload implementation in a compiled language.

Proposed Approaches for High-Performance Hot Reload

Since hot-reloading is a powerful and useful tool, we believe it is very likely to be a part of new languages that will be developed in the future. Yet, it is complex and hard to implement, especially post-factum in a language that is already in use, because current memory layouts in existing programs cannot be changed.

One way to overcome this is to start building the compiler around the idea of hot-reloading, extensibility and improvability. Most popular compiled languages don't support hot-reloading out of the box. We believe that possible growth of hot reload implementations and usage is limited, partly, by the general approach, which is common for most compiled languages – treating the executable as an immutable artefact.

Of course, such a way of thinking has its benefits. It provides for encapsulation, modularity and more strict separation of responsibilities. And in the early days of computing it was enough: simple programs that automated mathematical tasks, such as simple physics simulations, followed the same path. The program started at a clean slate, allocated and initialized some resources, executed some computations, freed the resources and died. Today, many programmers work on long-running servers that might have uptime of days, weeks and months. Yet, for compiled systems languages, these servers still need to be put to a complete stop for every change that needs to be introduced. Entire systems, like Kubernetes, were implemented for ensuring the uptime of big systems, and one of the features they provide is an ability to update a running service to a newer version without disrupting the users' workflow. This is achieved by starting multiple instances of the same service, and gradually switching some of them to a newer version, while redirecting users to them. While this solution is elegant, it does introduce a certain level of complexity, which is only required because the executables that are running the servers are immutable. Perhaps, it would be beneficial to write programs in an environment that allows them to change while running, providing evolution of systems, not revolutionary (breaking) changes.

Performance-critical systems, which cannot allow for large overhead of an orchestrator, such as Kubernetes, would still benefit from an ability to upgrade its software. In 1988, part of

a space probe software, which was written in Lisp, malfunctioned, and the mission was saved, because Lisp allowed to debug and change its code while live and in the process of a spaceflight [26]. While this shows how important it is to be able to debug and improve the program in runtime, a strict type system and compile-time checks can prevent many cases of undesired behavior. Combining both the safety of a compiler with the flexibility of hot reloading is a way to achieve the most stable and adaptable systems.

Compiler built around Hot Reloading feature

As was mentioned earlier, implementing the Hot Reload feature in an existing language is quite difficult, and it almost always results in some overhead. For example, in compiled OOP-languages, like Java and C++, one of the issues that could arise is related to the object's layout in memory. If a new field is added to an object via Hot Reloading, how to add this field to existing objects?

Such a problem would be solved, if the compiler is designed with the idea of hot reloading in mind. Then all the tooling, memory layouts and runtime can be implemented in such a way that allows to change them easily, even while the program is running. This also allows the implementation of different approaches to hot-reloading, exploring different trade-offs.

One approach is to do memory mapping in a way that makes the addresses of functions in memory predictable and easy to calculate. This allows to make the hot swapping process as fast and painless as possible, but might result in the final executable being larger, which sometimes could be an issue, if the target machine doesn't have a lot of spare memory. Alternatively, it is possible to provide the most efficient memory layout for the executable, and calculate which parts need to be changed at each hot swap event, factoring in additional factors like compiler optimizations. This will result in smaller executable sizes and, in some cases, faster code, for the cost of longer hot-reload times. Building the compiler with the concept of hot reload in mind allows to implement both approaches much easier. And of course, such a compiler can still be used in an old-fashioned way: just a simple executable, no hot-reloading capabilities, for the sake of security, speed and reliability.

This way of thinking, in terms of evolving, runtime-developed systems, allows for potential new approaches to programming in general. As an example, it might be possible to write some interface with default implementations, and combine it with logic that will invoke new implementations, when they are implemented, and until then will fall back to the default implementation. In other words, it becomes possible to not only reason about what the code *is*, but also about what it *will be*.

Proposed compiler architecture

In this section we present a concept of a future approach to compiler architecture, for a statically compiled language to achieve maximum native hot reloading capabilities. We propose to combine the Dynamic Library Reload (DLR) approach with Runtime Code Patching (RCP). In principle, DLR allows for the easiest approach, and the one providing the least overhead. However, it has its limitations in terms of program state and function signatures, which were already discussed earlier. RCP, in turn, allows for more flexibility, but can cause more additional overhead. What is meant here by combination of those approaches, is that the compiler separates the code, designating some of it to dynamic libraries for ease of hot swapping in runtime. At the same time, the controlling unit, which in regular DLR approach is left immutable, is compiled in a way that allows the use of the RCP approach, allowing for changes here as well, with the possibility of state transformations. This way it is possible to allow for maximum flexibility, while keeping memory overhead to a minimum. What is also important, compiler optimizations also take this layout into account, so the developer won't be forced to choose between powerful optimizations and hot reloading. The proposed design is presented at Figure 1. This approach

doesn't require a separate process to handle hot reloading, as it is done by the compiler itself, which can be executed as a background process. When the hot reload is triggered, the compiler checks for changes since the last compilation, compiles them, without recompiling the whole project, but using previous compilation info to provide type checks and other validation. Upon successful compilation, a patch with new shared object files and changes necessary to the main unit are applied to the running executable.

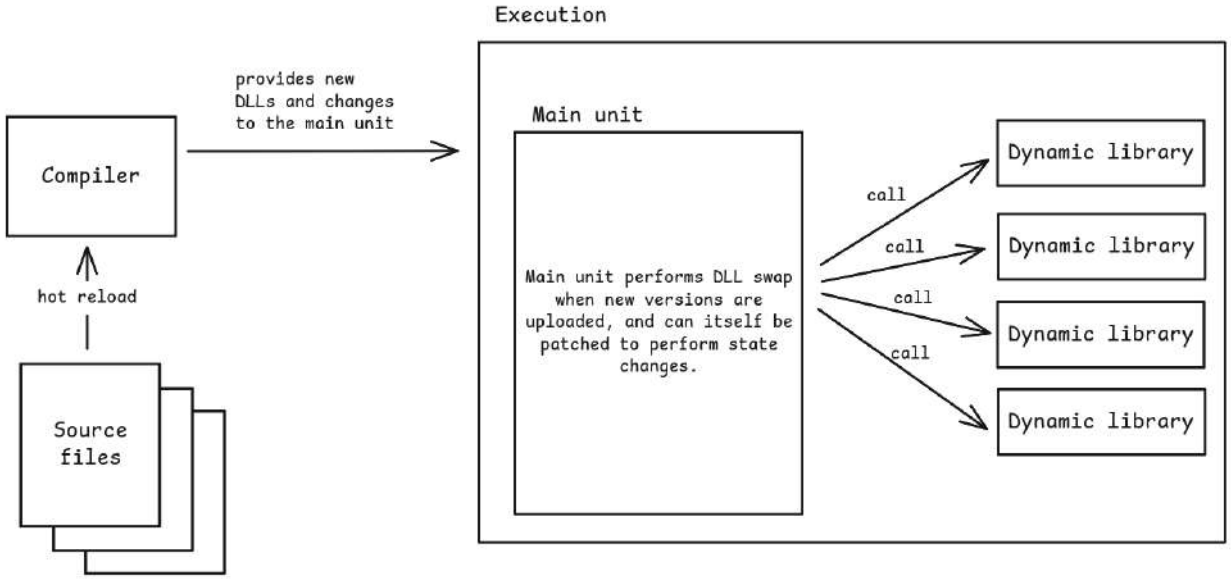

Fig. 1. Proposed compiler architecture

Prototype implementation

To test how two approaches can be used together, we implemented a simple proof of concept project. The source code is available on GitHub at [27], and general representation of the project workflow is presented at Figure 2. The main loop iterates through calls to simple mathematical functions, one of which is loaded from a dynamic library via DLR module, and accessed through a function pointer, and another one is compiled in a static module and directly called from the main loop. Both functions can be modified, and after a call to the compilation system the changes will be injected into the main loop, either by swapping the shared object for a newer version, or by providing a byte sequence to the RCP module, which is directly injected into the running program memory. A compilation system in this context is a standard C++ compiler, combined with a number of simple bash scripts, needed to perform file substitution and pass build parameters to the compiler. This project also shows that DLR is capable of loading completely new functions, while RCP cannot do that. RCP on the other hand allows much more flexibility, but requires more preparation and is harder to implement. Benchmarking of this project showed that patching time varies for the DLR approach, with bigger batches taking more time to load. With RCP, bytes can be loaded independently of the main program running, so the time for which execution is interrupted is limited to inserting the jump instruction to the code, and does not grow as patch gets bigger, as can be observed in Table 1.

