DOI: https://doi.org/10.33216/1998-7927-2025-293-7-17-30

УДК 004.42:005.334

АВТОМАТИЗОВАНА ПОБУДОВА РЕЄСТРУ РИЗИКІВ У РОЗРОБЦІ ПЗ НА ОСНОВІ ISSUE-ТРЕКІНГУ В GITHUB PROJECTS

Кіш Ю.В., Лях І.М.

AUTOMATED CONSTRUCTION OF A RISK REGISTER IN SOFTWARE DEVELOPMENT BASED ON ISSUE TRACKING IN GITHUB PROJECTS

Kish Y.V., Liakh I.M.

У статті розглянуто актуальну проблему автоматизації процесу побудови реєстру ризиків у розробці програмного забезпечення на основі аналізу даних issue-трекінгу в GitHub Projects. В умовах зростання складності програмних продуктів та високої динаміки змін життєвого циклу ІТ-проєктів традиційні ручні підходи до ідентифікації ризиків дедалі більше втрачають свою ефективність через затримки оновлення інформації та значні людські витрати. Автором запропоновано концепцію та реалізовано прототип Python-скрипта, який забезпечує автоматизоване вилучення, попередню обробку, багаторівневу класифікацію та формування структурованого реєстру ризиків, що дозволяє інтегрувати цей процес безпосередньо у звичний життєвий цикл управління якістю проєкту. Запропонований підхід ґрунтується на класифікації ризиків за визначеною таксономією, що охоплює сім основних типів: інженерні, середовищні, процесні, пов'язані з обмеженнями, безпекові, поведінкові та зовнішні ризики. Для кожної групи ризиків визначено перелік релевантних ключових слів і текстових індикаторів, що забезпечує ефективну роботу алгоритму keyword matching у поєднанні з аналізом labels, коментарів та стану задач. У межах апробації скрипт було протестовано на штучно сформованому датасеті, що містить понад 150 завдань (issues) з різноманітними текстами, наближеними до реальної практики відкритих проєктів. Важливою особливістю прототипу є можливість мультикласової класифікації, коли одна задача може одночасно відповідати кільком групам ризиків, що адекватно відображає міждисциплінарність сучасних проблем у сфері розробки ПЗ. Ефективність роботи скрипта забезпечується високою швидкодією: середній час обробки повного масиву даних не перевищує 1,5 секунди, що дозволяє інтегрувати рішення у регулярні CI/CD-конвеєри через GitHub Actions. Стаття також окреслює методологічну базу прогнозування ефективності розробленого підходу з використанням даних із сучасних досліджень, зокрема результатів моделі BEACon-TD, яка продемонструвала F1-міру понад 0.81–0.86 на реальних наборах issue-трекінгу. Це дозволяє аналітично прогнозувати потенційну точність запропонованого підходу на рівні 80–85% за належного донавчання та адаптації ключових слів під специфіку проєкту. Запропоноване рішення дозволяє не лише виявляти проблемні зони на ранніх етапах, але й формувати основу для подальшої прогнозної аналітики щодо впливу ризиків на часові, фінансові та якісні параметри розробки. Крім того, скрипт створює структурований реєстр у форматах JSON та Excel із налаштованим оформленням і надає візуалізацію розподілу типів ризиків за допомогою діаграм у Matplotlib, що підвищує прозорість та зручність використання результатів аналізу. Отримані дані можуть інтегруватися у системи управління якістю або передаватися стейкхолдерам для своєчасного прийняття рішень. У підсумку доведено, що запропонована автоматизована побудова реєстру ризиків на основі GitHub Projects є перспективним напрямом удосконалення процесів управління якістю ПЗ та управління технічним боргом, а розроблений прототип демонструє приклад практичної реалізації цієї концепції у поєднанні з гнучкими підходами Agile та DevOps.

Ключові слова: *issue-трекінг GitHub, реєстр ризиків, автоматизована класифікація, управління якістю ПЗ, технічний борг, Python-скрипт, прогнозування ризиків.*

Вступ. В умовах зростаючої складності програмних систем та широкого застосування гнучких методологій розробки управління ризиками залишається критичною складовою забезпечення якості програмного забезпечення. Незважаючи на розвиток численних підходів до ідентифікації та класифікації ризиків, практична інтеграція ризик-менеджменту у повсякденний життєвий цикл IT-продукту часто ускладнена через відсутність оперативної, актуальної інформації про прояви ризиків у робочих артефактах проєкту. Одним із перспективних напрямів вирішення цієї проблеми є використання даних issue-трекінгу як джерела сигнальної інформації про ризики різних категорій.

