DOI https://doi.org/10.32689/maup.it.2026.1.4
УДК 004.056:005.8

МЕТОД РИЗИК-ОРІЄНТОВАНОГО УПРАВЛІННЯ ЯКІСТЮ ІТ-ПРОДУКТУ НА ОСНОВІ АНАЛІЗУ ДАНИХ ТА МАШИННОГО НАВЧАННЯ В МЕЖАХ SDLC

Ю. В. Кіш, І. М. Лях

A METHOD OF RISK-BASED QUALITY MANAGEMENT OF AN IT PRODUCT BASED ON DATA ANALYSIS AND MACHINE LEARNING WITHIN SDLC

Yurii Kish, Ihor Liakh

Анотація

Стаття присвячена розробленню методу ризик-орієнтованого управління якістю ІТ-продукту в межах життєвого циклу розробки програмного забезпечення. Об'єктом дослідження є процес забезпечення якості програмних систем за умов невизначеності та обмежених ресурсів, а проблемою, що вирішувалася, є відсутність інтегрованого механізму кількісного оцінювання, прогнозування та пріоритизації ризиків якості на основі даних SDLC. У роботі запропоновано формалізовану математичну модель, у якій ризик інтерпретується як імовірнісна характеристика дефектності програмних компонентів, а також механізм інтеграції оцінок ризику в процеси забезпечення якості у вигляді адаптивних контрольних точок і програмний прототип системи підтримки прийняття рішень. Отримані результати дозволили вирішити зазначену проблему завдяки поєднанню імовірнісного моделювання, методів машинного навчання та ризик-орієнтованої пріоритизації тестування, що забезпечує перехід від статичного контролю до data-driven управління якістю. Результати пояснюються встановленням зв'язку між метриками програмних артефактів і ймовірністю дефектів, що дає змогу ранжувати компоненти за рівнем ризику, формувати реєстр ризиків і оптимізувати розподіл ресурсів верифікації. Експериментальна апробація на відкритому наборі NASA Metrics Data Program засвідчила, що модель досягає ROC-AUC = 0,669 і PR-AUC = 0,382, а в сценарному аналізі забезпечує recall@top-k ≈ 0,38 проти ≈ 0,37 для LOC-орієнтованого підходу та ≈ 0,18 для випадкової стратегії. Практичне використання результатів доцільне в системах управління якістю програмного забезпечення, DevOps-аналітиці, середовищах CI/CD і процесах пріоритизації тестування за наявності історичних даних про метрики коду, результати перевірок, характеристики модулів і дефектність, а також обмежених ресурсів контролю.

Ключові слова: *ризик-орієнтоване управління якістю, SDLC, машинне навчання, прогнозування дефектів, DSS, метрики програмного забезпечення, data-driven.*

Abstract

The article is devoted to the development of a risk-oriented quality management method for IT products within the software development life cycle. The object of the study is the process of ensuring software quality under conditions of uncertainty and limited resources, while the problem addressed is the lack of an integrated mechanism for quantitative assessment, prediction, and prioritization of quality risks based on SDLC data. The paper proposes a formalized mathematical model in which risk is interpreted as a probabilistic characteristic of software component defectiveness, as well as a mechanism for integrating risk assessments into quality assurance processes in the form of adaptive quality gates and a decision support system prototype. The obtained results made it possible to solve the identified problem due to the combination of probabilistic modeling, machine learning methods, and risk-oriented test prioritization, which ensures the transition from static control to data-driven quality management. The results are explained by establishing relationships between software metrics and defect probability, enabling component ranking by risk level, formation of a risk register, and optimization of verification resource allocation. Experimental validation on the open NASA Metrics Data Program dataset demonstrated that the model achieves ROC-AUC = 0.669 and PR-AUC = 0.382, and in scenario analysis provides recall@top-k ≈ 0.38 compared to ≈ 0.37 for the LOC-based approach and ≈ 0.18 for random selection. Practical application of the results is advisable in software quality management systems, DevOps analytics, CI/CD environments, and test prioritization processes, provided that historical data on code metrics, testing outcomes, module characteristics, and defectiveness are available under resource constraints.

Key words: *risk-based quality management, SDLC, machine learning, defect prediction, DSS, software metrics, data-driven.*

1. Вступ. Сучасні процеси розробки програмного забезпечення характеризуються високою складністю архітектур, динамічністю вимог та інтеграцією практик безперервної доставки, що зумовлює зростання кількості ризиків, пов'язаних із якістю ІТ-продуктів на всіх етапах життєвого циклу. Традиційні підходи до забезпечення якості переважно базуються на процедурних регламентах, експертних оцінках і постфактум-контролі дефектів, що не дозволяє своєчасно ідентифікувати потенційно проблемні компоненти та оптимально розподіляти ресурси тестування. Водночас існуючі методи управління ризиками розглядають їх як окрему управлінську категорію, не пов'язану безпосередньо з кількісними метриками якості та даними SDLC, що ускладнює інтеграцію ризик-менеджменту в процеси забезпечення якості. За умов переходу до data-driven розробки та використання DevOps-практик

виникає потреба у формалізованих моделях, які дозволяють кількісно оцінювати ризики на основі фактичних даних програмних артефактів, прогнозувати їх вплив на показники якості та використовувати отримані оцінки для прийняття управлінських рішень у межах SDLC. Таким чином актуальною є наукова задача розроблення інтегрованого методу ризик-орієнтованого управління якістю IT-продукту, який поєднує математичне моделювання, машинне навчання та підтримку прийняття рішень.

У роботі [1] пропонується нейро-нечітка система оцінювання ризиків безпеки, орієнтована на етапи SDLC, де ризикові фактори перетворюються на керовані правила й нечіткі висновки для підтримки рішень у розробці. Методологічно такий підхід важливий тим, що формалізує нечіткі експертні судження (ймовірність/вплив/критичність) у вигляді моделі, яку можна навчати на даних і застосовувати як інструмент ранньої профілактики дефектів безпеки; однак типова обмеженість подібних систем полягає в залежності від якості початкових лінгвістичних змінних і правил, а також у складності переносимості між доменами та командами, коли «однакові» ризики на практиці мають різну семантику й наслідки для різних продуктів. Вимірювані результати в таких роботах зазвичай подаються як точність/узгодженість класифікації рівнів ризику або покращення коректності пріоритизації порівняно з базовими експертними шкалами; сильна сторона полягає в тому, що отримані оцінки можна інтегрувати в контрольні «quality gates» SDLC та пов'язувати з тестовими стратегіями, але слабким місцем лишається валідація на репрезентативних промислових наборах і довгострокова стабільність моделі під дрейфом вимог і загроз.

Огляд [2] систематизує техніки інтеграції безпеки в SDLC і показує, що значна частина підходів залишається фрагментованою: одні концентруються на практиках (на кшталт threat analysis, security testing), інші – на фреймворках і стандартах, при цьому авторам важливою видається кількісна оцінка ризиків у числовій шкалі для керованого розподілу ресурсів. Дослідження підкреслює, що «точність» оцінювання/пріоритизації ризиків і трудомісткості захисних активностей є системною проблемою, а перспективним шляхом названо використання AI/ML (зокрема глибокого навчання) та автоматизації, але з обов'язковою верифікацією таких рішень і застосуванням відомих чек-листів/еталонів для порівнюваності результатів. Це створює методологічний міст між ризик-менеджментом і якістю: без стандартизованих метрик та узгоджених протоколів оцінювання складно довести, що інтеграція безпеки реально покращує якість продукту, а не лише збільшує кількість «процедур» у процесі.

