

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНИХ КОДОВИХ БАЗ

¹Сумський державний університет

Запропоновано методологію та подано результати розроблення інтелектуальної системи підтримки прийняття рішень (СППР) для автоматизованого оцінювання архітектурної якості об'єктно орієнтованого програмного забезпечення. Основою запропонованого підходу є інформаційно екстремальна інтелектуальна технологія (ІЕІТ), що забезпечує побудову адаптивного класифікатора класів за показниками згуртованості коду. Розроблена система поєднує кількісні метрики (LCOM2–LCOM5 і YALCOM) з інформаційним критерієм оптимізації, що дає змогу виконувати класифікацію програмних компонентів і пояснювати отримані результати через значення метрик.

Під час експериментів на вибірці зі ста Java класів досягнуто оптимальне значення інформаційного критерію $E^* = 0,55$, що відповідає високій точності розпізнавання. Система виявила 78 класів з потенційними архітектурними дефектами та зменшила кількість хибнопозитивних сповіщень порівняно з відомими статичними аналізаторами SonarQube та PMD. СППР може бути інтегрована у середовище CI/CD для безперервного контролю якості коду та надає пояснювальні рекомендації щодо рефакторингу.

Застосування інформаційно-екстремального підходу показало доцільність у задачах оцінювання архітектурної якості, проте потребує подальших досліджень на більших вибірках зі створення гібридних моделей і розширення набору програмних метрик.

Ключові слова: об'єктно-орієнтоване програмування; згуртованість; LCOM; інформаційно-екстремальна інтелектуальна технологія; ознака розпізнавання; інформаційний критерій Кульбака.

Вступ

У сучасних умовах цифрової трансформації суспільства якість програмного забезпечення стає одним із ключових чинників технологічного розвитку. Зростання складності програмних систем у поєднанні з широким використанням штучного інтелекту, наприклад, ChatGPT, Microsoft Copilot, вимагає надійних інструментів для оцінювання якості програмного коду. Попри автоматизацію низки процесів, саме людський фактор залишається вирішальним під час розроблення та підтримки ПЗ.

Низька якість архітектури або дизайну програмного забезпечення призводить до збільшення термінів розробки, зростання витрат і ускладнення підтримки. Тому виникає необхідність системного підходу до оцінювання якості об'єктно-орієнтованих кодових баз (ООП-кодів).

Одним із перспективних напрямів є створення інтелектуальних систем підтримки прийняття рішень (СППР), які здатні здійснювати самонавчання та формувати рекомендації для покращення структури коду.

Мета дослідження — підвищення об'єктивності та автоматизація процесу оцінки якості об'єктно-орієнтованого програмного забезпечення шляхом розробки моделі СППР. Система базується на інформаційно-екстремальній інтелектуальній технології (ІЕІТ), що дозволяє здійснювати навчання класифікатора на основі метрик згуртованості (Cohesion) для виявлення архітектурних дефектів коду з мінімальною ймовірністю помилки.



Завдання дослідження:

- провести аналіз наявних методів та метрик оцінки якості коду для визначення найінформативніших ознак розпізнавання (зокрема, метрик сімейства LCOM);
- розробити функціональну структуру СППР, яка використовує машинне навчання для адаптації до специфіки конкретного проєкту;
- побудувати та навчити класифікатор на основі IEIT-алгоритму, оптимізувавши параметри контейнерів класів розпізнавання (геометричні параметри в просторі ознак);
- експериментально перевірити ефективність запропонованої моделі на тестових наборах даних (Entity та Service класи) та оцінити точність класифікації.

Огляд літератури (State of the Art)

Оцінювання якості програмного забезпечення є одним з центральних напрямів сучасної інженерії програмних систем. Поширеним підходом до виявлення структурних дефектів (так званих *code smells*) є використання статичного аналізу коду, який передбачає визначення низки метрик і порівняння їх з фіксованими пороговими значеннями [1].

Галузеві інструменти, такі як SonarQube [2] і PMD [3], реалізують ці принципи, дозволяючи виявляти помилки, надлишкову складність, дублювання коду й порушення принципів SOLID. Проте дослідження Trautsch [4] та Shen [5] показали, що статичний аналіз часто супроводжується високою кількістю хибнопозитивних сповіщень, оскільки правила перевірки не враховують архітектурний контекст і специфіку проєкту.

Для підвищення достовірності оцінювання якості коду активно застосовуються методи машинного навчання. Зокрема, S. D. B. Effendi et al. [6] запропонували мовно-агностичну систему класифікації, що використовує статистичні ознаки програмних артефактів, а Mineau та Lalande [7] оцінили придатність ML-алгоритмів для аналізу мобільних застосунків. Алгоритми Support Vector Machine (SVM) та Random Forest демонструють вищу точність у порівнянні з традиційними пороговими підходами [8].

У складніших дослідженнях застосовуються глибинні нейронні архітектури, зокрема Convolutional Neural Networks і Recurrent Networks, які здатні враховувати контекст послідовності викликів методів у коді. Роботи Cairo [9] та Martins [10] доводять ефективність таких методів для прогнозування появи дефектів. До того ж, останні досягнення великих мовних моделей (LLM), таких як GPT-4 і Copilot, свідчать про їх потенціал для аналізу якості коду [11]. Проте зазначені системи вимагають великих обсягів навчальних вибірок і мають обмежену пояснюваність (проблема *black-box*).

Окрему групу досліджень становлять роботи, щодо метрик згуртованості та зв'язності об'єктно-орієнтованих систем. Класичні метрики LCOM (Lack of Cohesion of Methods), запропоновані Chidamber і Kemerer [12], стали основою для подальших модифікацій LCOM2–LCOM5 [13]—[14] і метрики YALCOM [15]. Ці показники кількісно оцінюють єдність методів і атрибутів класу, що безпосередньо відображає архітектурну якість системи.