Відомо, що на сучасних платформах розробки, таких як GitHub Projects, накопичується значна кількість відкритих даних: issues, pull requests, коментарі, історія комітів та метрики активності команд. Ці дані потенційно містять важливі індикатори як технічних боргів (technical debt), так і організаційних або процесних ризиків. Однак традиційний підхід до формування реєстру ризиків залишається переважно ручним та декларативним, що знижує його релевантність у швидкозмінному середовищі розробки.

Сучасні дослідження демонструють, що використання методів машинного навчання та NLP для автоматизованого аналізу змісту задач і обговорень дозволяє підвищити точність виявлення ризиків. Зокрема, у роботі Karthik Shivashankar et al. (2025) запропоновано модель BEACon-TD для класифікації технічних боргів на основі трансформерів, що довела свою ефективність при роботі з масштабними архівами GitHub Issues [1]. Подібним чином у CAPRA (Tang et al., 2025) розроблено контекстно-орієнтовану систему оцінки патч-ризиків, що враховує специфіку відкритого коду та допомагає своєчасно виявляти незрілі або потенційно небезпечні зміни [2]. Ці приклади засвідчують високий потенціал автоматизованих підходів, які використовують реальні трекінгові дані.

Водночас, у рамках сучасних концепцій управління якістю IT-продуктів, важливо не лише ідентифікувати окремі фактори ризику, але й забезпечити їх класифікацію та моніторинг у режимі реального часу.

Автоматизована побудова реєстру ризиків на основі issue-трекінгу у GitHub Projects дозволяє поєднати ідеї моніторингу якості, управління технічним боргом та динамічної адаптації заходів управління ризиками протягом життєвого циклу продукту. Основною метою цієї статті є розробка концепції та інструментального рішення (Python-скрипт або GitHub Action), що забезпечує автоматизоване вилучення, класифікацію та реєстрацію ризиків з урахуванням специфіки окремого проєкту та його поточного стану.

Автоматизована побудова реєстру ризиків у розробці програмного забезпечення (ПЗ) дедалі частіше розглядається як ключовий напрям удосконалення управління якістю та технічним боргом. Значна частина досліджень фокусується на інтеграції сучасних методів машинного навчання та обробки природної мови (NLP) для класифікації ризиків безпосередньо з трекерів задач, таких як GitHub Projects.

Аналіз існуючих досліджень. Дослідження Karthik Shivashankar та співавт. [1] присвячене побудові автоматизованої системи класифікації технічного боргу (TD) у завданнях issue-трекінгу для різних open-source проєктів. Автори розробили підхід BEACon-TD (Binary Ensemble for Automated Classification of Technical Debt), який поєднує кілька спеціалізованих двійкових класифікаторів на основі трансформерних моделей типу DistilRoBERTa. Така ансамблева архітектура дозволяє ідентифікувати як наявність боргу загалом, так і його тип (архітектурний, тестовий, кодовий тощо) із високою точністю.

Методологія дослідження включала збирання великого корпусу GitHub Issues за період 2015–2024 рр., а також перевірку моделей на аут-оф-дистрибуційних (OOD) даних, що дозволило оцінити їхню здатність до генералізації у реальних промислових умовах. Важливим аспектом є порівняльний аналіз різних моделей: автори довели перевагу спеціалізованих бінарних класифікаторів над мультикласовими та виявили, що менш ресурсоємна DistilRoBERTa після тонкого налаштування перевершує більші LLM-моделі (наприклад, GPT-4o) за ключовими метриками (Precision, Recall, F1, MCC).

Серед отриманих результатів слід відзначити середню F1-міру понад 0.81 на тестових наборах та стабільність показників на OOD-наборах. Додатково було підтверджено, що fine-tuning моделей на даних конкретного проєкту підвищує якість класифікації до 20–30% для більшості метрик. Це підходить до ідеї гнучкого, адаптивного реєстру ризиків у розробці ПЗ, коли класифікаційна модель може

враховувати специфіку проєкту та поточний контекст трекінгових даних.