У доліджені [3] фокус зроблено на вимірюванні загроз і вразливостей у контексті secure SDLC, тобто на перетворенні загального «переліку ризиків» у вимірювані конструкції, які можна відстежувати під час життєвого циклу. Цінність таких робіт у тому, що вони наближають ризик-орієнтоване забезпечення якості до інженерної дисципліни: ризики перестають бути лише описовими, а стають об'єктом регулярного вимірювання (наприклад, через індикатори вразливостей, рівні експозиції, критичність компонентів, результати перевірок і тестів). Водночас у подібних підходів часто лишається невирішеним питання причинно-наслідкового зв'язку «метрика ризику → дефекти якості/відмови/інциденти»: показники вразливостей можуть зростати через кращу детекцію, а не через реальне погіршення якості, що потребує коректного дизайну експерименту й часових моделей, які відрізняють детекцію від фактичного ризику.

Автори роботи [4] розглядають дизайн, реалізацію й автоматизацію підходу до ризик-менеджменту для специфічного класу загроз («man-at-the-end»), що є прикладом практико-орієнтованого ризик-контролю з інженерною реалізацією. Значення цього напряму полягає в демонстрації того, як ризик-моделі можуть переходити у «вбудовані» механізми захисту та контрольні процедури, які реально виконуються в конвеєрі розробки/впровадження, а не існують як документація. Разом із тим, вузька спеціалізація під конкретну загрозу підсилює проблему узагальнення: підхід може бути ефективним у своєму класі, але без єдиної моделі зіставлення ризиків різної природи (безпека, надійність, продуктивність, відповідність вимогам) важко забезпечити саме управління якістю як системною властивістю продукту протягом SDLC.

Робота [5] демонструє пріоритизацію ризиків в agile-проєктах через Analytic Hierarchy Process, тобто через парні порівняння та ієрархічне зважування критеріїв. Вимірюваним результатом тут є отриманий ранжований перелік ризиків і узгоджені ваги факторів (а також, як правило, перевірка узгодженості експертних суджень), що цінно для організаційної керованості: команда отримує прозору логіку вибору «що гасити першим». Проте AHP методологічно залежить від експертності та стабільності суджень; при зміні складу команди, домену чи фази життєвого циклу ваги можуть «плисти», а сам підхід слабко використовує фактичні дані розробки (дефекти, тести, інциденти, метрики репозиторію), через що виникає розрив між пріоритизацією на папері та реальними драйверами якості, які видно в телеметрії SDLC.

У [6] дослідники запропонували модель включення «частки ризику» в оцінювання трудомісткості розробки, що зближує ризик-менеджмент і планування якості через прогнозування ресурсів та

очікуваної складності. Вимірювані результати в таких роботах зазвичай подаються через метрики точності оцінки effort (наприклад, похибка/відносна похибка/узгодженість прогнозів), і прикладна цінність полягає в тому, що ризик стає фактором, який впливає на план якості (скільки тестування, рев'ю, hardening потрібно закласти). Але залишається відкритим питання, чи «ризик-пропорція» є стабільною величиною в динамічних умовах SDLC: ризики можуть різко змінюватися через вимоги, інтеграції, залежності та зовнішні вектори атак, тому статичне вбудовування ризику в effort потребує адаптивних моделей, які оновлюються за фактичними даними виконання.

Автори [7] запропонували security testing framework, що розкладає практики безпекового тестування на всі фази процесу, підкреслюючи потребу визначати цілі, критерії оцінювання/KPI, формувати гайдлайни, генерувати звіти та забезпечувати фазу безперервного поліпшення. Важливо, що автори прямо вказують на ключове обмеження – відсутність експериментальної оцінки, і рекомендують майбутні експерименти для вимірювання ефективності та перевірки застосовності в конвеєрах (зокрема хмарних пайплайнах). Саме ця прогавина є типовою: багато фреймворків добре описують структуру процесу, але не дають вимірюваного ефекту на показники якості (defect leakage, security defect density, MTTR інцидентів, покриття тестів, вартість виправлень на пізніх фазах тощо), через що управління якістю ризик-орієнтованими практиками лишається частково декларативним.

У [8] дослідники виконали систематичний огляд і порівняння підходів «security by design» та «privacy by design», показуючи відмінності за метою (уникнення вразливостей/атак проти уникнення privacy-ризиків), засобами (організаційні й технічні заходи) та моментом у життєвому циклі, де акцент може бути раннім або наскрізним. Методологічно це важливо для ризик-орієнтованого управління якістю тим, що якість у сучасних продуктах включає не лише функціональність і надійність, а й безпеку та приватність як невід'ємні властивості, які мають різні механізми досягнення й різні критерії оцінювання. Водночас, навіть у систематичних оглядах часто не вистачає єдиної операціоналізації результатів: «by design» підходи описуються на концептуальному рівні, але слабше пов'язуються з конкретними, відтворюваними метриками впливу на якість у SDLC, що ускладнює побудову data-driven моделей ризику й доведення ефективності рішень у порівняльних експериментах.

У [9] систематичне мапування зосереджене на requirements engineering для регуляторної відповідності: автори отримали 6914 робіт (2017–2023) і звузили до 280 релевантних, класифікували виклики та практики, а також зафіксували низьку частку досліджень зі спільною участю інженерів і юристів (близько 13,6 %) та обмеженість робіт, що пов'язують RE з іншими процесними областями життєвого циклу (близько 20,7 %). Це дає чітко вимірюваний зріз незрілості поля: відповідність і пов'язані з нею ризики часто розглядаються ізольовано від решти SDLC, хоча саме міжпроцесні зв'язки (вимоги → архітектура → реалізація → тестування → експлуатація) визначають реальний ризик-профіль і якість продукту. Об'єктивною причиною такої прогалини є різна мова артефактів і стейкхолдерів (юридичні норми проти інженерних вимог), а суб'єктивною – організаційні бар'єри та відсутність інструментів, які перетворюють регуляторні вимоги на машинно-читані, перевірювані правила, що можна пов'язати з метриками якості та ризику в даних SDLC.

В [10] науковці аналізують вплив DevOps у площині IT Service Management на основі багато-кейсного підходу, тобто емпірично розглядають, як практики DevOps змінюють сервісні процеси, що напряму пов'язані з якістю в експлуатації (інциденти, зміни, відновлення, стабільність сервісу). Сильна сторона таких робіт можливість фіксувати вимірювані ефекти на операційних показниках (швидкість реакції, стабільність релізів, узгодженість змін, зниження ручних операцій за рахунок автоматизації), але типовою слабкістю є складність відокремити ефект DevOps від ефекту супутніх трансформацій (перебудова команд, зміна архітектури, модернізація моніторингу), через що причинність потребує або довших часових рядів, або квазіекспериментальних дизайнів. Для ризик-орієнтованого управління якістю важливо, що DevOps додає новий клас ризиків (швидкі зміни, dependency-ризики, конфігураційні помилки) і одночасно створює нові джерела даних (пайплайни, логи, моніторинг), які можуть бути основою для ML-оцінювання ризику та прогнозу якості в режимі near-real-time.

Узагальнюючи критично, у розглянутих джерелах чітко простежується два невирішені ядра. Перше – нестача наскрізної, кількісно верифікованої моделі, яка б поєднувала ризики різної природи (безпека, відповідність, процесні ризики agile/DevOps, ризики оцінки effort) з вимірюваними показниками якості на різних фазах SDLC: частина робіт дає моделі оцінювання (нейро-нечіткі, AHP, risk-пропорції), частина – фреймворки практик, частина – мапування/огляди, але часто без єдиного мосту до метрик якості та експериментальної доказовості в порівняльних постановках. Друге – дефіцит відтворюваної емпіричної валідації й стандартизованих протоколів оцінювання: навіть там, де пропонуються повні фреймворки, автори прямо вказують на брак експериментів і потребу виміряти ефективність; в оглядах наголошується на необхідності еталонів, чек-листів і валідації AI-інтегрованих методів.