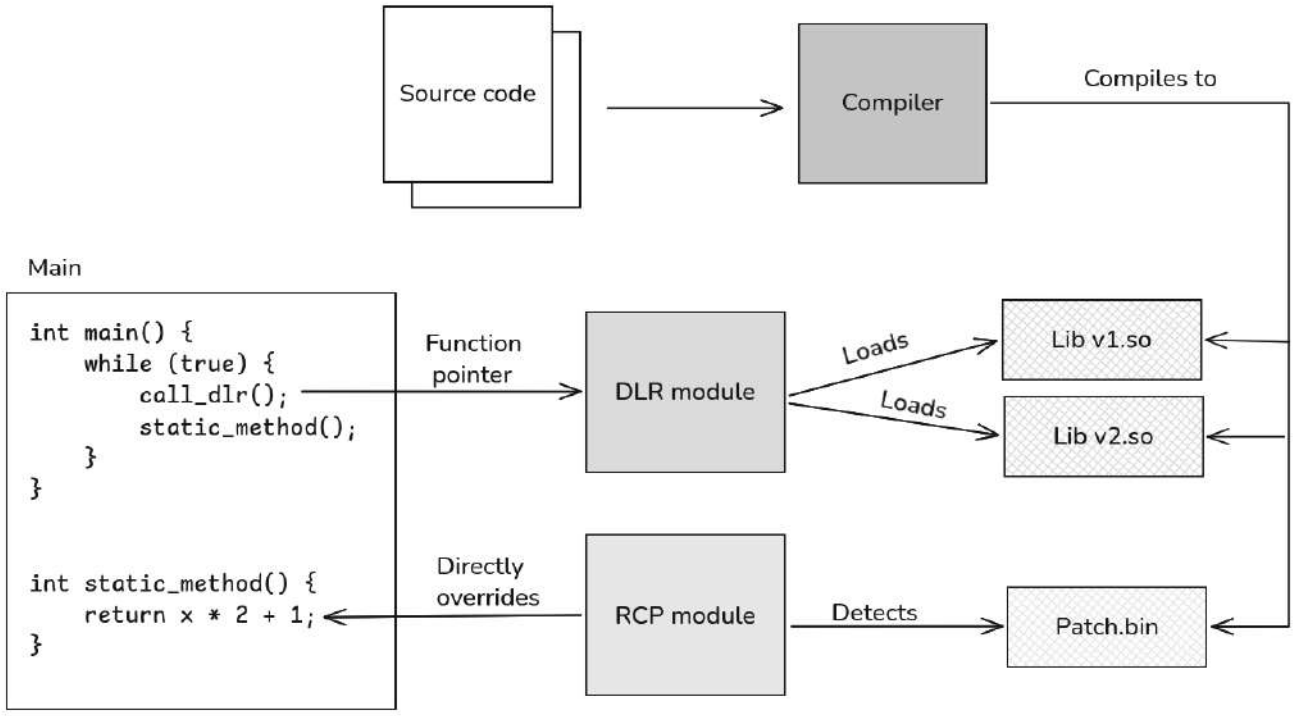

Fig. 2. Implementation of combined DLR and RCP method

Table 1.

Benchmarking results for different hot reload approaches

Patching mechanism	Mean patch time (ns)	Min patch time (ns)
RCP, ~40 bytes	2073	1607
RCP, ~300 bytes	1993	1594
DLR, 1 symbol	278	267
DLR, 1000 symbols	29565	27985

Conclusion

In this paper we explored the history of the hot reloading feature in different programming languages, starting with the first examples, which came from interpreted languages. We then discussed approaches to hot reloading in the most popular compiled languages, ranging from academic results, such as DCEVM for Java, to proprietary solutions like Live++ for C++. We also considered implementations and proposals thereof in less popular languages. In later sections, we discussed the importance of the hot reload feature for modern and future programming languages, and proposed an approach of incorporating this feature early in the development of future programming languages, to ease its implementation and usage. Lastly, we introduced a proposal for compiler architecture, which allows for non-restrictive and fast hot reloading by combining the Dynamic Library Reload with Runtime Code Patching. The proof of concept implementation, provided in the last section, confirmed the benefits of using both approaches, and provided a foundation for a future full-scale compiler, built around the idea of hot reloading.

References

1. Appavoo, J., et al. (2003) ‘Enabling autonomic behavior in systems software with hot swapping’, in *IBM Systems Journal*, vol. 42, no. 1. Armonk, NY, USA: International Business Machines Corporation, pp. 60–76.

2. Kay, A.C. (1996) ‘The early history of Smalltalk’, in History of Programming Languages---II. New York, NY, USA: Association for Computing Machinery, pp. 511–598.
3. Steele Jr, G. L. (1990). *Common Lisp the Language* (2nd ed.). Digital Press.
4. Python Software Foundation (2025), *importlib — The implementation of import.* Available at: https://docs.python.org/3/library/importlib.html (Accessed: 15 October 2025)
5. Godbolt, M. (2020) ‘Optimizations in C++ compilers’, *Commun. ACM*, 63(2), pp. 41–49.
6. Oracle Corporation (2025) *HotSwap*. Available at: https://wiki.openjdk.org/spaces/mlvm/pages/7897093/HotSwap (Accessed: 15 November 2025)
7. Microsoft (2025) *Edit and continue for C#.* Available at: https://learn.microsoft.com/en-us/visualstudio/debugger/edit-and-continue-visual-cshar (Accessed: 30 October 2025)
8. Würthinger, T., Wimmer, C. and Stadler, L. (2010) ‘Dynamic code evolution for Java’, in *Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java*. New York, NY, USA: Association for Computing Machinery (PPPJ ’10), pp. 10–19.
9. Andersson, J. and Ritzau, T. (05 2000) ‘Dynamic Code Update in JDRUMS’.
10. Subramanian, S., Hicks, M. and McKinley, K. (06 2009) ‘Dynamic Software Updates: A VM-centric Approach’, in, pp. 1–12.
11. Lyalin, D. (2021) *Introducing the .NET Hot Reload experience for editing code at runtime*. Available at: https://devblogs.microsoft.com/dotnet/introducing-net-hot-reload (Accessed: 01 November 2025)
12. Microsoft (2025) *Write and debug running code with Hot Reload in Visual Studio.* Available at: https://learn.microsoft.com/en-us/visualstudio/debugger/hot-reload (Accessed: 01 November 2025)
13. Hayden, C.M. et al. (2012) ‘Kitsune: efficient, general-purpose dynamic software updating for C’, in Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications. New York, NY, USA: Association for Computing Machinery (OOPSLA ’12), pp. 249–264.
14. Zylinski K. (2023) Hot Reload Gameplay Code: What, why, limitations and examples. Available at: https://zylinski.se/posts/hot-reload-gameplay-code (Accessed: 1 November 2025)
15. Buck, B. and Hollingsworth, J.K. (2000) ‘An API for Runtime Code Patching’, in *The International Journal of High Performance Computing Applications*, vol. 14, no. 4. Thousand Oaks, CA, USA: SAGE Publications, pp. 317–329.
16. Molecular Matters GmbH (2025) *Live++ for Windows - Documentation.* Available at: https://liveplusplus.tech/docs/documentation.html (Accessed: 30 October 2025)
17. Neamtiu, I. et al. (2006) ‘Practical dynamic software updating for C’, in Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation. New York, NY, USA: Association for Computing Machinery (PLDI ’06), pp. 72–83.
18. Neamtiu, I. and Hicks, M. (2009) ‘Safe and timely updates to multi-threaded programs’, in Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation. New York, NY, USA: Association for Computing Machinery (PLDI ’09), pp. 13–24.
19. Perkin, L. (2024) *Hot-reloading with Raylib.* Available at: https://zig.news/perky/hot-reloading-with-raylib-4bf9 (Accessed: 26 October 2025)
20. Konka, J. (2022) *Hot-code reloading on macOS/arm64 with Zig.* Available at: https://www.jakubkonka.com/2022/03/16/hcs-zig.html (Accessed: 8 November 2025)
21. Armstrong, J. (2007) ‘A history of Erlang’, in Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages. New York, NY, USA: Association for Computing Machinery (HOPL III), pp. 6-1-6–26.
22. The Rust Foundation (2025) Dynamic_reload crate documentation. Available at: https://crates.io/crates/dynamic_reload (Accessed: 26 October 2025)
23. The Rust Foundation (2025) hot_lib_reloader crate documentation. Available at: https://crates.io/crates/hot-lib-reloader (Accessed: 26 October 2025)
24. Nguyen, H. (2024) *Guide for Implementing Live Reload Using Golang Air.* Available at: https://reliasoftware.com/blog/live-reload-using-golang-air (Accessed: 14 November 2025)
25. Mun language documentation. Available at: https://docs.mun-lang.org/ (Accessed: 15 November 2025)