Незважаючи на значний прогрес у використанні ML-моделей, залишаються відкритими питання: (1) як забезпечити пояснюваність процесу розпізнавання; (2) як стабільно працювати на невеликих вибірках; (3) як узгодити результати аналізу з експертними критеріями архітектурної якості. Ці проблеми стимулюють розвиток нових методів, здатних поєднати математичну строгість і інтерпретованість.

Одним з таких напрямів є інформаційно-екстремальна інтелектуальна технологія (IEIT), запропонована Довбишем і Шелеховим [16]. На відміну від класичних алгоритмів машинного навчання, IEIT базується на максимізації інформаційної ємності системи у процесі навчання, що забезпечує побудову безпомилкових вирішальних правил відносно навчальної вибірки. Цей підхід застосовувався для діагностики технічних систем і в завданнях розпізнавання образів [17]—[18], але практично відсутній у сфері оцінювання програмної якості. Особливість запропонованого підходу полягає у використанні інформаційно-екстремальної інтелектуальної технології для оцінювання якості об'єктно-орієнтованого коду за метриками згуртованості. У межах роботи ці метрики розглядаються як інформативні ознаки, що відображають внутрішню структурну цілісність класу, рівень взаємопов'язаності його методів та розподіл відповідальностей. Модель формує вирішальні правила з урахуванням архітектурного типу програмного компонента, що дає змогу адаптувати оцінювання до різних типів класів і пояснювати отримані результати через відхилення значень метрик.

Отже, існує наукова й практична потреба в побудові СППР, яка використовує метрики

згуртованості як інформативні ознаки для автоматизованого розпізнавання дефектних класів у кодових базах. Запропонована в цій роботі модель поєднує принципи ІЕІТ з метричними характеристиками об'єктно-орієнтованих систем, забезпечуючи адаптивність, інтерпретованість та можливість інтеграції у процес безперервної розробки програмного забезпечення.

Методологія дослідження та архітектура СППР

Запропонована інтелектуальна система підтримки прийняття рішень призначена для автоматизованого оцінювання якості об'єктно-орієнтованих кодових баз. Її основне завдання — виявлення класів, що містять архітектурні дефекти, так звані code smells, та формування рекомендацій щодо рефакторингу. Робота системи ґрунтується на використанні метрик згуртованості класів і на застосуванні інформаційно-екстремальної інтелектуальної технології як базового алгоритму навчання. Такий підхід є придатним для задач, у яких навчальна вибірка має обмежений обсяг, а результат класифікації має бути інтерпретованим.

Архітектура СППР складається з трьох основних рівнів (вхідного, опрацювання інформації та вихідного) і схематично може бути представлена схемою (рис. 1).

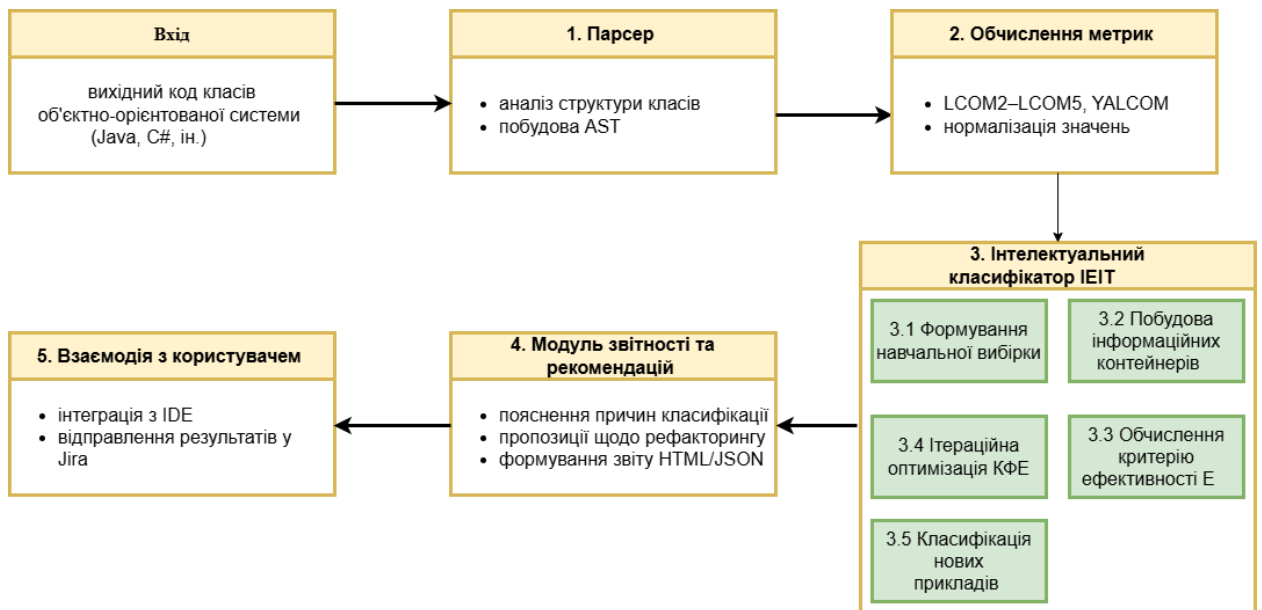


Рис. 1. Архітектура інтелектуальної СППР

На вхід надходить вихідний код програмного проєкту. Модуль «Парсер» аналізує структуру файлів об'єктно-орієнтованої мови Java чи іншої, виділяє класи, методи й атрибути та готує дані для обчислення метрик. Далі модуль «Обчислення метрик» розраховує показники згуртованості, зокрема метрики сімейства LCOM (LCOM2, LCOM3, LCOM4, LCOM5) та модифікацію YALCOM. Ці значення утворюють вектор ознак, який характеризує кожний клас з погляду його структурної цілісності. Отже, у межах запропонованої технології саме метрики згуртованості виступають основою ознакового опису програмного класу, оскільки вони дозволяють оцінити, наскільки узгоджено методи класу взаємодіють з його атрибутами та чи не концентрує клас надмірну кількість відповідальностей.