Об'єктивно ці питання лишаються невирішеними через гетерогенність даних SDLC (різні трекери, різні практики, різні домени), складність доступу до промислових датасетів з інцидентами/вразливостями, а також через дрейф загроз і вимог, який застарює моделі. Суб'єктивними причинами виступають організаційні розриви між ролями (інженери, безпекарі, юристи, ITSM), фрагментоване впровадження практик «острівцями» та прагнення описувати процеси концептуально без затрат на строгий дизайн експерименту й доведення причинності.

Систематизація локальних проблем із кожного джерела зводиться до однієї узагальненої невирішеної проблеми: у сучасних SDLC бракує інтегрованого, даними підкріпленого та експериментально верифікованого механізму, який би дозволяв кількісно оцінювати, прогнозувати й пріоритизувати ризики якості (включно з безпекою, приватністю, відповідністю, процесними та ресурсними ризиками), пов'язуючи їх із метриками якості продукту та забезпечуючи адаптацію під зміну контексту розробки. Саме з цієї невирішеної проблеми логічно випливає мета дослідження як побудова ризик-орієнтованого управління якістю на основі аналізу даних і машинного навчання в межах SDLC, де ризик розглядається не як разова експертна оцінка, а як динамічна, вимірювана й керована величина протягом життєвого циклу.

Метою статті є розроблення та експериментальна валідація формалізованої математичної моделі ризик-орієнтованого управління якістю IT-продукту, інтегрованої в SDLC, що забезпечує кількісне оцінювання, прогнозування та пріоритизацію ризиків на основі аналізу даних і методів машинного навчання з подальшим використанням отриманих оцінок у системі підтримки прийняття рішень щодо оптимізації процесів забезпечення якості.

Для досягнення поставленої мети у дослідженні сформульовано такі задачі:

- розробити формалізовану математичну модель ризик-орієнтованого управління якістю, у якій ризик інтерпретується як імовірнісна характеристика дефектності програмних компонентів;
- обґрунтувати механізм інтеграції оцінок ризику в процеси забезпечення якості у вигляді адаптивних контрольних точок (quality gates) у межах SDLC;
- реалізувати програмний прототип системи підтримки прийняття рішень, що забезпечує ідентифікацію, оцінювання та пріоритизацію ризиків на основі даних;
- провести експериментальну валідацію запропонованої моделі на реальних даних та оцінити її ефективність у порівнянні з традиційними підходами до управління якістю.

2. Матеріали і методи. Об'єктом дослідження є процес управління якістю програмного забезпечення в умовах невизначеності та обмежених ресурсів, що реалізується в межах життєвого циклу розробки програмного забезпечення (SDLC).

Основною гіпотезою дослідження є припущення про те, що використання формалізованої ризик-орієнтованої моделі, яка базується на ймовірнісній оцінці дефектності програмних компонентів та застосуванні методів машинного навчання, дозволяє підвищити ефективність управління якістю за рахунок оптимізації процесів тестування та пріоритизації ресурсів.

У роботі прийнято такі припущення: вхідні дані про дефекти та характеристики програмних компонентів є репрезентативними та достатніми для побудови моделі; процеси виникнення дефектів можуть бути описані статистично; залежності між ознаками та ймовірністю дефектів є такими, що можуть бути апроксимовані сучасними методами машинного навчання.

У роботі використано такі спрощення: вплив зовнішніх організаційних факторів (людський фактор, зміни вимог у реальному часі) не враховується безпосередньо; модель розглядається в дискретному часовому представленні; оцінювання ефективності здійснюється на основі доступних історичних даних без урахування повної варіативності реальних проєктів.

У межах дослідження запропоновано формалізовану математичну модель ризик-орієнтованого управління якістю IT-продукту, інтегровану в SDLC, яка розглядає ризик як кількісну, динамічну та керовану величину, що оцінюється на основі даних програмних артефактів і використовується для прийняття управлінських рішень щодо забезпечення якості. Базовим об'єктом моделювання є множина програмних компонентів або змін $M = \{m_1, \ldots, m_n\}$, кожен з яких описується вектором ознак $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$, що включає структурні метрики коду, показники складності, історію змін та індикатори дефектності, інтерпретовані як фактори ризику. У запропонованій моделі ризик якості визначається як умовна ймовірність появи дефекту для компонента m_i за наявних значень ознак, тобто $R_i = P(y_i = 1 | x_i, \theta)$, де y_i – бінарна змінна дефектності, а θ – параметри моделі машинного навчання, що апроксимує нелінійну залежність між характеристиками програмного модуля та фактом виникнення дефекту. Таким чином ризик набуває імовірнісної інтерпретації та може використовуватись як інтегральний показник якості, що узгоджується з концепцією проактивного управління якістю у SDLC.

На відміну від традиційних експертних або статичних моделей, у запропонованому підході ризик формується на основі емпіричних даних життєвого циклу та оновлюється ітеративно в процесі

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

розробки, що забезпечує його адаптивність до змін вимог, архітектури та середовища виконання. Інтегральний індекс ризику для підмножини компонентів $S \subseteq M$ визначається як агрегована величина

$$R(S) = \frac{1}{S} \sum_{m_i \in S} R_i, \tag{1}$$

що дозволяє оцінювати ризик на рівні підсистем, релізів або фаз SDLC та використовувати його як критерій прийняття рішень щодо розподілу ресурсів тестування та верифікації. Для забезпечення керованості процесу введено порогову функцію якості Q_g, що задає умови проходження контрольної точки: компонент вважається таким, що потребує посиленого контролю, якщо $R_i > \tau$, де τ – адаптивний поріг, визначений на основі статистичних характеристик навчальної вибірки або цільових показників якості. Така постановка дозволяє формалізувати механізм quality gates у SDLC як функцію від прогнозованого ризику, що є відмінністю від практик, де контрольні точки визначаються виключно часовими або процедурними критеріями.

Для оцінювання ефективності управлінських стратегій у моделі введено функціонал очікуваної якості релізу

$$Q = 1 - \frac{1}{M} \sum_{i=1}^{n} w_i R_i, \tag{2}$$

де w_i – ваговий коефіцієнт критичності компонента, що відображає його вплив на системні властивості, такі як надійність або безпека. Цей показник інтерпретується як нормована міра очікуваної бездефектності системи та використовується для порівняння альтернативних сценаріїв управління якістю, зокрема ризик-орієнтованого та традиційного розподілу тестового покриття. Таким чином модель забезпечує кількісну основу для прийняття рішень, що узгоджує оцінювання ризику з метриками якості продукту.

Наукова новизна запропонованої моделі полягає у поєднанні імовірнісної оцінки ризику, отриманої з використанням методів машинного навчання, з формалізованим механізмом управління якістю у вигляді DSS, інтегрованого в SDLC, що забезпечує перехід від статичного експертного оцінювання до безперервного data-driven моніторингу та пріоритизації ризиків. Запропоновано математичне представлення quality gates як функції прогнозованого ризику та введено агрегований індекс якості, який дозволяє кількісно порівнювати сценарії управління та оцінювати ефект від концентрації ресурсів на високоризикових компонентах. На відміну від існуючих підходів, де ризик і якість розглядаються окремо, у даній моделі ризик виступає керованою змінною, безпосередньо пов'язаною з метриками якості та використовується як параметр оптимізації процесу забезпечення якості в межах життєвого циклу розробки програмного забезпечення.