26. Cassel, D. (2022), *NASA Programmer Remembers Debugging Lisp in Deep Space.* Available at: https://thenewstack.io/nasa-programmer-remembers-debugging-lisp-in-deep-space (Accessed: 27 October 2025)
27. Implementation of the prototype described in the paper. Available at: https://github.com/DShcherbak/hot-reload-poc (Accessed: 23 March 2026)

Дата першого надходження до видання: 26.03.2026
Внутрішня рецензія отримана: 14.04.2026
Зовнішня рецензія отримана: 20.04.2026
Дата прийняття статті до друку:05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Денис Щербак,
аспірант
Denys Shcherbak
post-graduate student
https://orcid.org/0009-0006-7842-565X

Костянтин Жереб,
кандидат фізико-математичних наук,
асистент кафедри
Kostyantyn Zhereb
Ph.D., (physical and mathematical sciences),
assistant professor
https://orcid.org/0000-0003-0881-2284

Місце роботи авторів:

Київський національний університет імені Тараса Шевченка,
факультет комп'ютерних наук та кібернетики
Kyiv Taras Shevchenko National University,
Faculty of computer science and cybernetics
тел. +38(044) 521-32-74
E-mail: csc@knu.ua
https://csc.knu.ua

УДК 004.942:37.018.43 https://doi.org/10.15407/pp2026.02.087

М.Ю. Полторацький, А.С. Волянюк

ФОРМАЛЬНА МОДЕЛЬ ВЕРИФІКАЦІЇ ПЕРСОНАЛІЗОВАНИХ ОСВІТНІХ ТРАЄКТОРІЙ НА ОСНОВІ ALLOY

У статті розглянуто проблему верифікації персоналізованих освітніх траєкторій у сучасній освітній інженерії в умовах цифровізації освітнього процесу. Обґрунтовано необхідність перевірки коректності навчальних маршрутів, сформованих з урахуванням індивідуальних освітніх потреб, попереднього досвіду, рівня підготовки та професійних цілей здобувачів освіти. Акцентовано увагу на тому, що ефективність персоналізованого навчання визначається не лише якістю проєктування індивідуальної траєкторії, а й можливістю її формальної перевірки на узгодженість, повноту та досяжність результатів. Запропоновано формальну модель, реалізовану засобами мови специфікації Alloy, яка дозволяє перевіряти відповідність набутих навичок вимогам кар'єрної мети. У межах моделі визначено ключові сутності, зокрема здобувача освіти, навчальні дисципліни, набуті та необхідні навички, а також професійні орієнтації, і формалізовано зв’язки між ними. Запроваджено систему інваріантів і обмежень, що забезпечують перевірку коректності освітніх траєкторій, включаючи наявність визначеної кар’єрної мети, змістовність навчальних компонентів і відповідність результатів навчання вимогам професійної діяльності. Модель забезпечує динамічну перевірку досяжності освітніх результатів шляхом аналізу змін у наборі набутих навичок на різних етапах навчання. Це дає змогу виявляти потенційні помилки у формуванні траєкторій, оцінювати ефективність алгоритмів рекомендації навчальних дисциплін і підтверджувати досяжність визначених цілей. Отримані результати засвідчують, що застосування формального підходу сприяє підвищенню валідності персоналізованого навчання, обґрунтованості освітніх рішень та ефективності адаптивних освітніх систем. Перспективи подальших досліджень пов’язані з розширенням системи обмежень, урахуванням невизначеності даних і вдосконаленням механізмів адаптації освітніх траєкторій.

Ключові слова: персоналізоване навчання, освітня траєкторія, верифікація, освітня інженерія, адаптивність, навички, кар'єрна мета, алгоритм, модель, навчальний курс

M. Poltoratskyi, A. Volianiuk

FORMAL MODEL FOR VERIFICATION OF PERSONALIZED EDUCATIONAL TRAJECTORIES BASED ON ALLOY

The article addresses the problem of verification of personalized educational trajectories in modern educational engineering under the conditions of digitalization of the educational process. The necessity of verifying the correctness of learning pathways formed with regard to individual educational needs, prior experience, level of training, and professional goals of learners is substantiated. It is emphasized that the effectiveness of personalized learning is determined not only by the quality of individual trajectory design but also by the possibility of its formal verification in terms of consistency, completeness, and attainability of results.

A formal model implemented using the Alloy specification language is proposed, which enables verification of the correspondence between acquired skills and the requirements of a career goal. Within the model, key entities are defined, including the learner, educational subjects, acquired and required skills, as well as professional aspirations, and the relationships between them are formalized. A system of invariants and constraints is introduced to ensure the correctness of educational trajectories, including the presence of a defined career goal, the meaningfulness of educational components, and the alignment of learning outcomes with professional requirements. The model provides dynamic verification of the attainability of educational results through the analysis of changes in the set of acquired skills at different stages of learning. This makes it possible to identify potential errors in trajectory construction, evaluate the effectiveness of course recommendation algorithms, and confirm the attainability of defined goals. The obtained results demonstrate that the application of a formal approach contributes to increasing the validity of personalized learning, the soundness of educational decisions, and the efficiency of adaptive educational systems. Prospects for further research are related to extending the system of constraints, taking into account data uncertainty, and improving mechanisms for adapting educational trajectories.

Key words: personalized learning, educational trajectory, verification, educational engineering, adaptability, skills, career goal, algorithm, model, course

Вступ

Постановка проблеми. Персоналізоване навчання стало однією з провідних тенденцій сучасної освіти, набуваючи особливої актуальності в контексті цифровізації освітнього процесу. Його сутність полягає в адаптації навчання до індивідуальних потреб, інтересів, здібностей та цілей кожного учня. Такий підхід дозволяє відійти від стандартизованих моделей і створити умови для реалізації потенціалу кожного здобувача освіти. Розвиток інформаційно-комунікаційних технологій значною мірою сприяв поширенню персоналізації, зробивши її досяжною навіть у масштабних онлайн- або змішаних формах навчання.

Однією з ключових передумов ефективної персоналізації є створення повного профілю здобувача освіти, який враховує його індивідуальні особливості, зокрема попередній досвід, рівень знань, ставлення до навчання та інтереси [1]. Такий профіль слугує основою для формування унікальної освітньої траєкторії, яка постійно оновлюється у відповідь на зміни в навчальних досягненнях і мотивації здобувача. У цьому контексті важливою складовою персоналізованого підходу є адаптивність, що реалізується через динамічне коригування змісту, темпу та форм подачі матеріалу [2, с. 872-875]. Саме адаптивні навчальні траєкторії дозволяють забезпечити релевантність навчального досвіду на кожному етапі навчання, водночас підтримуючи інтерес та розвиток самостійності здобувача освіти.

Важливою умовою реалізації персоналізації є створення гнучких освітніх середовищ, які забезпечують здобувачам можливість самостійно обирати темп, формат і зміст навчання відповідно до власних потреб і переваг. Це не лише сприяє підвищенню мотивації та залученості, а й формує навички саморегуляції та відповідальності за результати власного навчання [3, с.501].

Однак, попри очевидні переваги, персоналізоване навчання стикається з рядом проблем, серед яких особливо актуальною є проблема перевірки коректності побудови індивідуальних освітніх траєкторій. Оскільки траєкторія формується автоматизовано на основі аналізу великої кількості даних, існує ризик виникнення помилок у її конструюванні або у надмірному покладанні на алгоритми, що не завжди здатні адекватно врахувати складну мотиваційно-особистісну структуру здобувача. Тоді постає завдання забезпечити не лише технологічну точність, а й валідність запропонованих шляхів навчання. Це потребує розробки інструментів для верифікації освітніх траєкторій з урахуванням педагогічної доцільності, гнучкості системи адаптації та ефективного зворотного зв'язку.

Аналіз останніх досліджень і публікацій. Персоналізована освітня траєкторія на сьогодні є одним із ключових концептів сучасної освіти, що передбачає створення унікального шляху навчання для кожного учня, побудованого відповідно до його здібностей, потреб, інтересів та цілей. У центрі цього підходу - особистість здобувача освіти, його активна участь у власному навчанні, а також створення умов для гнучкості та самостійності в здобутті знань.