Після цього оброблений набір даних подається на інтелектуальний класифікатор, реалізований за принципами ІЕІТ. Формалізоване формулювання задачі в рамках блоку класифікатора, спрямоване на оцінку якості кодових баз, можна сформулювати так.

Нехай задано відомий алфавіт класів розпізнавання $\{X_m^0 \mid m = (1, M)\}$, що характеризує M типів класів ООП застосунку Java. Для кожного класу формується двовимірна вхідна навчальна матриця $\|y_{(m,i)}^{(i)}\|$ типу «об'єкт–ознака» на основі нормалізованих LCOM-метрик. Кожний рядок цієї навчальної матриці $\{y_{(m,i)}^{(i)} \mid j = (1, n)\}$ визначає векторне представлення екземпляра, де M — кількість ознак розпізнавання, а кожен стовпець відповідає ознаці розпізнавання, де n — розмір вибірки. До того ж, на етапі проєктування СППР, здатної до навчання, визначається структурований вектор просторово-часових операційних параметрів:

$$g = \langle x_m, r_m, \delta_{K,i}, \varphi_m \rangle, \quad (1)$$

де x_m — опорне (усереднене) векторне представлення екземпляра, вершина якого визначає геометричний центр контейнера класу; r_m — радіус гіперсферичного контейнера класу; $\delta_{K,i}$ — поле допуску для i -ї ознаки розпізнавання; φ_m — рівень вибору координат бінарного опорного векторного представлення класу.

Структурно вектор (1) складається з набору фенотипових операційних параметрів, які визначають геометрію контейнерів класів розпізнавання, реконструйованих у радіальному базисі простору ознак, та генотипових параметрів, які впливають на розподіл представлень екземплярів у просторі ознак, охоплюючи інші операційні параметри СППР.

Таким чином, в рамках технології ІЕІТ виникає проблема визначення в процесі машинного навчання СППР оптимальних значень операційних параметрів вектора (1), які забезпечують максимізацію інформаційно-орієнтованого функціонального критерію навчання СППР в операційній (допустимій) області його функціонування.

$$E_m^* = \max_{G} E_m, \quad (2)$$

де E_m^* — інформаційно-орієнтований критерій функціональної ефективності (КФЕ) навчання СППР для розпізнавання екземплярів класів; G — діапазон допустимих операційних параметрів для СППР. Як інформаційний КФЕ розглянуто модифіковану версію інформаційної міри Кульбака [14], яка має такий вигляд:

$$E_m^{(k)} = \log_2 \left(\frac{2 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d))}{\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)} \right) \left[1 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)) \right], \quad (3)$$

де $\alpha_m^{(k)}(d)$ — помилка I роду (хибнопозитивна) під час прийняття рішень на k -му кроці навчання; $\beta_m^{(k)}(d)$ — помилка II роду (хибнонегативна); d — міра відстані, що визначає радіус гіперсферичних контейнерів класів розпізнавання.

У режимі екзамєну, який реалізується безпосередньо в операційному контурі функціонування системи підтримки прийняття рішень, здійснюється процедура класифікації програмної реалізації до одного з класів наперед заданого алфавіту. У межах цього процесу система виконує пошук так званої робочої області параметрів, що визначається множиною значень радіусів контейнерів у просторі ознак, за яких досягається мінімальне значення похибки класифікації. Формалізація зазначеної області забезпечує побудову стійких до варіацій вхідних даних вирішальних правил, які мають інтерпретований характер і дозволяють не лише здійснювати класифікацію, але й пояснювати отримані результати. Таким чином, формується аналітичний базис для ухвалення обґрунтованих рішень щодо структурної відповідності програмних компонентів визначеним архітектурним типам.

Навчальна вибірка, використана для експериментальної верифікації запропонованої моделі, складалася зі ста програмних класів. Формування цієї вибірки здійснювалося за комбінованим підходом: 50 % об'єктів синтезовано із застосуванням великої мовної моделі GPT-5 на основі типових шаблонів об'єктно-орієнтованого проектування, тоді як інша половина відібрана з відкритих репозиторіїв Java-екосистеми на платформі GitHub, які побудовані із застосуванням фреймворку Spring. Добір класів до навчальної вибірки здійснювався з урахуванням їхньої належності до типових архітектурних компонентів Java/Spring-застосунків. До вибірки включалися лише класи, структура яких містила достатню кількість методів і атрибутів для коректного обчислення метрик згуртованості LCOM2–LCOM5 та YALCOM. З вибірки вилучалися службові, тестові та конфігураційні класи, оскільки їхня структура переважно визначається технічними вимогами фреймворку й не завжди відображає архітектурну якість прикладного коду. Після формування вибірки весь відібраний та згенерований код підлягав ретельному аналізу. За результатами цього аналізу в ручному режимі кожному програмному класу присвоювалися мітки, що дозволило рівномірно розподілити вибірку між двома новими цільовими категоріями: «якісний код» та «код, що потребує рефакторингу». Маркування класів здійснювалося на основі експертного аналізу їхньої структури, значень програмних метрик, кількості методів і атрибутів, а також розподілу відповідальностей між методами. До категорії «код, що потребує рефакторингу» відносилися класи з ознаками надмірної складності, низької внутрішньої зв'язаності або порушення принципу єдиної відповідальності. Класи без виражених