3. Результати і обговорення

3.1. Розробка математичної моделі ризик-орієнтованого управління якістю. У межах дослідження запропоновано формалізовану математичну модель ризик-орієнтованого управління якістю програмного забезпечення, у якій ризик інтерпретується як ймовірність виникнення дефектів у програмних компонентах. На відміну від традиційних підходів, де ризик розглядається переважно на якісному рівні, запропонована модель забезпечує кількісне оцінювання ризику на основі даних.

Формально ризик для і-го компонента визначається як:

$$R_i = P(D_i = 1 \mid X_i), \tag{3}$$

де X_i – вектор ознак, що характеризує компонент (метрики коду, історія змін, результати тестування тощо), а D_i – бінарна змінна, що відображає наявність дефекту.

Оцінювання ймовірності здійснюється з використанням методів машинного навчання, що дозволяє враховувати складні нелінійні залежності між ознаками та дефектністю. Це забезпечує вищу точність у порівнянні з класичними евристичними або експертними підходами.

На відміну від підходів, у яких процес розробки описується на концептуальному рівні без чіткої математичної інтерпретації параметрів управління, запропоноване рішення дозволяє здійснювати кількісне оцінювання впливу ризиків на якість програмних компонентів завдяки формалізації взаємозв'язків між метриками та ймовірністю дефектів [12]. Саме ця особливість пояснює отриманий ефект підвищення керованості процесу якості, оскільки модель забезпечує обґрунтоване ранжування компонентів за рівнем ризику. У порівнянні з існуючими підходами, де ризик розглядається як допоміжна характеристика, запропонована модель інтегрує його безпосередньо у механізм прийняття рішень, що дозволяє частково усунути проблему розриву між ризик-менеджментом і якістю, визначену у вступі. Разом з тим, обмеженням є залежність точності моделі від якості вхідних даних і припущення

про стаціонарність розподілу дефектів, що може знижувати ефективність у динамічних середовищах. Подальший розвиток доцільно пов'язати з урахуванням часової динаміки ризиків і адаптивним оновленням параметрів моделі.

3.2. Інтеграція оцінок ризику в процеси забезпечення якості. На основі запропонованої моделі розроблено механізм інтеграції оцінок ризику у процеси забезпечення якості програмного забезпечення у вигляді адаптивних контрольних точок (quality gates) в межах SDLC.

На відміну від традиційних підходів, де контроль якості здійснюється за фіксованими правилами, запропонований механізм передбачає адаптивне управління на основі рівня ризику. Зокрема, для кожного етапу життєвого циклу визначається порогове значення ризику R^*, перевищення якого ініціює додаткові перевірки або тестування.

Інтеграція ризик-оцінок реалізується через:

- пріоритизацію тестових сценаріїв відповідно до значень R_i;
- динамічний розподіл ресурсів тестування;
- автоматизоване прийняття рішень щодо переходу між етапами SDLC.

Концентрація ресурсів на найбільш ризикованих компонентах дозволяє зменшити загальний рівень дефектності системи при тих самих витратах. Це стає можливим завдяки використанню кількісних оцінок ризику як керуючого параметра.

На відміну від традиційного статичного контролю якості, у якому перевірки виконуються незалежно від контексту та стану системи, запропонований підхід дозволяє динамічно блокувати або пропускати артефакти залежно від їх критичності, що відповідає концепції quality gates у безперервних конвеєрах розробки [13]. Це стає можливим завдяки інтеграції механізму оцінювання ризику з pipeline-підходом, де кожен етап завершується перевіркою умов якості. У результаті досягається зменшення накопичення дефектів на пізніх стадіях і більш ефективний розподіл ресурсів тестування, що безпосередньо закриває проблему неефективного контролю якості, визначену у вступі. Водночас обмеженням є необхідність налаштування порогових значень ризику, які можуть залежати від конкретного проєкту або домену. Недоліком також є потенційне збільшення часу виконання pipeline при жорстких умовах контролю. Подальші дослідження доцільно спрямувати на автоматичну адаптацію порогів quality gates на основі історичних даних.

3.3. Реалізація програмного прототипу системи підтримки прийняття рішень. Із метою експериментальної перевірки та валідації запропонованої математичної моделі ризик-орієнтованого управління якістю ІТ-продукту було розроблено програмний прототип системи підтримки прийняття рішень, що реалізує механізм кількісного оцінювання ризиків на основі аналізу даних життєвого циклу розробки програмного забезпечення. Програмна реалізація виконана мовою Python у середовищі VS Code та побудована відповідно до структурних елементів концептуальної моделі третього розділу дисертації, зокрема алгоритму ідентифікації, аналізу, оцінювання та пріоритизації ризиків, інтегрованого в процеси SDLC.

Архітектурно система складається з чотирьох взаємопов'язаних рівнів: рівня даних, аналітичного рівня, рівня моделювання ризику та рівня підтримки управлінських рішень (рис. 1).

На рівні даних здійснюється завантаження, валідація та попередня обробка історичних метрик програмних модулів, що відображають структурні, процесні та дефектні характеристики, які інтерпретуються як індикатори потенційного ризику якості. Передбачено автоматичне перетворення вихідних форматів у внутрішнє табличне представлення, нормалізацію ознак, обробку пропущених значень і формування вектора ознак, що відповідає формалізованій системі факторів ризику.

Аналітичний рівень реалізує формалізацію ризику як функції від набору вимірюваних параметрів, де кожному програмному компоненту або зміні ставиться у відповідність ймовірнісна оцінка виникнення дефекту, що інтерпретується як інтегральний показник ризику якості. Для цього використано методи машинного навчання, які дозволяють апроксимувати нелінійні залежності між характеристиками програмних артефактів та фактичними проявами дефектності. У межах системи передбачено можливість використання різних алгоритмів класифікації як альтернативних моделей оцінювання ризику, що забезпечує порівняльний аналіз та підвищує обґрунтованість управлінських рішень.

Рівень моделювання ризику реалізує процедуру ранжування програмних компонентів за ступенем критичності, формування реєстру ризиків та визначення пріоритетності заходів контролю якості. Результатом є впорядкований список об'єктів розробки з відповідними числовими значеннями ризику, що дозволяє інтерпретувати отримані оцінки у вигляді рекомендацій щодо спрямування ресурсів тестування, рев'ю та верифікації. Таким чином забезпечується перехід від реактивної моделі контролю дефектів до проактивного управління якістю на основі прогнозованих ризиків.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