Науковиця Алєксєєва С. підкреслює, що «індивідуальна освітня траєкторія має стати дидактичною системою, в якій навчання здійснюється за індивідуальними програмами, змістом, формами, засобами, темпом та відповідними формами контролю і оцінювання» [4, с. 5]. За її словами, головною метою такого підходу є максимальний розвиток потенціалу кожного здобувача освіти через формування самостійності, ініціативності, дослідницького стилю, творчості, впевненості та відповідального ставлення до праці. Особливої ваги індивідуальний підхід набуває в умовах нестабільності, як-от воєнний стан чи період післявоєнного відновлення, адже «цінність дидактичного алгоритму розроблення індивідуальної освітньої траєкторії... у тому, що він підвищує мотивацію, розвиває вміння вчитися та допомагає учням досягати кращих результатів навчання» [4, с.6].

У контексті освітньої реформи в Україні індивідуалізація навчання розглядається як необхідний крок до оновлення системи освіти. Науковці Лавренова М., Лалак Н., Молнар Т. та Фенчак Л. зазначають, що «одним із найважливіших аспектів реформування системи освіти в Україні є цілеспрямована та системна робота педагогів

над створенням індивідуальної освітньої траєкторії розвитку здобувача освіти як персонального шляху реалізації його особистісного потенціалу» [5, с.131]. Для втілення цього підходу необхідно переосмислити не лише зміст навчання, а й способи його організації, форми контролю та критерії оцінювання. Групою науковців під керівництвом Співаковського О. доведено, що використання ІКТ у збалансованій системі (фундаментальна, інваріантна та варіантна складові здобуття освіти) впливає на конкурентоспроможність випускників, що приводить до капіталізації студентів як майбутніх спеціалістів, здатних інтегрувати свої знання в умовах мінливого соціального та кооперативного середовища [6].

Реалізація персоналізованої освітньої траєкторії вимагає створення умов, за яких здобувачі освіти отримають можливість самостійно визначати зміст навчання, обирати індивідуальні цілі, підбирати методи та темпи засвоєння матеріалу, а також усвідомлювати та оцінювати власний поступ. Як зазначають дослідники, «учень зможе просуватись індивідуальною траєкторією в тому випадку, якщо йому надаватимуть такі можливості: визначати індивідуальний зміст вивчення навчальних предметів; ставити власні цілі у вивченні конкретної теми або розділу; вибирати оптимальні форми та темпи навчання; застосовувати ті способи навчання, що найбільш відповідають його індивідуальним особливостям; рефлексивно усвідомлювати отримані результати, оцінювати і коригувати свою діяльність» [5, с.137].

Таким чином, персоналізована освітня траєкторія не лише сприяє розвитку особистості здобувача, а й трансформує саму логіку освітнього процесу до особистісно орієнтованого, гнучкого і динамічного підходу, здатного відповідати викликам сучасності.

На основі аналізу наукових джерел ми притримуємось думки, що персоналізоване навчання як інноваційна педагогічна парадигма вимагає не лише створення, а й постійної перевірки коректності індивідуальних освітніх траєкторій, що формуються для кожного здобувача освіти.

Сучасні дослідження пропонують низку технологічних рішень цієї проблеми. Наприклад, у статті З. Папамітцоу [7] розглядається два підходи до формування та верифікації індивідуальних освітніх маршрутів у цифрових середовищах. Перший з них передбачає використання генетичних алгоритмів, які дозволяють знайти оптимальні шляхи навчання, враховуючи індивідуальні освітні потреби. Цей підхід виявився ефективним у генерації динамічних і релевантних траєкторій, особливо в умовах онлайн-навчання, де гнучкість та адаптивність є критично важливими. Другий підхід базується на застосуванні нейронних мереж, що формують рекомендації щодо подальших кроків у навчанні, адаптуючи траєкторії на основі аналізу освітньої поведінки учня. Обидва підходи підтримують автоматизоване ухвалення рішень, однак потребують постійного оновлення даних і уточнення моделей, аби гарантувати їхню педагогічну валідність [7].

Інший важливий напрям досліджень представлений у роботі Ковалюк Т., Кобець Н. та Дворник В. [8, с.115-128], де основний акцент зроблено на аналізі мотиваційних факторів як передумови формування індивідуального освітнього шляху. Авторки розробили інформаційну технологію, що дозволяє формувати траєкторії на основі латентно-семантичного аналізу мотиваційних листів здобувачів освіти, результатів опитувань щодо професійних інтересів та оцінки поточного рівня знань. Отримані дані використовуються для побудови онтологій предметних галузей у вигляді тезаурусів, які виступають концептуальними моделями змісту навчання. У результаті створюється індивідуальний навчальний план, що поєднує особистісну мотивацію, професійну спрямованість і рівень підготовки здобувача вищої освіти.

Загалом розроблені алгоритмічні підходи та аналітичні інструменти значно розширюють можливості перевірки й адаптації освітніх траєкторій, однак їхня ефективність залежить від якості вхідних даних та здатності моделей адекватно відображати складну структуру мотиваційно-ціннісної сфери учня. Тому подальший розвиток персоналізованого навчання потребує

інтеграції технологічної точності з педагогічною доцільністю, а також створення механізмів зворотного зв'язку, що дозволять коригувати траєкторії відповідно до змін в особистісному розвитку учня. Успішна реалізація цих завдань може забезпечити високу ефективність персоналізованого підходу та його стійку інтеграцію в освітні системи майбутнього.

Мета дослідження. Метою статті є обґрунтування можливості ефективного застосування формальної мови моделювання Alloy для перевірки досяжності під час формування персоналізованих освітніх траєкторій.

Методологія дослідження. У попередніх публікаціях було запропоновано модель побудови персоналізованих освітніх траєкторій із використанням технологій Semantic Web [9], розроблено RDF-модель [10] персоналізованого освітнього середовища засобами мови Notation3 [11], а також систему імплікаційних правил за допомогою мови N3 Logic Rules [12].

У статті Полторацького М. та Коннової О. [13] представлено опис методів валідації освітніх моделей на основі RDF і мови опису обмежень SHACL [14] та мови запитів SPARQL [15] для оцінки їхньої відповідності сучасним вимогам ринку праці.

На основі попередніх (зазначених вище) досліджень було вдосконалено модель сучасного персоналізованого освітнього середовища, а також розроблено алгоритми рекомендації курсів відповідно до кар'єрних уподобань здобувача вищої освіти.

У цій статті основну увагу зосереджено на розробці формальної моделі та перевірці коректності запропонованого підходу до побудови персоналізованих освітніх траєкторій із використанням мови специфікації Alloy [16].

Результати дослідження. Опис моделі.

У цьому розділі статті увагу зосереджено на розробці формальної моделі та перевірці коректності запропонованого підходу в статті [16] за допомогою мови специфікації Alloy.

Сутності в Alloy розглядаються як сигнатури. Отож Student - це сигнатура, яка має певний набір відношень, що можуть зв'язувати кожну сутність з іншими. Давайте розглянемо сигнатуру Student детальніше:

```
abstract sig Student {
  acquiredSkills: set Skill,
  aspiresTo: set careerAspir,
  enrollTo: set Subject
}
```

Сутність Student - це абстрактна сигнатура, яка має три відношення, що зв'язуються з множиною інших сутностей, а саме:

- **acquiredSkills** — відношення, яке пов'язує здобувача освіти з множиною навичок, які він уже здобув. Це ті навички, які необхідні для запису на іншу дисципліну (освітню компоненту). Прикладом можуть бути "Основи програмної інженерії" та "Формальні методи програмного забезпечення";
- **aspiresTo** — відношення, яке вказує на множину професійних цілей здобувача вищої освіти;
- **enrollTo** — відношення, яке вказує на множину навчальних дисциплін (курсів), на які зареєстрований здобувач вищої освіти.