структурних відхилень маркувалися як «якісний код». Для кожного класу обчислювався набір з п'яти метрик згуртованості, що характеризують внутрішню зв'язаність програмних компонентів. Отримані значення підлягали процедурі нормалізації в інтервалі $[0;1]$, що забезпечувало їхню порівнянність та коректність подальшої обробки. Надалі вибірку розподілено на навчальну (80 %) та тестову (20 %) підмножини. Як критерій ефективності навчання використовувався інформаційний показник E^* , максимізація якого досягалася шляхом оптимізації параметрів моделі, зокрема радіусів контейнерів і вибору базових векторів класів.

Запропонована система може бути ефективно інтегрована в сучасні середовища розроблення програмного забезпечення, зокрема в конвеєри безперервної інтеграції та доставки (CI/CD). Інтеграція передбачає включення відповідного програмного модуля до корпоративної інфраструктури контролю версій або інструментів автоматизованого складання та тестування. У процесі кожного циклу аналізу вихідного коду система автоматично виконує синтаксичний і семантичний парсинг програмних файлів, формує вектор ознак (метрик) для кожного класу та здійснює його класифікацію відповідно до сформованих вирішальних правил. Результатом функціонування є структурований аналітичний звіт, що містить перелік досліджуваних класів з прогнозованою оцінкою рівня якості та ступеня відповідності еталонним архітектурним характеристикам. Для класів, ідентифікованих як такі, що потребують рефакторингу, система додатково генерує інтерпретовані пояснення, які базуються на значеннях метрик, наприклад, перевищення порогових значень згуртованості, як-от $LCOM4 > 1$, що може слугувати підставою для рекомендацій щодо декомпозиції або перегляду структури методів.

Блок-схема інтеграції системи підтримки прийняття рішень у CI/CD-конвеєр показана на рис. 2.

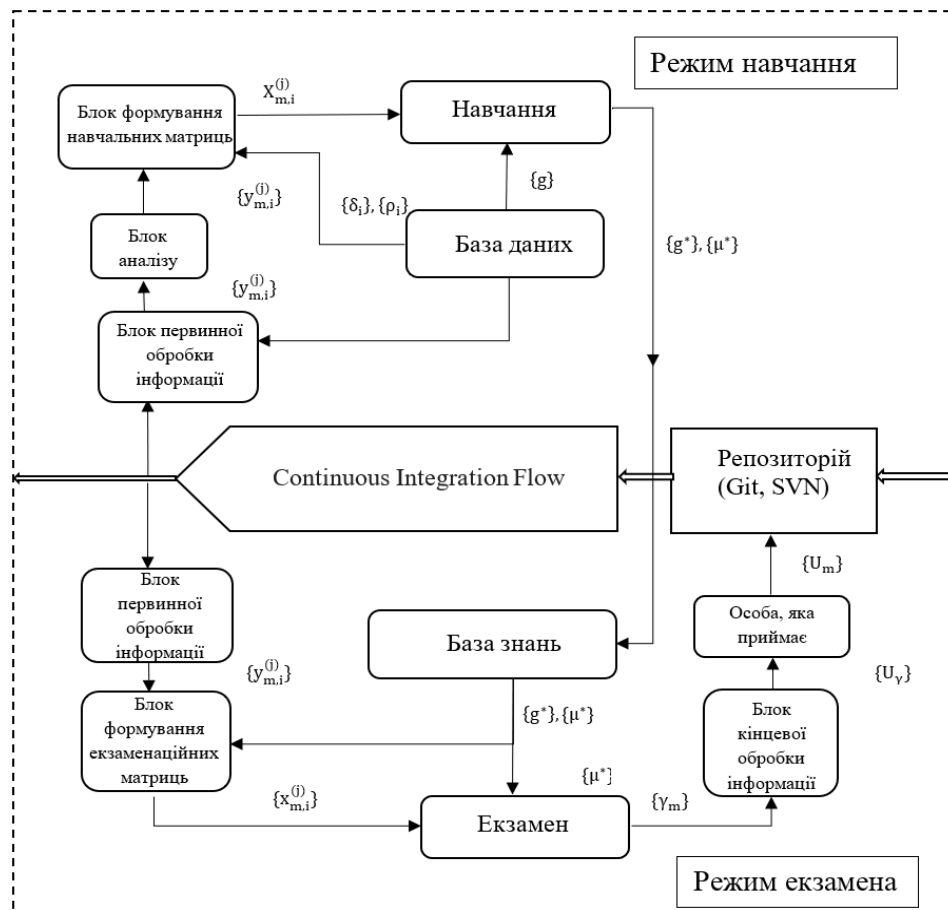


Рис. 2. Блок-схема інтеграції СППР в CI/CD процесі

У блок-схемі відображено ключові інформаційні потоки, взаємодію між основними компонентами системи, а також відповідні робочі змінні, що визначають процес обробки даних. Запропонована схема враховує сучасні підходи до побудови інтелектуальних систем аналізу програмного забезпечення, включаючи вимоги до масштабованості, автоматизації процесів контролю якості та забезпечення інтерпретованості результатів аналізу.