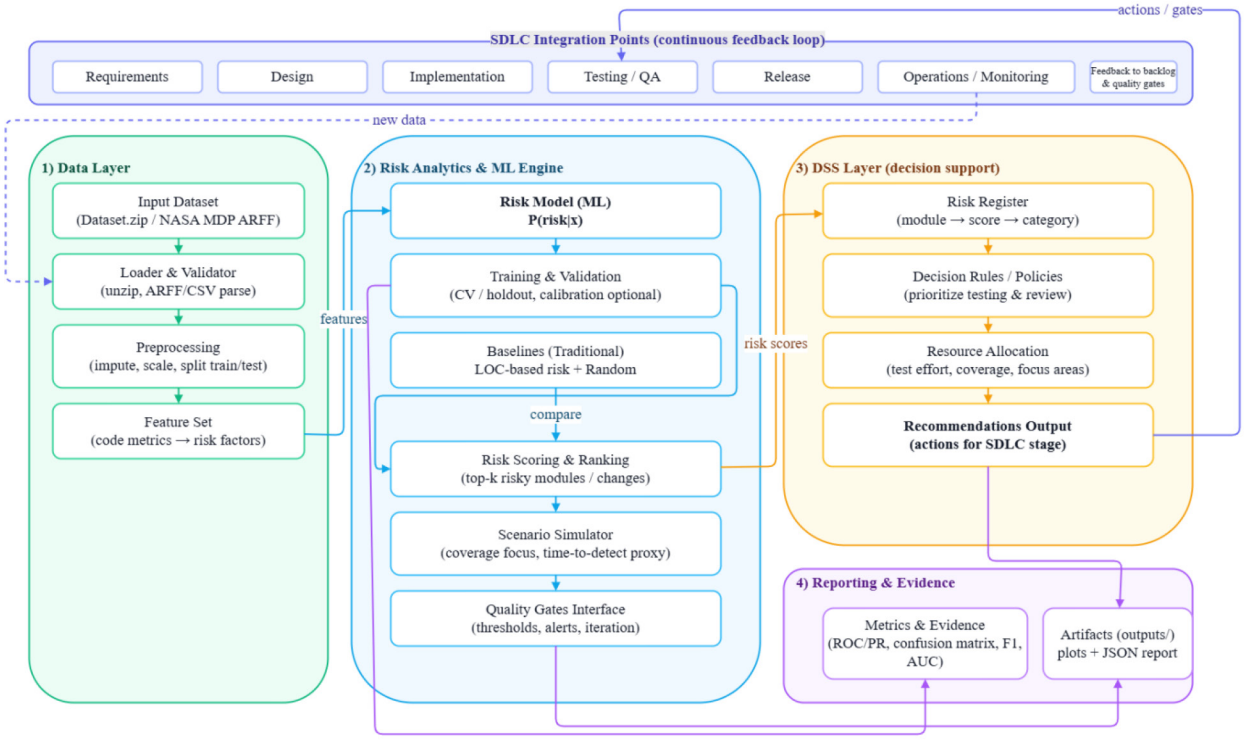

Рис. 1. Архітектура програмного прототипу системи підтримки прийняття рішень (побудовано автором)

Рівень підтримки прийняття рішень реалізує механізм сценарного моделювання, який дозволяє оцінити вплив різних стратегій управління якістю (зокрема, ризик-орієнтованого тестування, перерозподілу покриття або фокусування на критичних модулях) на очікувані показники якості. У системі передбачено формування аналітичних звітів, метрик точності прогнозування ризику та візуалізацій, що відображають структуру ризиків, значущість факторів та результати порівняльного аналізу альтернативних підходів. Генеровані графічні та числові показники забезпечують можливість інтеграції результатів у процеси моніторингу SDLC та використовуються як інструмент кількісного обґрунтування управлінських рішень.

Принцип дії програмного прототипу полягає у послідовному проходженні етапів: формування вхідного набору даних, навчання моделі оцінювання ризику, прогнозування рівня ризику для кожного програмного компонента, агрегування результатів у реєстр ризиків та генерації рекомендацій щодо пріоритизації заходів забезпечення якості. Така логіка відповідає концепції безперервного моніторингу ризиків у межах SDLC, де оцінювання ризику виконується ітеративно на основі нових даних, що надходять у процесі розробки та тестування.

Розроблений програмний засіб реалізує формалізовану математичну модель ризик-орієнтованого управління якістю у вигляді DSS-прототипу, який забезпечує автоматизовану ідентифікацію, кількісне оцінювання та пріоритизацію ризиків якості програмного забезпечення на основі даних SDLC, що створює підґрунтя для подальшої експериментальної апробації та аналізу ефективності запропонованого методу.

У якості емпіричної бази для навчання та валідації моделі використано відкритий набір даних NASA Metrics Data Program, що містить метрики програмних модулів, отримані з реальних проєктів розробки програмного забезпечення та широко застосовується в задачах прогнозування дефектності та оцінювання ризиків якості. Набір включає структурні характеристики коду (зокрема показники складності, обсягу, глибини вкладеності, кількості операторів та умовних переходів), які інтерпретуються як фактори ризику виникнення дефектів, а також цільову змінну, що відображає факт наявності дефекту в модулі. Таке представлення дозволяє формувати вектор ознак, який безпосередньо відповідає формалізованій системі індикаторів ризику та забезпечує можливість навчання імовірнісної моделі $P(y = 1 \mid x)$ на основі історичних даних. У межах програмної реалізації здійснюється автоматичне завантаження, валідація та попередня обробка даних, включаючи обробку пропущених значень, нормалізацію показників та поділ вибірки на навчальну і тестову підмножини, що забезпечує

коректність статистичного оцінювання. Використання саме цього набору даних обумовлено його репрезентативністю для задач прогнозування дефектів, наявністю стандартизованих метрик програмних модулів та можливістю порівняння отриманих результатів із відомими дослідженнями у сфері ризик-орієнтованого забезпечення якості програмного забезпечення [11].

На відміну від монолітних реалізацій, де всі функції інтегровані в єдину систему, запропонований підхід дозволяє незалежно модифікувати окремі компоненти, що відповідає сучасним принципам побудови експертних систем [14]. Це стає можливим завдяки використанню сервісно-орієнтованої архітектури та відокремлення механізму логічного виведення від сховища даних. У результаті система забезпечує більш ефективну ідентифікацію та пріоритизацію ризиків, що дозволяє зменшити інформаційне перевантаження при прийнятті рішень і підвищити точність управління якістю. Запропонований прототип закриває проблему відсутності інструментальної підтримки ризик-орієнтованого управління. Обмеженням є залежність ефективності системи від якості навчання моделі та обмежений обсяг використаних даних. Недоліком також є відсутність інтеграції з реальними CI/CD середовищами. Подальший розвиток пов'язаний із розширенням функціональності системи та інтеграцією з DevOps-інфраструктурою.

3.4. Експериментальна валідація моделі та аналіз ефективності. У ході експериментальної апробації запропонованої математичної моделі було виконано навчання моделі прогнозування ризику дефектності програмних модулів та проведено її порівняння з традиційними підходами до пріоритизації контролю якості. Отримані результати свідчать про здатність моделі адекватно розрізняти модулі з підвищеною ймовірністю дефектів та формувати обґрунтований реєстр ризиків, що підтверджується значенням площі під ROC-кривою *AUC* = 0,67, яке перевищує випадкову класифікацію та демонструє наявність стійкого прогностичного сигналу (рис. 2).

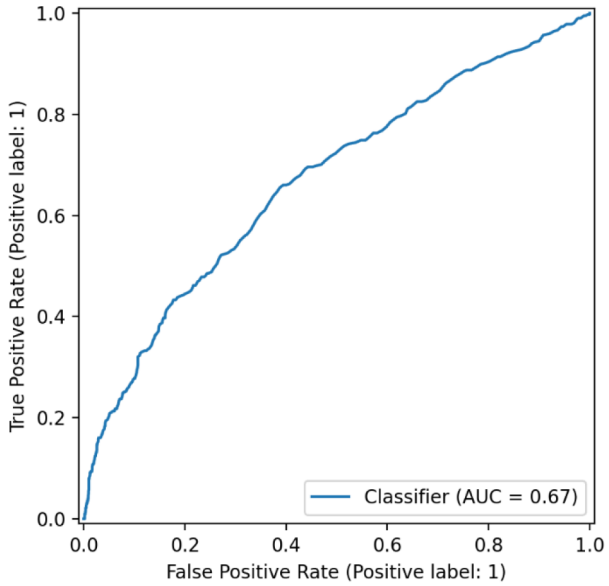

Рис. 2. ROC-крива моделі прогнозування ризику дефектності (побудовано авторським програмним забезпеченням)

З метою підтвердження конкурентоспроможності запропонованого підходу виконано порівняння з базовими моделями прогнозування дефектів, що широко застосовуються у сучасних дослідженнях (Logistic Regression, Random Forest, Linear SVM, Gaussian Naive Bayes та HistGradientBoosting). Оцінювання здійснювалося на однаковому розбитті навчальної та тестової вибірок із використанням метрик ROC-AUC, PR-AUC, F1, Precision, Recall та Accuracy. Отримані результати наведено в табл. 1. Порівняння показує, що запропонована модель забезпечує не нижчий рівень прогностичної здатності порівняно з базовими підходами, зберігаючи при цьому інтерпретованість результатів і можливість їх безпосереднього використання в DSS-контурі управління якістю та формуванні адаптивних quality gates. Збіг метрик із Random Forest пояснюється використанням його як базового класифікатора, тоді як новизна запропонованої моделі полягає в інтеграції моделі в DSS-контур SDLC.