BEACon-TD демонструє реальну можливість масштабованої та детальної ідентифікації технічного боргу з використанням даних issue-трекінгу, що безпосередньо перегукується з завданнями автоматизованого формування реєстру ризиків у GitHub Projects у контексті даного дослідження.

У дослідженні Benxiao Tang та співавт. [2] запропоновано метод CAPRA (Context-Aware Patch Risk Assessment), що фокусується на виявленні так званих «immature vulnerabilities» у відкритому програмному забезпеченні ще на стадії перегляду патчів. Автори обґрунтовують необхідність такого підходу тим, що в колаборативних open-source проєктах різнорідний рівень кваліфікації розробників та складність коду часто ускладнюють своєчасне виявлення дефектів без урахування контексту змін. Традиційні методи статичного аналізу або ручного аудиту, як правило, спрацьовують лише після злиття патча, що відкриває вікно для тривалого існування прихованих вразливостей.

CAPRA реалізує інтеграцію статичного аналізу безпосередньо у CI/CD-процес за допомогою code property graph (CPG), який об’єднує синтаксичну структуру, контрольні потоки виконання та залежності даних. Система виконує поелементний аналіз кожного патча, порівнюючи початкові (pre-patched) та змінені файли (post-patched), що дозволяє визначати потенційні пам’яттєві витоки (memory leaks) та Use-After-Free вразливості до їх потрапляння у production. Особливість CAPRA — детекція ризиків за сценаріями «доданих» та «видалених» рядків коду, що робить її гнучкою та релевантною для рутинних рев’ю невеликих патчів.

За результатами експериментів на оригінальному датасеті (з використанням OpenSSH та «hypocrite commits») модель досягла середньої точності 97,3% при recall близько 98% та лише 3,5% false positive, що демонструє високу практичну ефективність запропонованого рішення [2]. Автори підкреслюють, що така контекстно-орієнтована оцінка підвищує якість перевірки безпеки навіть для малих патчів (<30 LOC), які традиційно отримують мінімальну увагу рецензентів.

CAPRA демонструє можливість застосування контекстного аналізу змін для підвищення точності оцінки ризиків у динамічних середовищах розробки. Це безпосередньо корелює з метою даної статті — розширити принципи класифікації ризиків з рівня патчів на рівень задач issue-трекінгу GitHub, формуючи основу для автоматизованого та адаптивного реєстру ризиків протягом життєвого циклу ПЗ.

Дослідження Cataldo Basile, Bjorn De Sutter та співавт. [3] присвячене розробці, впровадженню та автоматизації комплексного підходу до управління ризиками в контексті захисту ПЗ від атак типу Man-At-The-End (MATE). У таких атаках зловмисник має повний контроль над середовищем виконання і може застосовувати будь-які інструменти реверс-інжинірингу, що ставить під загрозу ключові активи — алгоритми, ключі шифрування чи ліцензійні механізми. Автори пропонують застосовувати стандартизовану методологію управління ризиками на основі підходу NIST SP800-39, адаптованого для сфери програмного захисту.

У роботі окремо наголошено, що класичні SP (Software Protection) техніки часто базуються на security-through-obscurity і залишаються малоефективними через відсутність формалізації та автоматизації. Для подолання цих недоліків Basile та колеги поєднують bottom-up і top-down дизайн: знизу вони розробляють Proof-of-Concept (PoC) інструмент підтримки прийняття рішень (ESP), а зверху — формують універсальну модель ризик-менеджменту, що описує конструкти, моделі, методи й можливості їх автоматизації.

Важливо, що валідовані експерименти PoC показали, що навіть часткова автоматизація дозволяє підвищити узгодженість рішень експертів і зменшити залежність від суб’єктивних оцінок. Зокрема, перевірка інструменту на кейсах реальних Android-додатків засвідчила корисність моделі для промислових сценаріїв. Авторами також сформульовано відкриті дослідницькі питання щодо формалізації оцінки ефективності SP, розробки спільної термінології та обґрунтованого ризик-аналізу в домені MATE.