Створимо конкретний інстанс сигнатури Student, що успадковує всі задані вище відношення, а також через оператора fact визначимо початковий набір базових навичок, інваріантів, які завжди мають бути істинними в усіх допустимих випадках моделі. Нижче наведено фрагмент такого формалізму:

```
one sig Alice extends Student {},

fact CareerAssignments {
        Alice.acquiredSkills = A2 +
Microsoft_Access
        Alice.aspiresTo =
Business_Analyst
    }
```

Як уже згадувалося, сутність Student через відношення enrollTo пов'язана з сутністю Subject. Створення такого типу відношення дозволить нам проаналізувати досяжність побудованих траєкторій згідно із

заданими обмеженнями та початковими інваріантами, заданими вище. Розглянемо сигнатуру(сутність) типу Subject та його набір відношень:

```
abstract sig Subject {
  provide: set Skill,
  requires: set Skill
}
```

Фактично ми маємо два відношення provide та requires із множиною навичок:

- **provide** - відношення, що визначає множину навичок, яка дана дисципліна надає після прослуховування;
- **requires** - відношення, що визначає множину навичок, необхідних як базові для успішного засвоєння матеріалу курсу. Варто зазначити, що деякі дисципліни можуть мати значення none для цього поля, якщо така дисципліна не потребує специфічних попередньо засвоєних навичок.

Створення конкретних інстансів типу Skill є аналогічним до інстансів типу Student, тому приклад даного формалізму не наводимо. Розглянемо фрагмент інваріанту SubjectProvide, що описує формальний зв'язок між навчальними дисциплінами (Subject) та навичками (Skill):

```
fact SubjectProvide {
  English.requires = A2
  English.provide = B1
  BusinessProcessModeling.requires
= none
  BusinessProcessModeling.provide
= Business_Process_Modeling
  SQLFundamentals.requires =
Microsoft_Access
  SQLFundamentals.provide =
SQL_Knowledge
  ….
}
```

Фактично ми вказуємо, що базовий курс "English" надає рівень B1, але потребує початкових навичок рівня A2. Аналогічно дисципліна "Основи SQL" передбачає опанування навичкою Microsoft Access. Водночас дисципліна "Моделювання бізнес-процесів" надає навичку Business_Process_Modeling, але не вимагає попередніх знань.

Аналогічною є сигнатура careerAspir, пов'язана зі Student через відношення aspiresTo, а також інваріант, який визначає необхідні навички, актуальні на ринку праці.

```
abstract sig careerAspir  {
  necessary: set Skill
}

fact CareerAssignmentsNecessaty {
  Business_Analyst.necessary  = B1 +
Business_Process_Modeling  +
  SQL_Knowledge + ArchiMate_Language
  …
}
```

Обмеження та перевірка досяжності.

Обмеження в Alloy використовуються для формальної перевірки інваріантів, тобто властивостей, які мають виконуватись у будь-якому валідному стані моделі. Для побудови коректних персоналізованих траєкторій треба визначити, що кожен здобувач вищої освіти повинен мати конкретно сформульовану кар'єрну мету. Цей факт можна визначити наступним чином:

```
assert EachStudentHasCareerAspir {
 all b: Student | some b.aspiresTo
}
```

Застосування обмеження EachStudentHasCareerAspir дозволяє Alloy Evaluator [17] знайти множину контрприкладів (якщо це досяжно). У цьому випадку контрприкладом може бути ситуація, коли буде знайдено хоча б один здобувач вищої освіти, для якого не встановлено кар'єрні цілі. Аналогічно, не менш важливою для валідації є перевірка відсутності дисциплін, які не забезпечують хоча б однієї навички. Цей формалізм можна виразити наступним чином:

```
assert EachSubjectHasSkill {
all s: Subject | some s.provide
}
```

Перевірка відсутності контрприкладів лише частково свідчить про коректність побудованої мети. Важливо також перевірити, чи дійсно виконується умова:

$$\exists s \in \text{Student}\ s.\text{aspiresTo.necessary} \subseteq s.\text{enrollTo.provide} \quad (1)$$

Виразити це в синтаксисі Alloy можна наступним чином:

```
some {
 std:Student| std.aspiresTo.necessary in
 std.enrollTo.provide
}                                    (2)
```

Якщо ми хочемо окремо переглянути список набутих навичок для певного здобувача освіти або список необхідних навичок для визначеної кар'єрної мети, можемо застосувати відповідні команди в Alloy Evaluator:

- Alice.aspiresTo.necessary
- Alice.enrollTo.provide

Результат роботи представлений на рис. 1.

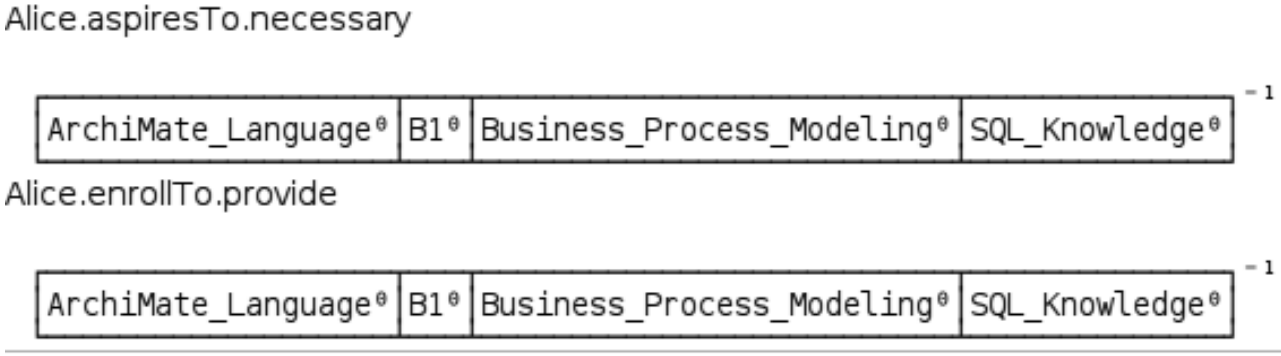

Рис. 1. Alloy Evaluator. Список набутих і необхідних навичок відповідно до професійних уподобань здобувача вищої освіти

Як видно з Рис. 1, список навичок є еквівалентним, що загалом свідчить про коректність роботи моделі та відсутність порушень. Перевіримо умову (1), яка буде дійсно свідчити, про те що існує хоча б один здобувач вищої освіти, який має потрібний набір навичок. На Рис. 2 представлено відображення цього факту.

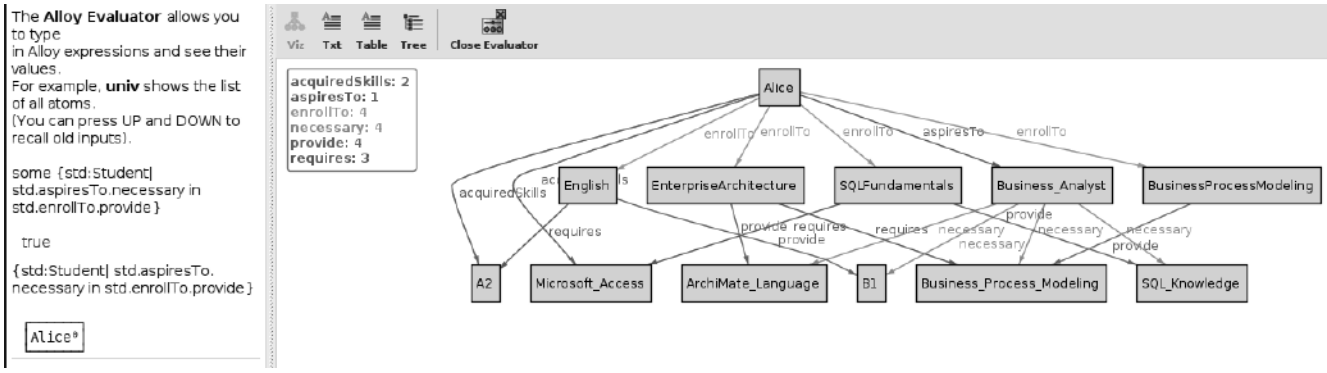

Рис. 2. Перевірка досяжності професійної мети здобувачем вищої освіти відповідно до алгоритму рекомендації курсів засобами Alloy

У цьому випадку ми бачимо, що справді існує один здобувач вищої освіти, який досяг необхідної кількості навичок, що свідчить про досяжність кар'єрної мети та правильність побудови персоналізованої освітньої траєкторії.

Наступним нашим кроком є демонстрація працездатності та правильності побудованого підходу відповідно до запропонованого алгоритму рекомендацій в роботі [18]. У Таблиці 1 представлено початкові навички та набуті після навчання на запропонованих алгоритмом курсів в синтаксисі мови Notation 3 [11].

Таблиця 1.