Аналіз матриці помилок показує, що модель коректно ідентифікує більшість бездефектних модулів (1463 істинно негативних результатів), водночас виявляючи значну частку дефектних компонентів (77 істинно позитивних), що є критично важливим для задачі ризик-орієнтованого управління якістю, де

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Таблиця 1

Порівняння запропонованої моделі з базовими SOTA-моделями прогнозування дефектів

Модель	roc_auc	pr_auc	f1	precision	recall	accuracy
LogisticRegression	0.659819	0.382651	0.403425	0.334913	0.507177	0.677801
RandomForest	0.668988	0.381586	0.275	0.542254	0.184211	0.791367
LinearSVM	0.65631	0.379322	0.201161	0.525253	0.124402	0.78777
GaussianNB	0.623592	0.348285	0.257391	0.471338	0.177033	0.780576
HistGradientBoosting	0.664983	0.381317	0.230216	0.463768	0.15311	0.780062
Запропонована модель	0.668988	0.381586	0.275000	0.542254	0.184211	0.791367

пріоритетом є виявлення потенційно проблемних елементів на ранніх етапах SDLC (рис. 3). Наявність помилок другого роду (341 пропущений дефект) пояснюється дисбалансом класів у вихідному наборі даних та складністю апроксимації нелінійних залежностей між метриками коду та фактичними дефектами, однак навіть за таких умов модель забезпечує кращу селективність порівняно з базовими стратегіями.

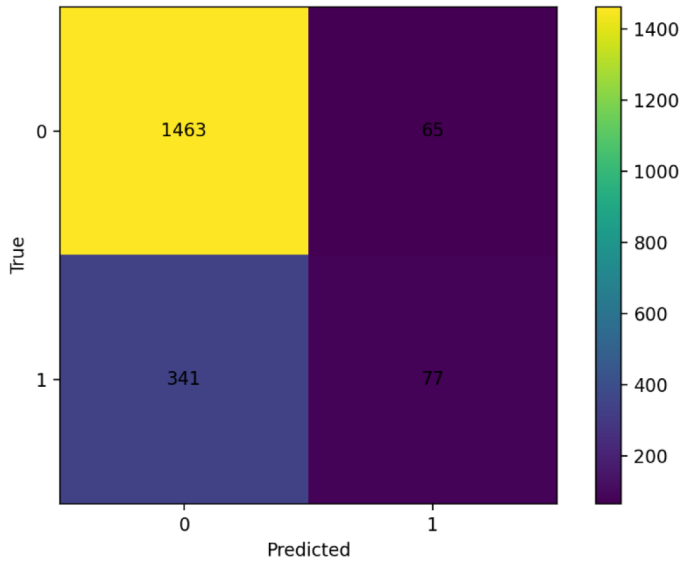

Рис. 3. Матриця помилок класифікації дефектності програмних модулів (побудовано авторським програмним забезпеченням)

Крива Precision–Recall демонструє середнє значення $AP = 0{,}38$, що є типовим для задач прогнозування дефектів із суттєвим дисбалансом класів і свідчить про здатність моделі концентрувати високий рівень точності в області малих значень recall, тобто ефективно відбирати найбільш критичні модулі для пріоритетного тестування (рис. 4). Саме така поведінка є бажаною в контексті обмежених ресурсів забезпечення якості, коли необхідно мінімізувати кількість перевірок при максимальному виявленні дефектів.

Результати ранжування модулів за прогнозованою ймовірністю ризику підтверджують можливість формування впорядкованого реєстру ризиків, що використовується DSS для пріоритизації заходів контролю якості (рис. 5). Спостерігається характерний спад імовірності ризику зі збільшенням рангу, що дозволяє виділити підмножину високоризикових компонентів і спрямувати на них основні ресурси тестування, реалізуючи принцип ризик-орієнтованого покриття.

Порівняльне моделювання сценаріїв управління якістю показало, що застосування запропонованої моделі забезпечує вищий рівень виявлення дефектів у підмножині перевірених модулів (recall @ top-k ≈ 0,38) порівняно з традиційним підходом, заснованим на обсязі коду (≈ 0,37), та суттєво перевищує випадкову стратегію (≈ 0,18), що підтверджує ефективність ризик-орієнтованої пріоритизації (рис. 6). Одночасно спостерігається зменшення медіанного часу до виявлення дефекту при використанні моделі ризику порівняно з LOC-орієнтованою стратегією, що свідчить про прискорення процесу верифікації та підвищення оперативності реагування на потенційні проблеми якості.

Результати SOTA-порівняння підтверджують, що отриманий ефект зумовлений не лише використанням конкретного алгоритму машинного навчання, а насамперед інтеграцією імовірнісної оцінки

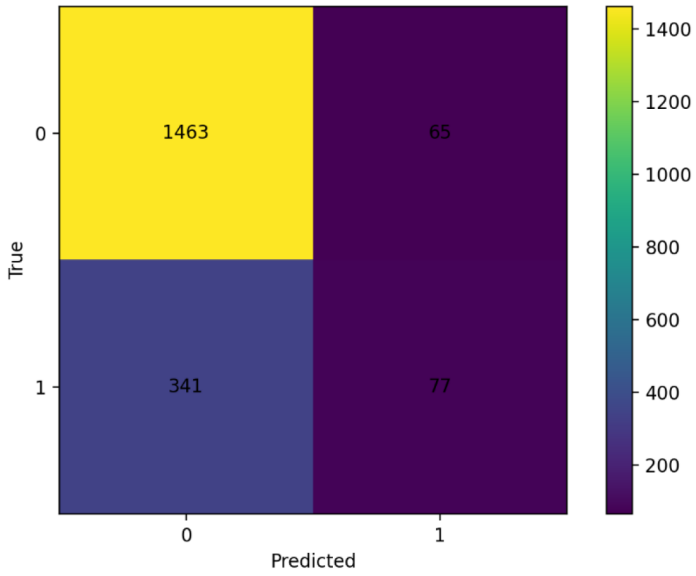

Рис. 4. Крива Precision-Recall моделі оцінювання ризику (побудовано авторським програмним забезпеченням)

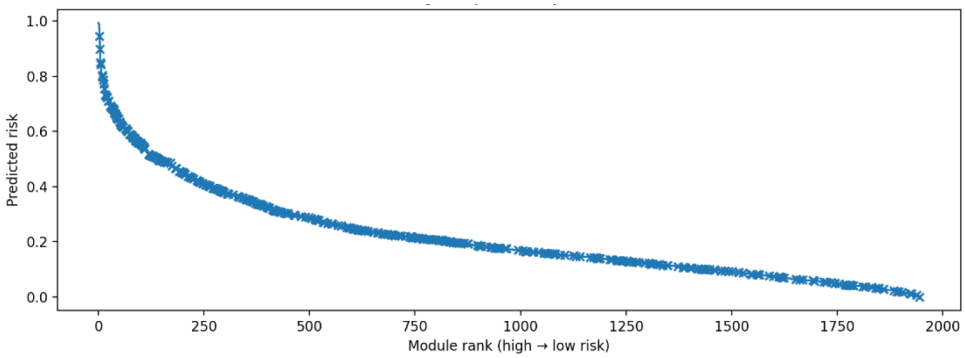

Рис. 5. Ранжування програмних модулів за прогнозованою ймовірністю ризику (побудовано авторським програмним забезпеченням)

ризику в контур управління якістю. На відміну від базових моделей, які виконують лише класифікацію дефектності, запропонований підхід забезпечує формування реєстру ризиків, підтримку сценарного аналізу та реалізацію адаптивних quality gates, що дозволяє перейти від задачі прогнозування до задачі управління якістю в межах SDLC.