Розроблена структура (рис. 2) дозволяє системі функціонувати як в режимі навчання, так і екзамену. Робота СППР у режимі навчання базується на навчальній матриці, класифікованій апіорі

відповідно до архівних трендів розроблених кодових баз. «Блок первинної обробки інформації» здійснює нормалізацію, фільтрацію та структурування даних. Далі, вибірккові послідовності ознак розпізнавання $\{y_{(m,i)}^{(i)}\}$ вводяться в «Блок аналізу» для оцінки їхньої статистичної стійкості та однорідності за відомими статистичними критеріями. Якщо умови статистичної сталості або однорідності вибірок не виконуються, «Блок аналізу» блокує доступ до інформації в СППР. З виходу «Блок аналізу» дані надходять до «Блоку формування навчальних матриць», де формуються вхідні багатовимірні бінарні матриці навчання для заданого алфавіту класів розпізнавання $\|x_{(m,i)}^{(i)}\|$ типу «об’єкт–властивість». Навчальна матриця та значення за замовчуванням інших параметрів, що зберігаються в «Базі даних», надходять до блоку «Навчання», де відбувається оптимізація параметрів контейнерів класів розпізнавання для побудови правил прийняття рішень. Оптиміальні параметри навчання $\{g^*\}$ та правила прийняття рішень $\{\mu^*\}$ надходять до «Бази знань».

Під час роботи СППР у режимі екзамену «Блок первинної обробки інформації» перетворює інформацію про поточний функціональний стан кодової бази у форму, зручну для подальшої обробки комп’ютером, та передає її до «Блоку формування екзаменаційних матриць», яка формує бінарний вектор-реалізацію $\{x_i^{(j)} \mid i = \overline{1, N}\}$, використовуючи оптиміальні параметри $\{g^*\}$ та $\{\mu^*\}$, що надходять з «Бази знань». Вектор, сформований на виході, потрапляє до блоку «Екзамен», де відбувається визначення належності до одного з класів заданого алфавіту $\{x_m^0\}$ згідно з вирішальними правилами, побудованими на етапі навчання СППР. Виходом блоку «Екзамен» є гіпотези $\{y_m \mid m = \overline{1, M}\}$ про поточний функціональний стан кодової бази. Гіпотеза y_m аналізується відповідальною особою, яка видає команду $\{U_\gamma\}$ на «Блок кінцевої обробки інформації» для генерації рекомендації $\{U_m\}$ з метою покращення показника якості кодової бази Е.

Запропонована методологія відрізняється пояснюваністю результатів і здатністю працювати на обмежених вибірках навчальних даних. Геометричне представлення класів у просторі ознак дозволяє наочно ідентифікувати відхилення від типових архітектурних патернів, а інформаційно-екстремальний алгоритм забезпечує адаптивне налаштування параметрів класифікації без потреби в повторному навчанні всієї моделі, через що СППР є придатною для використання в реальних умовах процесів розроблення ПЗ з урахуванням динаміки еволюції кодової бази.

Результати експериментів та їх обговорення

Експериментальні дослідження проводилися з метою оцінювання ефективності розробленої системи підтримки прийняття рішень (СППР) у задачі класифікації класів програмного коду за рівнем згуртованості. Для налаштування моделі використано навчальну вибірку, сформовану зі ста класів, описаних раніше, де кожний елемент представлено п’ятьма нормалізованими ознаками — метриками сімейства LCOM2–LCOM5 та YALCOM.

З метою аналізу алгоритму навчання СППР досліджено динаміку усередненого інформаційного критерію (3) відносно радіусів реконструйованих контейнерів (рис. 3).

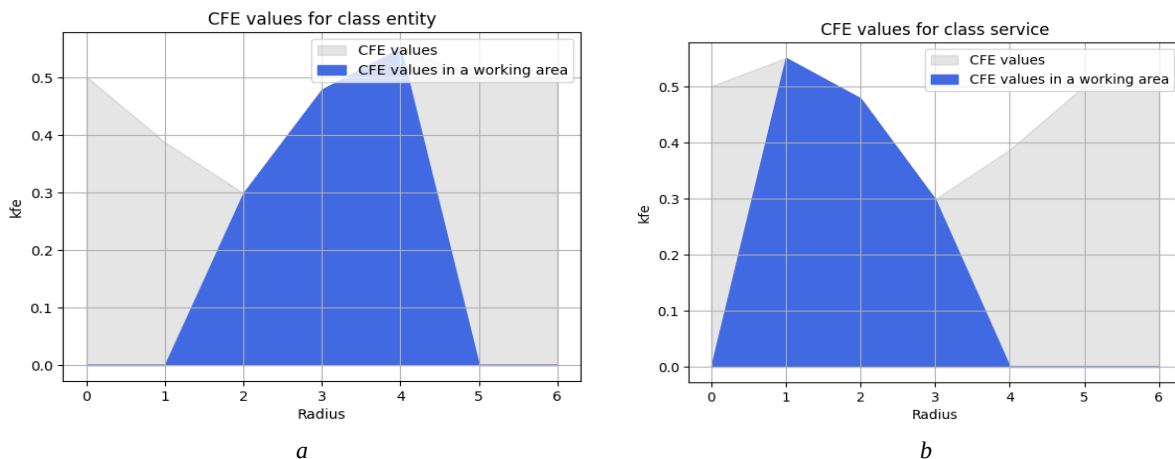


Рис. 3. Графіки залежності усередненого КФЕ від радіуса класу: a — class X_1^0 ; b — class X_2^0

Аналіз рис. 3 демонструє, що реалізація базового алгоритму, навіть без оптимізації поля контрольних допусків, характеризується наявністю операційної (допустимої) області, в якій значення точності першого та другого порядку відповідно перевищують похибки першого та другого роду. Темні області позначають операційні області визначення інформаційної критеріальної функції.