Навички здобувача вищої освіти до та після застосування алгоритму рекомендації

До застосування алгоритму рекомендації	**Після застосування алгоритму рекомендації**
ex:Alice a ex:Student ; ex:acquiredSkills ex:A2, ex:Business_Process_Modeling, ex:Microsoft_Access, ex:Python_Language ; ex:careerAspir ex:Business_Analyst, ex:Software_Architect .	ex:Alice a ex:Student ; ex:careerAspir ex:Business_Analyst; ex:acquiredSkills ex:A2, ex:ArchiMate_Language, ex:B1, ex:B2, ex:Business_Process_Modeling, ex:Communication_Advanced_English_Skills, ex:Communication_English_Skills, ex:Data_Modeling, ex:ETL_Processes, ex:Microsoft_Access, ex:PowerBI, ex:Python_For_Data, ex:Python_Language, ex:Requirement_Engineering, ex:SQL_Knowledge, ex:UML_Modeling ;

Варто зазначити, що наша модель стає з n-ітераціями. Прикладом таких ітерацій може бути послідовність запису на курс {English -> Advanced_English}, фактично нам треба acquiredSkills оновлювати динамічно, але рекурсія в Alloy має серйозні обмеження [19], тому введемо оновлення в код - стан для моделювання динаміки.

```
sig Student {
  acquired: State -> set Skill,
  aspiresTo: set careerAspir,
  enrollTo: set Subject
}
```

Застосувавши розроблену модель специфікації до даного набору початкових навичок та відповідного списку дисциплін отримаємо набір навичок у розрізі кожного стану. Приклад роботи зображено на Рис.3. Як бачимо, в State0 (початковий стан) еквівалентний з RDF-моделлю до застосування алгоритму.

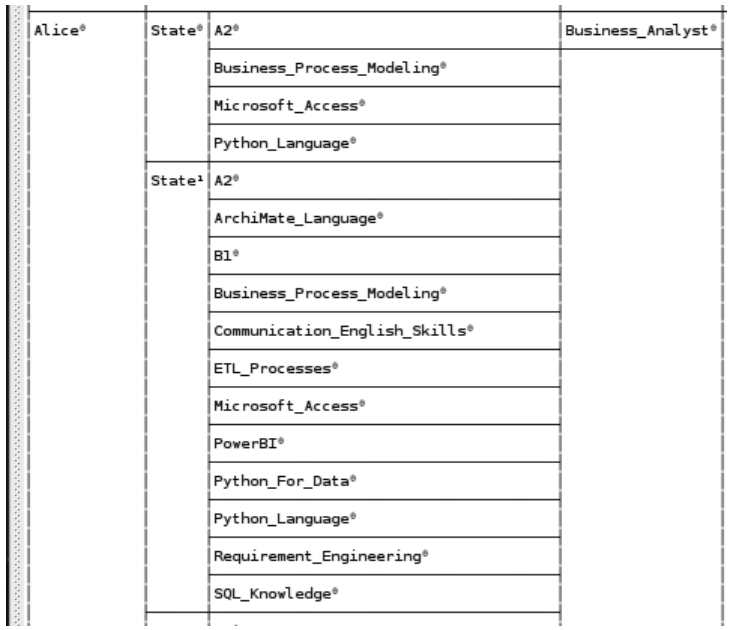

Рис. 3. Набуті навички здобувача освіти до станів (1-2)

Для перевірки досяжності набору навичок у кінцевому стані треба модифікувати умови (2) відповідно до нашого оновлення. Тому для перевірки вимоги, чи існує хоча б один здобувач освіти, який досяг кар'єрної мети, використано оновлену вимогу:

```
some {s:Student|
s.aspiresTo.necessary in s.acquiredSkill[last]
}                    (3)
```

Перевіряємо умови (3) за допомогою Alloy Evaluator. Результат роботи представлено на Рис. 4.

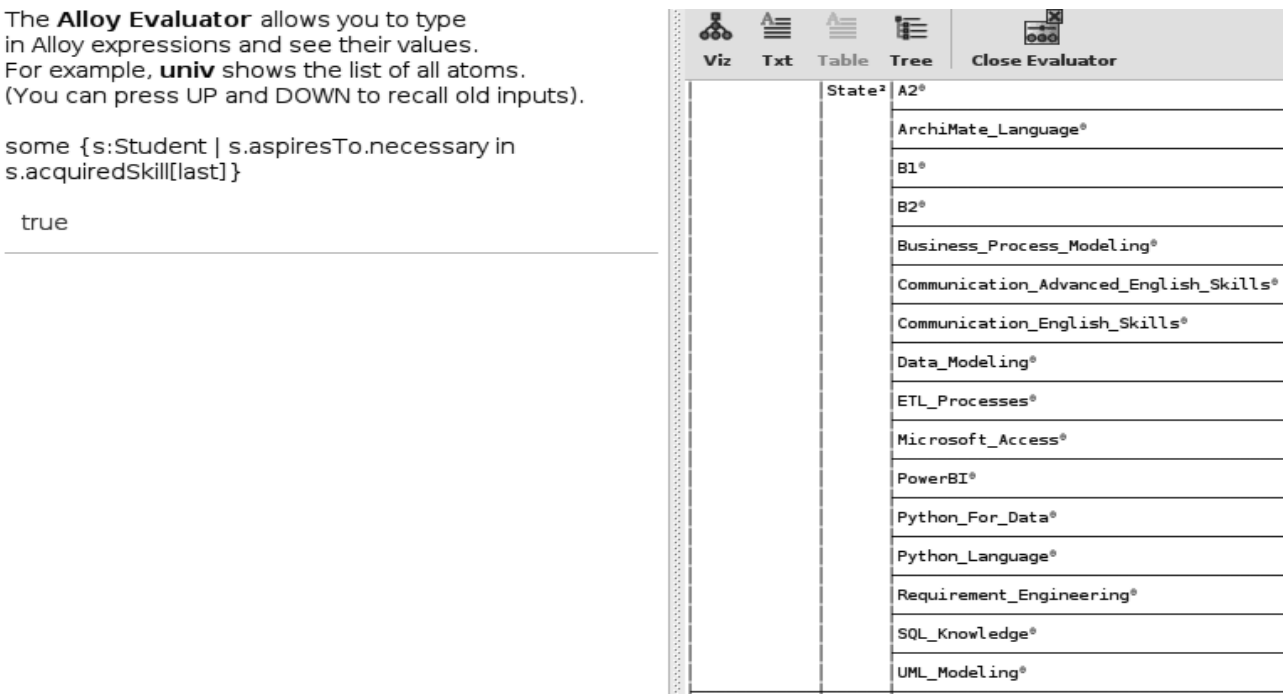

Рис. 4. Перевірка досяжності кар’єрної мети здобувача освіти

Якщо уважно переглянути результат роботи на Рис. 4, то можна побачити еквівалентний набір навичок у стані last=3 та в Таблиці №1 з RDF-моделлю після застосування алгоритму рекомендації. Даний факт демонструє правильність роботи алгоритму, представленого в роботі.

Висновки

У процесі дослідження було проаналізовано теоретичні засади та практичні підходи до перевірки коректності персоналізованих освітніх траєкторій у межах сучасної освітньої інженерії. З’ясовано, що персоналізоване навчання як інноваційна педагогічна парадигма потребує не лише розроблення індивідуального маршруту здобувача освіти, а й впровадження механізмів його верифікації, які забезпечують педагогічну доцільність, адаптивність і відповідність реальним цілям та потребам здобувача освіти.

Аналіз наукових джерел і прикладів застосування цифрових технологій дав змогу виокремити сучасні напрями у верифікації персоналізованих траєкторій. Розглянуто застосування алгоритмічних підходів, зокрема генетичних алгоритмів і нейронних мереж, які дають змогу формувати та перевіряти навчальні маршрути з урахуванням освітніх досягнень, мотивації та індивідуального прогресу здобувача. Досліджено використання семантичного аналізу мотиваційних і професійних даних, що забезпечує глибше розуміння мотиваційно-ціннісної сфери учня та підвищує валідність індивідуального навчального плану.

Під час нашого дослідження було розроблено формальну модель алгоритму побудови персоналізованих освітніх траєкторій засобами мови специфікації Alloy. В

результаті роботи було доведено коректність і досяжність розробленого алгоритму.

У подальших дослідженнях планується розширити список обмежень та перевірити правильність побудови персоналізованих освітніх траєкторій відповідно до цих обмежень, а також перевірити траєкторії за відсутності недетермінованої поведінки.