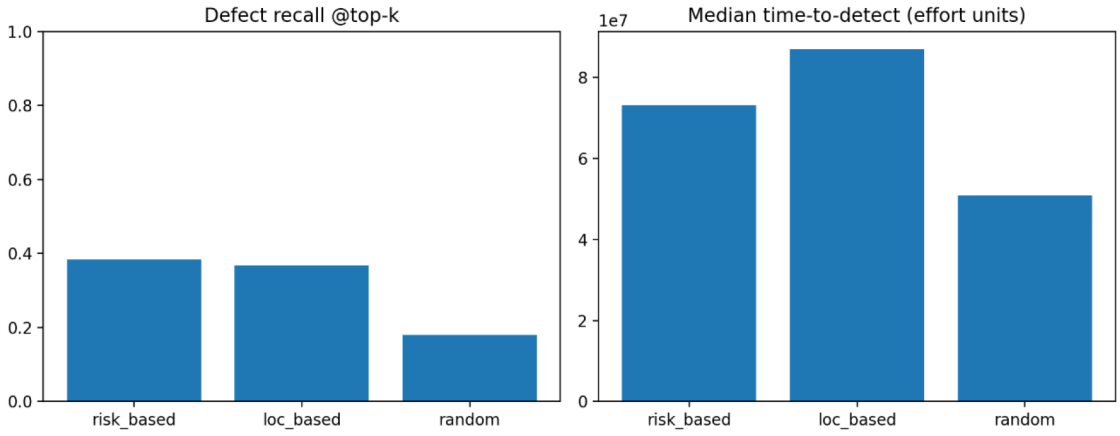

Рис. 6. Порівняння сценаріїв управління якістю: recall @ top-k та медіанний time-to-detect (побудовано авторським програмним забезпеченням)

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Аналіз важливості ознак показав, що найбільший внесок у прогноз ризику мають метрики обсягу коду та показники складності Холстеда, зокрема HALSTEAD_VOLUME, HALSTEAD_CONTENT, HALSTEAD_EFFORT, а також цикломатична та проектна складність (рис. 7). Це узгоджується з теоретичними положеннями про вплив структурної складності на ймовірність дефектів і підтверджує коректність формування вектора факторів ризику в запропонованій моделі.

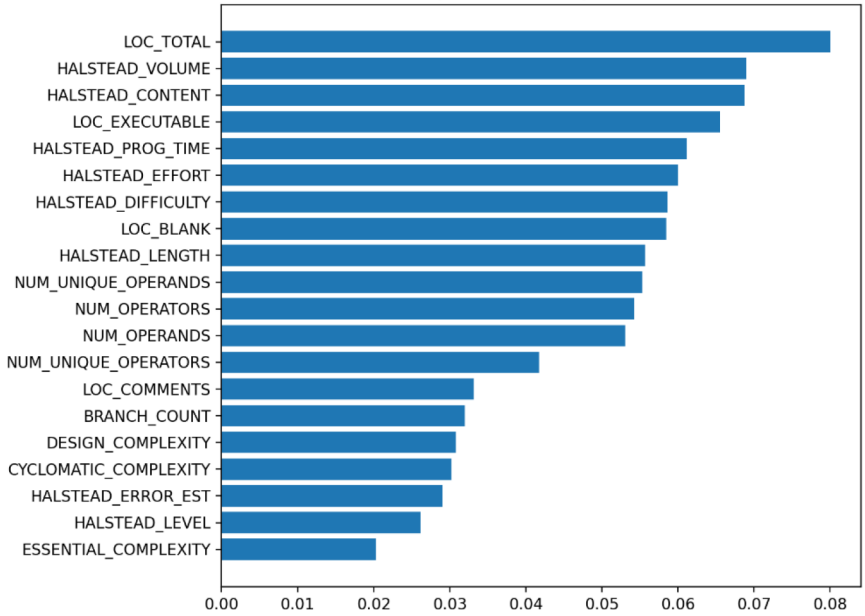

Рис. 7. Важливість ознак у моделі прогнозування ризику дефектності (побудовано авторським програмним забезпеченням)

Отримані результати підтверджують досягнення поставленої мети дослідження, а саме побудову формалізованої моделі ризик-орієнтованого управління якістю, інтегрованої в SDLC та реалізованої у вигляді DSS-прототипу, що забезпечує кількісне оцінювання ризиків, їх ранжування та підтримку управлінських рішень щодо пріоритизації тестування. Модель демонструє здатність підвищувати ефективність використання ресурсів контролю якості, скорочувати час до виявлення дефектів та формувати обґрунтований реєстр ризиків на основі даних, що свідчить про її перевагу над традиційними статичними підходами. Таким чином запропонований метод є оптимальним з точки зору поєднання імовірнісного прогнозування ризику, адаптивності до даних SDLC та можливості інтеграції у процеси безперервного моніторингу якості програмного забезпечення.

На відміну від підходів, які оцінюють якість моделей лише за класичними метриками точності, запропонований підхід враховує практичну цінність ранжування дефектних компонентів, що відповідає концепції effort-aware метрик [15]. Це дозволяє пояснити отриманий результат збільшення частки виявлених дефектів у перевіреній підмножині компонентів, оскільки система орієнтується не лише на правильність класифікації, а й на ефективність використання ресурсів. У порівнянні з традиційними підходами тестування, де перевірка виконується рівномірно або за експертною оцінкою, запропонований метод забезпечує більш раціональний розподіл зусиль, що безпосередньо закриває проблему неефективної пріоритизації. Обмеженням дослідження є використання одного набору даних, що може обмежувати узагальнення результатів. Недоліком також є відсутність аналізу впливу різних типів моделей машинного навчання. Подальші дослідження доцільно спрямувати на розширення експериментальної бази та порівняння різних алгоритмів прогнозування.

4. Висновки. У результаті проведеного дослідження було отримано формалізовану математичну модель ризик-орієнтованого управління якістю програмних систем, у межах якої ризик інтерпретується як імовірнісна характеристика дефектності компонентів, що дозволило перейти від описових підходів до кількісного представлення процесів забезпечення якості. Особливістю запропонованої моделі є інтеграція ризику безпосередньо у механізм прийняття рішень, що забезпечує можливість ранжування компонентів за рівнем потенційної дефектності та підвищує керованість процесу розробки. Отриманий результат пояснюється встановленням аналітичних залежностей між метриками програмного забезпечення та ймовірністю виникнення дефектів, що дозволяє здійснювати обґрунтоване прогнозування та мінімізувати невизначеність при оцінюванні якості.

Інтеграція оцінок ризику в процеси забезпечення якості реалізована у вигляді адаптивних контрольних точок, що забезпечують динамічне прийняття рішень щодо переходу між етапами життєвого циклу програмного забезпечення. На відміну від традиційних підходів, у яких контроль якості має статичний характер і не враховує поточний стан системи, запропонований механізм дозволяє блокувати або пропускати артефакти залежно від їх рівня ризику, що знижує накопичення дефектів на пізніх стадіях розробки. Такий результат пояснюється використанням кількісних критеріїв прийняття рішень, які враховують прогнозовану дефектність компонентів, що забезпечує більш ефективний розподіл ресурсів тестування та підвищує результативність процесів забезпечення якості.