З точки зору контексту теми даної статті, такий підхід є показовим прикладом того, як стандартизований та частково автоматизований ризик-менеджмент може бути реалізований для складних сценаріїв, де ручна експертиза має обмеження масштабованості. Аналогічні принципи — формалізація, автоматизоване прийняття рішень і опора на перевірені стандарти — можуть бути використані при побудові реєстру ризиків у розробці ПЗ на основі issue-трекінгу GitHub, що прямо

підсилює обґрунтування практичного підходу у цій роботі.

У дослідженні Ashley T. van Can та співавт. [4] розглянуто проблему ідентифікації вимог у беклогах, що зберігаються в системах issue-трекінгу в умовах Agile-розробки. Автори акцентують увагу на тому, що через мінімалізм формальної документації у гнучких командах беклоги часто містять вимоги, змішані з багами, завданнями та ідеями, без чіткої структуризації. Це ускладнює простежуваність і подальше тестування, що є критичним для підтримки якості ПЗ.

В межах першої частини роботи автори виконали кількісний контент-аналіз 1636 елементів беклогу із 14 проєктів (як open-source, так і корпоративних). Результати показали високу непослідовність у маркуванні backlog items: у середньому понад 40% записів із позначкою «вимога» фактично не містили вимог, а близько 22% записів, що були марковані як завдання, навпаки, містили одну або кілька вимог. Також було встановлено, що часто в одному елементі беклогу фіксується одразу кілька вимог різного рівня деталізації, що ускладнює їх унікальну ідентифікацію та трасування.

Друга частина дослідження зосереджена на оцінці ефективності великих мовних моделей (LLM) для автоматизованого вилучення та класифікації вимог. Van Can та колеги експериментально порівняли encoder-only моделі (BERT, RoBERTa) з decoder-only моделями (ChatGPT, Mistral 7B, Llama 3) за задачами ідентифікації та класифікації вимог. Результати показали, що моделі типу BERT та RoBERTa демонструють стабільно вищі показники Precision, Recall і F1-міри для класів «користувацькі функціональні вимоги», «системно-орієнтовані функціональні» та «нефункціональні вимоги», ніж генеративні LLM. Зокрема, середня F1-міра для класу вимог склала понад 0.57 для RoBERTa, тоді як для ChatGPT — лише близько 0.38.

Автори роблять висновок про доцільність створення unobtrusive інструментів, які б автоматично визначали й класифікували вимоги без зміни звичних практик команди. Це напряму співзвучно із завданням побудови автоматизованого реєстру ризиків, оскільки такі підходи можна адаптувати для витягання індикаторів ризиків з backlog items у GitHub Projects. Отримані результати демонструють потенціал використання LLM та encoder-only моделей для покращення прозорості й керованості інформації у життєвому циклі розробки.

У роботі Shekar Ramachandran та співавт. [5] досліджується автоматизація класифікації системних логів із використанням глибинного навчання для підвищення точності та ефективності аналізу аномалій у великих обсягах неструктурованих даних. Автори підкреслюють, що традиційні методи аналізу логів часто потребують значних ручних зусиль і не масштабуються для складних кластерних середовищ, що дедалі активніше використовуються у хмарних технологіях.

Ключовим внеском дослідження є розробка власного конвеєра обробки, що включає нову техніку векторизації LogWord2Vec, методи очищення й аугментації даних, а також гібридну архітектуру на основі CNN-LSTM. Техніка LogWord2Vec дозволяє перетворювати неструктуровані логи у числові вектори, зберігаючи частотні характеристики слів та обробляючи синоніми, що мінімізує ризики переобучення моделей на вузькому словнику розробників. Автори також застосували лівостороннє падінгування, що дає змогу LSTM ефективніше відновлювати послідовності після «шуму» нульових значень.

Важливою складовою підходу є продумана аугментація даних для балансування класів: оскільки помилкові рядки трапляються значно рідше, застосовується заміна слів на синоніми, перестановка та вибіркове видалення слів у рядках логів. Такий підхід дозволяє уникнути перекосу даних і підвищує стійкість класифікатора.