Література

1. Peng H., Spector J. Personalized adaptive learning: an emerging pedagogical approach enabled by a smart learning environment. *Smart Learning Environments*. 2019. Vol. 6. URL: https://doi.org/10.1186/s40561-019-0089-y
2. Tetzlaff L., Schmiedek F., Brod G. Developing personalized education: a dynamic framework. *Educational Psychology Review*. 2020. Vol. 33. С. 863–882. URL: https://doi.org/10.1007/s10648-020-09570-w
3. Shemshack A., Spector J. A comprehensive analysis of personalized learning components. *Journal of Computers in Education*. 2021. Vol. 8. С. 485–503. URL: https://doi.org/10.1007/s40692-021-00188-7
4. Алєксєєва С. Дидактичний алгоритм розроблення індивідуальної освітньої траєкторії. *Витоки педагогічної майстерності*. 2023. Вип. 32. С. 5–9. DOI: https://doi.org/10.33989/2075-146x.2023.32.292607
5. Лавренова М., Лалак Н., Молнар Т., Фенчак Л. Траєкторія розвитку здобувача початкової освіти: від теорії до практики. *Вісник Львівського університету. Серія педагогічна*. 2021. Вип. 35. С. 130–138. ISSN 2078-5526. URL: http://publications.lnu.edu.ua/bulletins/index.php/pedagogics/article/view/11314
6. Spivakovsky A., Petukhova L., Anisimova O., Horlova A., Kotkova V., Volianiuk A. ICT as a Key Instrument for a Balanced System of Pedagogical Education. *ICTERI.* 2020. URL: https://www.scopus.com/record/display.uri?eid=2-s2.0-85096419582&origin=AuthorNamesList&txGid=cff8324efb62ed78a72d9cda35befe10
7. Papamitsiou Z. Development and research of algorithms for the formation the individual educational trajectories of students in the digital educational platform. 2019. URL: https://www.semanticscholar.org/paper/Development-and-Research-of-Algorithms-for-the-the-Papamitsiou/be314ade46c3232508961eebf2b6e692527fa508
8. Kovaliuk T., Kobets N., Dvornyk V. Information technology for constructing individual educational trajectories based on latent-semantic analysis of motivational letters and professional achievements of students. 2020. С. 115–128. URL: https://www.semanticscholar.org/paper/Information-Technology-for-Constructing-Individual-Kovaliuk-Kobets/89c1e0bb06dcea223278d1f07fbec89a822917b7
9. W3C Semantic Web Wiki. URL: https://www.w3.org/2001/sw/wiki/Main_Page
10. RDF 1.1 Primer [Електронний ресурс]. W3C. 2014. URL: https://www.w3.org/RDF/
11. Notation3 (N3): A readable RDF syntax. URL: https://notation3.org/
12. Berners-Lee T. N3Logic: A logical framework for the World Wide Web. W3C. 2008. URL: https://www.w3.org/DesignIssues/N3Logic
13. Poltoratskyi M., Konnova O. Use of semantic web technologies for validation of educational models. *Information Technologies and Learning Tools*. 2025. Вип. 106(2). С. 94–106. URL: https://doi.org/10.33407/itlt.v106i2.5985
14. Shapes Constraint Language (SHACL) W3C Recommendation. 20 July 2017. URL: https://www.w3.org/TR/shacl/
15. SPARQL 1.1 Query Language. W3C Recommendation. 21 March 2013. URL: https://www.w3.org/TR/sparql11-query/
16. Alloy: A language and analyzer for software modeling URL: https://alloytools.org/
17. Alloy Analyzer – User Guide. URL: https://alloy.readthedocs.io/en/latest/tooling/analyzer.html
18. Полторацький М. Один з підходів до створення персоналізованого освітнього середовища. *Measuring and computing devices in technological processes*, 2026. Вип. 1, с. 77–86. https://doi.org/10.31891/2219-9365-2026-85-10
19. Alloy Documentation. Analyzer. URL: https://alloy.readthedocs.io/en/latest/tooling/analyzer.html

References

1. H. Peng, J. Spector, Personalized adaptive learning: an emerging pedagogical approach enabled by a smart learning environment, in: Smart Learning Environments, 2019. doi: 10.1186/s40561-019-0089-y.
2. L. Tetzlaff, F. Schmiedek, G. Brod, Developing personalized education: a dynamic framework, in: Educational Psychology Review, 33 (2020) 863–882. doi: 10.1007/s10648-020-09570-w.
3. A. Shemshack, J. Spector, A comprehensive analysis of personalized learning components,

in: Journal of Computers in Education, 8 (2021) 485–503. doi: 10.1007/s40692-021-00188-7.
4. S. Alekseeva, Didactic algorithm for designing an individual educational trajectory, in: Origins of Pedagogical Mastery, 32 (2023) 5–9. doi: 10.33989/2075-146x.2023.32.292607. [in Ukrainian]
5. M. Lavrenova, N. Lalak, T. Molnar, L. Fenchak, Development trajectory of primary education students: from theory to practice, in: Visnyk of Lviv University. Pedagogical Series, 35 (2021) 130–138. URL: http://publications.lnu.edu.ua/bulletins/index.php/pedagogics/article/view/11314
6. A. Spivakovsky, L. Petukhova, O. Anisimova, A. Horlova, V. Kotkova, A. Volianiuk, ICT as a key instrument for a balanced system of pedagogical education, in: ICTERI, 2020. URL: https://www.scopus.com/record/display.uri?eid=2-s2.0-85096419582&origin=AuthorNamesList&txGid=cff8324efb62ed78a72d9cda35befe10
7. Z. Papamitsiou, Development and research of algorithms for the formation of individual educational trajectories of students in the digital educational platform, 2019. URL: https://www.semanticscholar.org/paper/Development-and-Research-of-Algorithms-for-the-the-Papamitsiou/be314ade46c3232508961eebf2b6e692527fa508
8. T. Kovaliuk, N. Kobets, V. Dvornyk, Information technology for constructing individual educational trajectories based on latent-semantic analysis of motivational letters and professional achievements of students, 2020, pp. 115–128. URL: https://www.semanticscholar.org/paper/Information-Technology-for-Constructing-Individual-Kovaliuk-Kobets/89c1e0bb06dcea223278d1f07fbec89a822917b7
9. W3C Semantic Web Wiki, 2001. URL: https://www.w3.org/2001/sw/wiki/Main_Page
10. RDF 1.1 Primer [Electronic resource], W3C, 2014. URL: https://www.w3.org/RDF/
11. Notation3 (N3): A readable RDF syntax, URL: https://notation3.org/
12. T. Berners-Lee, N3Logic: A logical framework for the World Wide Web, W3C, 2008. URL: https://www.w3.org/DesignIssues/N3Logic
13. M. Poltoratskyi, O. Konnova, Use of semantic web technologies for validation of educational models, in: Information Technologies and Learning Tools, 106(2) (2025) 94–106. doi: 10.33407/itlt.v106i2.5985.
14. Shapes Constraint Language (SHACL) W3C Recommendation, 2017. URL: https://www.w3.org/TR/shacl/
15. SPARQL 1.1 Query Language. W3C Recommendation, 2013. URL: https://www.w3.org/TR/sparql11-query/
16. Alloy: A language and analyzer for software modeling, URL: https://alloytools.org/
17. Alloy Analyzer – User Guide, URL: https://alloy.readthedocs.io/en/latest/tooling/analyzer.html
18. M. Poltoratskyi, One approach to creating a personalized educational environment, in: Measuring and Computing Devices in Technological Processes, 1 (2026) 77–86. doi: 10.31891/2219-9365-2026-85-10.
19. Alloy Documentation. Analyzer. URL: https://alloy.readthedocs.io/en/latest/tooling/analyzer.html

Дата першого надходження до видання: 30.03.2026
Внутрішня рецензія отримана: 19.04.2026
Зовнішня рецензія отримана: 25.04.2026
Дата прийняття статті до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Полторацький Максим Юрійович,
доктор філософії (інформаційні технології), доцент
Poltoratskyi Maxym,
Ph.D. (information technology),
associate professor
https://orcid.org/0000-0001-9861-4438

Волянюк Анастасія Сергіївна,
викладачка кафедри педагогіки та психології дошкільної та початкової освіти
Volianiuk Anastasiya,
instructor, department of pedagogy and psychology
https://orcid.org/0000-0002-2890-4787

Місце роботи авторів:

Херсонський державний університет,
Kherson State University
тел. +380963102636,
E-mail:mpoltoratskyi@ksu.ks.ua
avolianiuk@ksu.ks.ua

ВИМОГИ ДО ОФОРМЛЕННЯ СТАТЕЙ

1. Загальні положення

У журналі "Проблеми програмування" публікуються наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, англійська. 16.07.2020 р. набули чинності положення Закону України «Про забезпечення функціонування української мови як державної». Відповідно до статті 22 «Державна мова у сфері науки» у наукових виданнях не повинно бути вміщено матеріалів іншими мовами, окрім державної, англійської та мов ЄС.