При значенні параметра поля допуску контролю для ознак розпізнавання $\delta_k = 50$ (вираженого у відсотках від номінального значення ознаки), що використовується під час формування вхідної двійкової матриці, оптимальні радіуси контейнерів у кодових одиницях становлять $d_1^* = 4$ та $d_2^* = 1$ відповідно. У цьому випадку глобальний максимум інформаційно-орієнтованої КФЕ в операційній області визначення функції (2) дорівнює $\bar{E}^* = 0,55$, що вказує на достатньо високі асимптотичні характеристики СППР, які визначаються функціональною ефективністю процесу навчання системи.

Правила прийняття рішень, побудовані СППР під час машинного навчання за розглянутим алгоритмом, не є безпомилковими відносно навчальної матриці, оскільки застосована система допуску керування для ознак розпізнавання є апіорі неоптимальною. Для побудови правил прийняття рішень, які є безпомилковими відносно навчальної матриці, відповідно до принципу відкладених рішень, необхідно оптимізувати інші операційні параметри, які впливають на функціональну ефективність навчання СППР. Такі параметри можуть включати, наприклад, поле допуску керування для ознак розпізнавання, словник ознак тощо.

У процесі навчання здійснювалося ітераційне коригування геометричних параметрів інформаційних контейнерів, що описують класи розпізнавання. Зміна радіусів контейнерів дозволила визначити оптимальну конфігурацію моделі, за якої досягається максимальна інформаційна ємність системи. Таке значення свідчить про те, що модель здатна відокремлювати класи «якісний код» та «код, що потребує рефакторингу» з імовірністю правильного розпізнавання понад 85 %.

Для обґрунтування ефективності розробленої системи та проведення порівняльного аналізу використано відомі інструменти статичного аналізу коду, а саме SonarQube та PMD. У межах експериментального дослідження оцінювалася здатність зазначених засобів ідентифікувати класи, що потребують рефакторингу, на основі аналізу їхніх структурних і поведінкових характеристик. Для цього застосовано вбудовані механізми виявлення дефектів, які реалізовані у відповідних інструментах. У системі PMD детектування здійснювалося шляхом виявлення антипатерна “God object”, який реалізований як вбудоване правило, що характеризує класи з надмірною концентрацією відповідальностей. У свою чергу, в SonarQube аналіз виконувався з використанням сукупності метрик, серед яких “Too many methods”, “Cognitive Complexity”, “Too many fields” та “Feature Envy”, що дозволяють комплексно оцінити рівень складності, зв’язності та розподілу відповідальностей у класах. За результатами виконаного аналізу для кожного інструмента обчислювалося середнє значення точності розпізнавання дефектного коду, що забезпечило можливість їх коректного порівняння за єдиним критерієм. Перевага запропонованої технології полягає в тому, що вона не обмежується застосуванням фіксованих правил або порогових значень окремих метрик. На відміну від SonarQube та PMD, які здебільшого виявляють дефекти на основі наперед визначених евристик, розроблена система адаптивно формує вирішальні правила в процесі навчання. Це дає змогу враховувати специфіку конкретної кодової бази та адаптувати вимоги до якості залежно від архітектурного типу програмного компонента. Завдяки цьому система зменшує кількість хибнопозитивних сповіщень і надає пояснювані рекомендації щодо рефакторингу на основі відхилень відповідних програмних метрик.

З метою валідації розробленої моделі та перевірки узгодженості отриманих результатів застосовано метод перехресної перевірки. Відповідна процедура передбачала п’ятикратне повторення експерименту з почерговим використанням підвибірок даних у ролі навчальної та тестової, що забезпечує об’єктивнішу оцінку узагальнювальної здатності моделі. На кожній ітерації формувалися різні комбінації навчальної та тестової вибірок, після чого виконувалося навчання моделі та її подальше тестування. Отримані результати усереднювалися за всіма ітераціями, що дозволило мінімізувати вплив випадкових факторів і забезпечити статистичну надійність оцінок. Узагальнені результати експериментальних досліджень показано на рис. 4.

Результати експериментальних досліджень свідчать про перевагу запропонованого підходу. Розроблена система підтримки прийняття рішень забезпечила правильну ідентифікацію 78 з 100 розглянутих екземплярів, тоді як відповідні показники для SonarQube та PMD становили 62 і 57 відповідно.

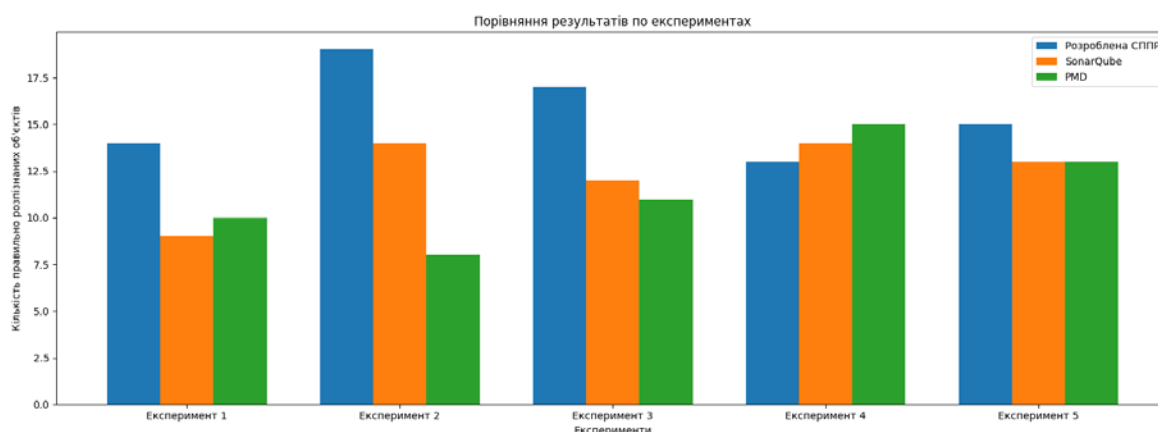


Рис. 4. Результати порівняння систем аналізу

Додатково встановлено, що кількість хибнопозитивних сповіщень (помилки I роду) у запропонованій системі є меншою приблизно у 1,5 раза порівняно з досліджуваними аналогами. Це знижує інформаційне навантаження на розробника під час аналізу результатів та підвищує практичну цінність отриманих оцінок.