У експериментах Ramachandran та колеги порівняли різні методи векторизації, включаючи BERT, TensorFlow TextVectorization та власний LogWord2Vec. Найкращі результати показала модель CNN-LSTM з LogWord2Vec, досягнувши точності класифікації 99.19% при лише 0.81% помилкових класифікацій. Для порівняння, BERT продемонстрував значно гірші результати через низьку релевантність загального словника для специфічних логів.

У контексті теми даної статті розглянута робота демонструє, що поєднання глибинного навчання, кастомних методів векторизації та аугментації може значно підвищити якість обробки текстових артефактів у розробці ПЗ. Аналогічні підходи можуть бути адаптовані для автоматизованої класифікації та виявлення ризиків на основі текстових даних issue-трекінгу у GitHub Projects.

У роботі Ziwei Liu та співавт. [6] представлено дизайн та прототип системи моніторингу транзакцій і зворотного зв'язку щодо ризиків, орієнтованої на DevOps-процеси у високонавантажених фінансових середовищах. Автори наголошують, що традиційні підходи до моніторингу та оцінки ризиків часто не справляються з обробкою великих обсягів транзакційних даних у реальному часі, що призводить до низької точності прогнозів і несвоєчасного реагування.

Запропонована система базується на мікросервісній архітектурі, що забезпечує модульність і масштабованість для підтримки високої конкурентності та низької затримки обробки даних. Ключовою особливістю є інтеграція методів машинного навчання для автоматичного виявлення аномальних транзакцій і генерації прогнозів ризиків. Архітектура включає чотири основні модулі:

1. Збір та попередня обробка даних (з використанням Python та Redis для кешування);
2. Модуль реального часу, що застосовує Elasticsearch для агрегації даних та аномалій;
3. Модуль прогнозування ризиків, де використовуються SVM, Decision Tree та Random Forest із балансуванням класів (SMOTE) для підвищення точності моделей;
4. Модуль зворотного зв'язку, який використовує Kafka та мультиканальне сповіщення (Email, SMS, корпоративні чати) для оперативної доставки повідомлень про ризики відповідальним особам.

Автори показали, що використання Grubbs' Outlier Test та KNN у стадії очищення даних дозволяє суттєво знизити рівень хибних спрацьовувань. Експериментальна оцінка продемонструвала, що комбінування мікросервісів та ML-моделей значно підвищує ефективність моніторингу, забезпечуючи швидке виявлення транзакцій з високим ризиком і своєчасну реакцію в рамках DevOps-конвеєра.

Цей підхід має безпосереднє значення для даного дослідження: поєднання архітектурних рішень, модульного моніторингу та алгоритмів класифікації може бути адаптовано для формування автоматизованого реєстру ризиків у розробці ПЗ. Зокрема, принципи цілісного циклу «збір–аналіз–прогноз–зворотний зв'язок» можуть лягти в основу Python-скриптів або GitHub Actions, що здійснюють класифікацію ризиків на основі даних issue-трекінгу GitHub Projects.

У роботі Mohammed Amin Almaiah та співавт. [7] виконано комплексну класифікацію кіберзагроз, вразливостей і відповідних заходів протидії у сфері систем управління базами даних (DBMS). Автори підкреслюють, що через критичність даних, які зберігаються у базах, ці системи залишаються однією з найпопулярніших цілей для кібератак. Дослідження систематизує технічні та нетехнічні загрози, їх поширеність та наслідки, а також окреслює практичні кроки з впровадження контрзаходів.

Методологія роботи ґрунтується на контент-аналізі, який охоплює велику вибірку даних з реальних інцидентів. Зокрема, SQL-ін'єкції та атаки типу DoS ідентифіковано як найбільш поширені технічні загрози — кожна з них становить 9% зареєстрованих випадків. До другої групи загроз увійшли уразливі audit trail-и, спроби вторгнень і атаки програм-вимагачів (ransomware). Серед нетехнічних загроз найбільш поширеними є внутрішні інсайдерські загрози (5%), що особливо актуально для організацій з великою кількістю користувачів з розширеними привілеями.

Особливу увагу Almaiah та колеги приділили взаємозв'язку між типами загроз і вразливостей. Найбільш поширеними вразливостями визначено слабку автентифікацію, непатчені бази даних, уразливі audit trail-и та мультиакаунтинг, кожен з яких становить близько 9% від усіх інцидентів. Другий рівень складають баги в ПЗ, небезпечні практики кодування, слабкі засоби контролю безпеки, проблеми з мережами, неправильне використання паролів та слабке шифрування.