Обсяг статті - від 6 до 16 сторінок формату А4. Документ зберігається у форматі doc або docx.

Назва файлу включає транслітерацію прізвища автора (авторів), наприклад, “Petrenko.doc”.

Стаття надається без нумерації сторінок.

Для передачі до редакції тексту статті, ділової переписки та правки при коректурі автори користуються електронною поштою редакції: alengoro@isofts.kiev.ua, alengoro2022@gmail.com. Для надійності інформаційного обміну в умовах можливих відключень електрики – прохання надсилати матеріали на обидві електронні пошти одночасно. Телефон: +380 (96) 418 3082.

2. Оформлення файлу з текстом статті

При підготовці файлу використовуються: стиль нормальний (звичайний) або normal; шрифт Times New Roman, розмір шрифту 12 пт.; міжрядковий інтервал – 1,0; абзацний відступ -1,25 см; вирівнювання – по ширині. У тексті не допускається вирівнювання пропусками; розстановка переносів – автоматична. Формат паперу А4, розміри полів документа – 20 мм. Текст статті після зазначення авторів, назви і анотацій (двома мовами) має бути оформлений **у 2 колонки,** ширина яких – 7,86 см, а пробіл між ними – 1,27 см.

3. Послідовність розміщення та оформлення матеріалу статті

1. ***Верхній колонтитул:*** назва рубрики відповідно до переліку, прийнятому редакцією журналу *(пропонується авторами, остаточно уточняється редакцією).*

УДК *(зліва під рискою верхнього колонтитулу)**:*** індекс за універсальною десятковою класифікацією. ***DOI:*** в тому ж рядку правіше *(заповнюється редакцією)*.

Автори: ініціали та прізвища авторів, курсив *(світлий)*.

Заголовок 1 ***(назва статті)***: не містить абревіатур та строго відповідає змісту статті. Шрифт 15 пт, напівжирний, регістр верхній, вирівнювання по центру.

Анотація: 1800-2100 знаків враховуючи пробіли, не має бути абревіатур. Шрифт 10 пт, звичайний.

Ключові слова: не більше 10 слів, не містить абревіатур, подаються в називному відмінку, розділені комами. Шрифт 10 пт, звичайний.

УВАГА!

Автори, заголовок статті, анотація і ключові слова зазначаються **ДВІЧІ:** українською і англійською мовами. Спочатку мовою статті, потім іншою мовою.

2. ***Нижній колонтитул*** (тільки для першої сторінки) включає стандартну інформацію Copyright: перший рядок – прізвища авторів, рік; другий рядок – номер ISSN, назва журналу, рік, номер випуску.
3. **Заголовок 2** ***(назва розділу)***: шрифт 14 пт, напівжирний; абзац із центральним вирівнюванням, без переносів. Заголовки нижчого рівня *(пункти і т.п.)* у самостійний абзац не виділяються і проходять першим реченням текстового абзацу, шрифт 12 пт, напівжирний.

4. **Формули** створюються в редакторі Microsoft Equation 3.0 або MathType. Формули, на які є посилання в тексті, повинні мати наскрізну нумерацію. Номер формули друкується в круглих дужках біля краю правого поля. Розмір основного шрифту редактора формул – 12 пт. Розміри символів у формулах: звичайний – 12 пт, великий індекс – 9 пт, дрібний індекс – 7 пт, великий символ – 18 пт, дрібний символ – 11 пт. Не допускається масштабування формульних об'єктів. Великі формули мають бути розбиті на декілька рядків.

Наприклад:

$$\frac{\partial T}{\partial t}+\frac{u}{a\cos\varphi}\frac{\partial T}{\partial \lambda}+\frac{v}{a}\frac{\partial T}{\partial \varphi}+w\frac{\partial T}{\partial z}=\frac{\delta}{c_V\rho}, \quad (1)$$

де λ – довгота, φ – широта, z – висота над рівнем моря, $V=(u,v,w)$, a – радіус Землі, ω – швидкість добового обертання Землі, $F_f=(F_\lambda, F_\varphi, F_z)$.

5. **Рисунки** мають бути створені вбудованим редактором Word Picture або експортовані з прикладних програм Windows у графічних форматах (bmp, pcx, gif, jpg або tif). Рисунки розташовуються по центру. Нумерація рисунків здійснюється відповідно до порядку згадування у тексті. Нумеровані підписи розміщуються під рисунком з позначенням "Рис. ", далі вказується номер рисунка і текст підпису.
6. **Таблиці** мають бути підготовлені стандартним вбудованим в Word інструментарієм "Таблиця". Таблиці нумеруються за порядком згадування. На номер таблиці мають бути посилання в тексті. Номер таблиці вказується в окремому рядку з вирівнюванням по правій стороні (наприклад: "Таблиця 1"). Назви таблиць розміщуються над таблицею з вирівнюванням по центру. Мінімальний розмір шрифту в таблицях – 11 пт.

При посиланні на формулу, рисунок, таблицю або літературне джерело, використовуйте наступні позначення відповідно: (1), (1, 2); Рис.1, Рис.1, 2; Табл.1., Табл.1, 2; [1], [1, 2].

7. **Основний текст статті** має такі необхідні елементи:

- постановка проблеми в загальному вигляді і її зв'язок з важливими науковими або практичними завданнями;
- аналіз останніх досліджень і публікацій, у яких розпочато рішення даної проблеми і на які спирається автор, виділення невирішених раніше частин загальної проблеми, яким присвячується дана стаття;
- формулювання цілей статті (постановка задачі);
- виклад основного матеріалу дослідження з повним обґрунтуванням отриманих наукових результатів;
- висновки з даного дослідження і перспективи подальших розробок у даному напрямку;
- подяка (за наявності такої).

Застосований у статті маркований список (наведений вище) має наступні параметри: маркер має відступ 1,25 см, текст для першого рядка – 1,9 см. Аналогічні відступи слід підтримувати і для нумерованого списку.

8. **Література**: єдиний нумерований список джерел згідно ДСТУ 8302:2015 від 01.07.2016 р., шрифт 11 пт, відступ: спеціальний, навислий, 0,63 см.
9. **References:** література англійською мовою подається як список використовуваних джерел згідно *Harvard Style*. Джерела з заголовками на латиниці наводяться без перекладу. Для літератури джерел на мовах, що не використовують латинський алфавіт, необхідно забезпечити переведення назв джерел і вказати після них у

дужках мову оригіналу. Прізвища та ініціали авторів, слід транслітерувати за правилами як для закордонного паспорта.

Література, що надана другою мовою не враховується при підрахунку кількості сторінок статті. У випадках, коли список джерел включає джерела тільки однією мовою, він подається один раз.

10. ***Дата надходження статті*** позначається редакцією цифрами окремим рядком після слова «Одержано:»/”Received:”.

11. ***Дата надходження внутрішньої рецензії*** позначається редакцією цифрами окремим рядком після слів «Внутрішня рецензія отримана»/”Internal review received:”.

12. ***Дата надходження зовнішньої рецензії*** позначається редакцією цифрами окремим рядком після слів «Зовнішня рецензія отримана:»/” External review received:”.

Відомості про рецензентів конкретної статті не розголошуються для підвищення об’єктивності рецензування.

13. ***Дані про авторів*****:** мають починатися рядком “*Про авторів:*”, напівжирний курсив. Далі вказуються для кожного з авторів ПІБ повністю, вчений ступінь, наукове звання, посада, обов’язково номер ORCID (сайт ORCID http://orcid.org/). Особисті телефони та електронні пошти авторів вказуються тут тільки, якщо автор хоче, щоб вони були опубліковані в журналі.

Перелік авторів подається під номерами (надстроковим шрифтом), що відповідають нумерації місць роботи, де вони працюють (надстроковим шрифтом).

14. ***Дані про місце роботи авторів*****:** починаються рядком “*Місце роботи авторів:*”, напівжирний курсив. Далі вказуються місце роботи, телефон, електронна пошта, веб-сайт.

15. Обов’язково вказати мобільний телефон і e-mail відповідального виконавця для роботи з редактором при підготовці статті до друку. Ця інформація не публікується і призначена виключно для контакту редактора з авторами.

    Для полегшення підготовки статей, що задовольняють вищенаведеним вимогам, редакція журналу розробила файл шаблону статті “shablon.dot”, який можуть використовувати автори.