Для наочності у таблиці подано типовий результат оброблення вибірки класів.

Приклад результатів класифікації класів проєкту

Код класу	Тип класу	LCOM3	LCOM5	YALCOM	Прогноз СППР	Рекомендація
C01	Entity	0,22	0,15	0,18	якісний код	—
C05	Service	0,84	0,80	0,75	потребує рефакторингу	поділ сервісів
C09	Entity	0,35	0,28	0,23	якісний код	—
C14	Service	0,90	0,88	0,81	потребує рефакторингу	перегрупування методів

Як випливає з таблиці, система здатна диференціювати класи навіть за незначних відмінностей у значеннях метрик. Високі показники LCOM5 та YALCOM для класів типу Service вказують на низьку взаємопов'язаність методів і відсутність спільних атрибутів, що типово для коду з порушенням принципу Single Responsibility. У таких випадках СППР формує зрозумілу рекомендацію для розробника щодо розділення класу на підкласи з чітко визначеними обов'язками.

Отримані результати підтверджують доцільність застосування ІЕІТ у задачах статичного аналізу програмного коду. Запропонований підхід потребує менший обсяг навчальних даних ніж типові нейронні моделі, і водночас зберігає інтерпретованість результатів. Завдяки геометричній природі моделі користувач може візуалізувати розташування класів у просторі ознак та виявити «перекриття» між контейнерами, що особливо може використовуватися для аналізу граничних випадків класифікації.

Таким чином, експериментальні дослідження підтвердили, що СППР на основі ІЕІТ забезпечує стабільну класифікацію класів за рівнем згуртованості, скорочує кількість хибних сповіщень порівняно з традиційними інструментами й надає пояснювані рекомендації щодо підвищення архітектурної якості програмного забезпечення.

Висновки

Розроблена функціональна модель продемонструвала задовільний рівень ефективності навчання. Максимальне значення інформаційного критерію ($E^* = 0,55$), досягнуте під час експерименту, свідчить про те, що система успішно визначила діапазон радіальних параметрів («робочу область»), де похибка класифікації мінімізована. Це підтверджує, що алгоритм інформаційно-екстремального навчання може ефективно оптимізувати правила прийняття рішень для оцінки якості програмного забезпечення, розрізняючи архітектурні компоненти, такі як класи Entity та Service.

Результати експерименту свідчать, що метрики згуртованості можуть бути використані як інформативні ознаки для класифікації об'єктно-орієнтованих класів. На відміну від простих метрик рядків коду, LCOM фіксує структурну складність та семантичну цілісність класу. Різне кластеризування класів Entity проти Service у просторі ознак вказує на те, що ці архітектурні шаблони, які характеризуються унікальним набором ознак взаємодії методів і полів, що утворюють його

структурний профіль, який ідентифіковано розробленою системою підтримки прийняття рішень.

На відміну від стандартних інструментів статичного аналізу, наприклад, SonarQube, PMD, які ґрунтуються на попередньо визначених, жорстких порогах для метрик, запропонована модель СППР використовує адаптивний підхід навчання. Аналізуючи статистичні тенденції певної кодової бази, що зберігається в блоці «База даних тенденцій», система адаптує свої правила прийняття рішень. Це дає змогу відрізнити обґрунтовано складні класи від класів з ознаками структурних недоліків і формувати більш контекстно-залежну оцінку якості.