Розроблений програмний прототип системи підтримки прийняття рішень забезпечує ідентифікацію, оцінювання та пріоритизацію ризиків на основі даних, що дозволяє автоматизувати процеси управління якістю та зменшити залежність від експертних оцінок. Відмінною рисою реалізованого рішення є використання модульної архітектури з розділенням рівнів даних, аналітики та прийняття рішень, що забезпечує гнучкість і масштабованість системи. Отриманий результат пояснюється використанням алгоритмів машинного навчання для виявлення прихованих закономірностей у даних програмних метрик, що підвищує точність оцінювання ризиків та дозволяє формувати обґрунтовані управлінські рішення.

Експериментальна валідація запропонованої моделі показала її ефективність у порівнянні з традиційними підходами до управління якістю, зокрема у частині підвищення точності виявлення дефектних компонентів та ефективності їх пріоритизації. Встановлено, що використання ризик-орієнтованого підходу дозволяє зосередити ресурси тестування на найбільш критичних компонентах, що призводить до зменшення витрат і підвищення якості програмного продукту. Отримані результати пояснюються здатністю моделі враховувати як статистичні характеристики даних, так і практичну значущість ранжування компонентів, що забезпечує більш ефективне використання ресурсів у процесі тестування. Разом з тим, встановлено, що ефективність запропонованого підходу залежить від якості вхідних даних та обсягу навчальної вибірки, що визначає межі його застосування. Подальший розвиток дослідження доцільно пов'язати з розширенням експериментальної бази, інтеграцією моделі у реальні DevOps-процеси та удосконаленням методів адаптації параметрів у динамічних умовах.

Конфлікт інтересів. Автори декларують, що не мають конфлікту інтересів стосовно даного дослідження, в тому числі фінансового, особистісного характеру, авторства чи іншого характеру, що міг би вплинути на дослідження тайого результати, представлені в даній статті.

Фінансування. Дослідження проводилося без фінансової підтримки.

Доступність даних. Рукопис має пов'язані дані у відкритому сховищі даних.

Використання засобів штучного інтелекту. Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Внесок авторів. Юрій Кіш: концептуалізація, методологія, програмна реалізація, формальний аналіз, валідація, візуалізація, написання – оригінальний проєкт; Ігор Лях: наукове керівництво, верифікація результатів, редагування, загальне управління дослідженням.

References:

1. Olusanya, O. O., Jimoh, R. G., Misra, S., & Awotunde, J. B. (2024). A neuro-fuzzy security risk assessment system for software development life cycle. Heliyon, 10(13), e33495. https://doi.org/10.1016/j.heliyon.2024.e33495

2. Saeed, H., Shafi, I., Ahmad, J., Khan, A. A., Khurshaid, T., & Ashraf, I. (2025). Review of Techniques for Integrating Security in Software Development Lifecycle. Computers, Materials & Continua, 82(1), 139–172. https://doi.org/10.32604/cmc.2024.057587

3. Humayun, M., Jhanjhi, N., Fahhad Almufareh, M., & Ibrahim Khalil, M. (2022). Security Threat and Vulnerability Assessment and Measurement in Secure Software Development. Computers, Materials & Continua, 71(3), 5039–5059. https://doi.org/10.32604/cmc.2022.019289

4. Basile, C., De Sutter, B., Canavese, D., Regano, L., & Coppens, B. (2023). Design, implementation, and automation of a risk management approach for man-at-the-End software protection. Computers & Security, 132, 103321. https://doi.org/10.1016/j.cose.2023.103321

5. M, A. Z., & J, C. (2024). Prioritization of Risks in Agile Software Projects Through an Analytic Hierarchy Process Approach. Procedia Computer Science, 233, 713–722. https://doi.org/10.1016/j.procs.2024.03.260

6. Dewi, R. S., & Dharmawan, Y. S. (2024). A Proposed Model for Embedding Risk Proportion in Software Development Effort Estimation. Procedia Computer Science, 234, 1777–1784. https://doi.org/10.1016/j.procs.2024.03.185

7. Mothanna, Y., ElMedany, W., Hammad, M., Ksantini, R., & Sharif, M. S. (2024). Adopting security practices in software development process: Security testing framework for sustainable smart cities. Computers & Security, 144, 103985. https://doi.org/10.1016/j.cose.2024.103985

8. Del-Real, C., De Busser, E., & van den Berg, B. (2024). Shielding software systems: A comparison of security by design and privacy by design based on a systematic literature review. Computer Law & Security Review, 52, 105933. https://doi.org/10.1016/j.clsr.2023.105933

9. Kosenkov, O., Elahidoost, P., Gorschek, T., Fischbach, J., Mendez, D., Unterkalmsteiner, M., Fucci, D., & Mohanani, R. (2025). Systematic mapping study on requirements engineering for regulatory compliance of software systems. Information and Software Technology, 178, 107622. https://doi.org/10.1016/j.infsof.2024.107622

10. Faustino, J., Pereira, R., Mira da Silva, M., Adriano, D., & Camargo, V. (2025). The Impact of DevOps in IT Service Management. Journal of Global Information Management, 33(1), 1–49. https://doi.org/10.4018/jgim.392902

11. Software defect prediction nasa. (б. д.). Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/datasets/aczy156/software-defect-prediction-nasa

12. Semenov, S., Tsukur, V., Molokanova, V., Muchacki, M., Litawa, G., Mozhaiev, M., & Petrovska, I. (2025). Mathematical Model of the Software Development Process with Hybrid Management Elements. Applied Sciences, 15(21), 11667. https://doi.org/10.3390/app152111667

13. Sabau, A.R., Hacks, S. & Steffens, A. Implementation of a continuous delivery pipeline for enterprise architecture model evolution. Softw Syst Model 20, 117–145 (2021). https://doi.org/10.1007/s10270-020-00828-z

14. Hnatushenko, V. V., Hnatushenko, Vik. V., Dorosh, N. L., Solodka, N. O., & Liashenko, O. A. (2022). Non-relational approach to developing knowledge bases of expert system prototype. Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu, (2), 112–117. https://doi.org/10.33271/nvngu/2022-2/112

15. Çarka, J., Esposito, M. & Falessi, D. On effort-aware metrics for defect prediction. Empir Software Eng 27, 152 (2022). https://doi.org/10.1007/s10664-022-10186-7

Відомості про авторів

Англ.	Укр.
Kish Yurii Postgraduate Student Department of Information Control Systems and Technologies Uzhhorod National University Universytets'ka St, 14, Uzhhorod, Zakarpattia Oblast, 88000 yurii_kish@protonmail.com ORCID: 0009-0000-6167-0129	Кіш Юрій аспірант Кафедра інформаційних управляючих систем та технологій Ужгородський національний університет 88000, Україна, Закарпатська обл., м. Ужгород, пл. Народна, 3 yurii_kish@protonmail.com ORCID: 0009-0000-6167-0129
Liakh Ihor Doctor of Technical Sciences, Professor Department of Informatics and Physical and Mathematical Disciplines Uzhhorod National University Universytets'ka St, 14, Uzhhorod, Zakarpattia Oblast, 88000 igor.lyah@uzhnu.edu.ua ORCID: 0000-0001-5417-9403	Лях Ігор доктор технічних наук, професор Кафедра інформатики та фізико-математичних дисциплін Ужгородський національний університет 88000, Україна, Закарпатська обл., м. Ужгород, пл. Народна, 3 igor.lyah@uzhnu.edu.ua ORCID: 0000-0001-5417-9403

Дата надходження статті: 20.03.2026
Дата надходження виправленої версії статті: 03.04.2026
Дата прийняття статті: 17.04.2026
Дата публікації статті: 01.06.2026