Варто зазначити кілька обмежень. По-перше, матрицю навчання побудовано з використанням набору даних зі 100 класів, частково згенерованих за допомогою LLM, що є спрощеним середовищем порівняно з великомасштабними застарілими системами. По-друге, дослідження зосереджувалося переважно на метриках згуртованості; майбутні ітерації повинні включати метрики зв'язності та складності для покращення інформаційної ємності системи. Нарешті, поточний експеримент обмежувався лише Java-подібними структурами ООП.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- [1] A. Trautsch, S. Herbold, and J. Grabowski, "Are automated static analysis tools worth it?" *Empirical Software Engineering*, vol. 28, no. 3, p. 66, 2023. <https://doi.org/10.1007/s10664-023-10301-2>.
- [2] X. Liu, and G. Woo, "Using SonarQube to enhance the accuracy of programming assignment evaluation," in *Proc. INTED2019*, 2019. <https://doi.org/10.21125/inted.2019.1310>.
- [3] S. A. Licorish, and M. Wagner, "Combining GIN and PMD for code improvements," in *Proc. GECCO Companion*, 2022, pp. 790-793. <https://doi.org/10.1145/3520304.3528772>.
- [4] A. Trautsch, S. Herbold, and J. Grabowski, "Empirical analysis of static analysis efficiency," *Empirical Software Engineering*, vol. 28, p. 66, 2023. <https://doi.org/10.1007/s10664-023-10301-2>.
- [5] M. Shen, A. Pillai, B. A. Yuan, J. C. Davis, and A. Machiry, "An empirical study on the use of static analysis tools in embedded software," *arXiv preprint arXiv:2310.00205*, 2023. <https://doi.org/10.48550/arXiv.2310.00205>.
- [6] S. D. B. Effendi et al., "A language-agnostic framework for mining static analysis rules from code changes," in *Proc. ICSE-SEIP*, 2023. <https://doi.org/10.1109/ICSE-SEIP58684.2023.00035>.
- [7] J.-M. Mineau and J.-F. Lalande, "Evaluating the reusability of Android static analysis tools," in *Proc. Int. Conf. Software Reuse*, 2024, pp. 153-170. https://doi.org/10.1007/978-3-031-66459-5_10.
- [8] A. Ampatzoglou, et al., "Applying the Single Responsibility Principle in industry," in *Proc. EASE Conf.*, 2019, pp. 347-352. <https://doi.org/10.1145/3319008.3320125>.
- [9] G. Cairo, G. Carneiro, and M. Monteiro, "The impact of code smells on software bugs: A systematic review," *Information*, vol. 9, no. 11, p. 273, 2018. <https://doi.org/10.3390/info9110273>.
- [10] J. Martins, C. Bezerra, A. Uchôa, and A. Garcia, "How do code smell co-occurrences removal impact internal quality attributes?" in *Proc. Brazilian Symp. Software Engineering*, 2021, pp. 54-63. <https://doi.org/10.1145/3474624.3474642>.
- [11] I. Chang, W. C. Choi, and I. C. Choi, "A systematic review of AIGC opportunities in programming," in *Proc. Int. Conf. Big Data and Education*, 2024, pp. 29-35. <https://doi.org/10.1145/3704289.3704301>.
- [12] S. R. Chidamber, and C. F. Kemerer, "Towards a metrics suite for object-oriented design," in *Proc. OOPSLA*, 1991, pp. 197-211. <https://doi.org/10.1109/32.295895>.
- [13] B. Henderson-Sellers, *Object-Oriented Metrics: Measures of Complexity*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995. [Online] Available: https://books.google.com.ua/books/about/Object_oriented_Metrics.html?id=j7sgAQAAIAAJ.
- [14] M. Hitz, and B. Montazeri, "Measuring product attributes of object-oriented systems," in *Proc. ESEC*, 1995, pp. 124-136. https://doi.org/10.1007/3-540-60406-5_11.
- [15] T. Sharma, and D. Spinellis, "Do we need improved code quality metrics?," *arXiv preprint arXiv:2012.12324*, 2020. <https://doi.org/10.48550/arXiv.2012.12324>.
- [16] A. S. Dovbysh, and I. V. Shelekhov, *Osnovy teorii rozpoznavannia obraziv*. Sumy, Ukraine: SumDU, 2015. ISBN: 978-966-657-597-8. [Online] Available: https://document.kdu.edu.ua/info_zab/2824.pdf.
- [17] I. V. Shelekhov, "Information-extreme machine learning system for functional diagnostics," *Radioelectronics and Com*.
- [18] A. S. Dovbysh, V. K. Obodiak, and I. V. Shelekhov, *Suchasni informatsiini tekhnologii v kiberbezpeki*. Sumy, Ukraine: SumDU, 2021. [Online] Available: https://essuir.sumdu.edu.ua/bitstream-download/123456789/82619/3/Obodiak_kiberbezpeka.pdf.

Рекомендована кафедрою комп'ютерних систем управління ВНТУ

Дата надходження 6.04.2026

Дата прийняття після доопрацювання 16.04.2026

Дата прийняття до друку після рецензування 6.06.2026

Дата публікації 2.10.2026

Паращенко Владислав Анатолійович — аспірант кафедри комп'ютерних наук, e-mail: vladyslav.parashchenko@student.sumdu.edu.ua . <https://orcid.org/0000-0002-0103-0434> ;

Берест Олег Борисович — канд. техн. наук, старший викладач кафедри комп'ютерних наук, email: o.berest@cs.sumdu.edu.ua. <https://orcid.org/0000-0003-0183-9871> .

Сумський державний університет, Суми

V. A. Parashchenko¹
O. B. Berest¹

Intelligent Decision Support System for Evaluating the Quality of Object-Oriented Codebases

¹Sumy State University

The paper presents the methodology and results of developing an intelligent decision support system (DSS) for the automated assessment of the architectural quality of object-oriented software. The proposed approach is based on information-extreme intelligent technology (IEIT), which provides the construction of an adaptive classifier of program classes according to code cohesion indicators. The developed system combines quantitative metrics, namely LCOM2–LCOM5 and YALCOM, with an information-based optimization criterion, which makes it possible to classify software components and explain the obtained results through metric values.

During the experiments on a dataset of one hundred Java classes, the optimal value of the information criterion $E^ = 0.55$ was achieved, which corresponds to a high recognition accuracy. The system identified 78 classes with potential architectural defects and reduced the number of false-positive alerts compared with well-known static analyzers, such as SonarQube and PMD. The DSS can be integrated into a CI/CD environment for continuous code quality monitoring and provides explainable refactoring recommendations.*

The application of the information-extreme approach demonstrated its efficiency in architectural quality assessment tasks; however, further research is required using larger datasets, hybrid models, and an extended set of software metrics.

Keywords: object-oriented programming; cohesion; LCOM; information-extreme intelligent technology; recognition feature; Kullback information criterion.

Parashchenko Vladyslav A. — Post-Graduate Student of the Chair of Computer Science, e-mail: vladyslav.parashchenko@student.sumdu.edu.ua . <https://orcid.org/0000-0002-0103-0434>;

Berest Oleh B. — Cand. Sc. (Eng.), Senior Lecturer of the Chair of Computer Science, e-mail: o.berest@cs.sumdu.edu.ua . <https://orcid.org/0000-0003-0183-9871>