У якості рекомендацій дослідження пропонує багаторівневий комплекс контрзаходів: від шифрування, багатофакторної автентифікації, сегментації мережі й Honeypots до застосування систем IDS/IPS, спам-детекторів та поведінкової аналітики. Окремо підкреслюється значення підвищення обізнаності працівників і регулярних аудитів.

У контексті даної статті цей підхід демонструє цінність класифікації ризиків і побудови динамічного реєстру на основі актуальних даних. Принципи картографування загроз і вразливостей можна безпосередньо застосувати до аналізу активності issue-трекінгу у GitHub Projects, що створює можливість для автоматизованої побудови і адаптації реєстру ризиків протягом життєвого циклу ПЗ.

Дослідження Hamed Hasani, Francesco Freddi та Riccardo Piazza [8] представляє

комплексний AI-драйвовий фреймворк для автоматизованого моніторингу структурного стану (Structural Health Monitoring, SHM), що інтегрує всі ключові етапи: модальну ідентифікацію, моніторинг стану та локалізацію пошкоджень — із урахуванням впливу зовнішніх і експлуатаційних факторів. Автори зазначають, що традиційні SHM-системи часто ігнорують змінні умови експлуатації (температура, вологість, навантаження), що призводить до хибних тривог і обмежує ефективність таких систем.

Запропонований фреймворк використовує поєднання стохастичної субпросторової ідентифікації (SSI-Cov) для автоматизованого визначення модальних параметрів із кластеризацією Gaussian Mixture Model (GMM) для стабільного вибору полюсів. Для безперервного моніторингу стану впроваджено autoencoder-нейронні мережі (AE-ANN), які навчаються на «здоровому» стані структури, а виявлення аномалій ґрунтується на пороговому аналізі помилок реконструкції. Для автоматизованої локалізації пошкоджень застосовано комбінований підхід: дискретне вейвлет-перетворення (DWT) для багаторівневої декомпозиції сигналу та AE-LSTM для розпізнавання шаблонів відхилень.

Фреймворк валідовано на лабораторній моделі мосту, де демонстровано його здатність точно ідентифікувати втрати жорсткості навіть при впливі температури та вологості. Результати підтвердили, що поєднання GMM, LSTM та wavelet-аналізу мінімізує хибні спрацьовування і дає високі показники достовірності пошкоджень у реальному часі.

У контексті теми цієї статті проналізована робота є прикладом того, як інтеграція ML-методів і автоматизованих процесів на всіх рівнях моніторингу дозволяє отримати гнучкі та адаптивні системи оцінки ризиків. Аналогічно, такі методи можуть застосовуватися для побудови автоматизованого реєстру ризиків на основі issue-трекінгу GitHub, де зовнішні фактори (наприклад, сезонність комітів чи зміни в командах) також можуть спотворювати оцінку ризиків. Таким чином, підхід Hasani та колег є релевантним взірцем для розробки Python-скриптів або GitHub Actions для динамічного спостереження й класифікації ризиків протягом життєвого циклу ПЗ.

У дослідженні Zsuzsa Farkas та співавт. [9] представлено комплексний підхід до систематичної ідентифікації нових ризиків у харчовому ланцюзі із застосуванням сучасних методів обробки даних і багаторівневої експертної оцінки. Автори наголошують, що завчасне виявлення так званих emerging risks є критично важливим для захисту здоров'я споживачів та забезпечення сталого розвитку галузі, однак воно ускладнюється великими обсягами різнорідних даних і прогалинами в інформації.

Ключовою складовою роботи є розробка уніфікованої процедури трирівневої фільтрації: від початкового збору та попереднього аналізу даних (PHASE I), через відбір потенційних emerging risks (PHASE II), до їх остаточної оцінки за системою багатокритеріального скорингу (PHASE III). Для підтримки цього процесу команда використовує різні алгоритми машинного навчання, методи text mining і мережевий аналіз, що суттєво скорочує трудомісткість ручної перевірки.

Емпірична перевірка методики показала її ефективність: у 2020–2021 рр. командою Farkas та колег було виявлено 58 emerging risks, класифікованих у 10 основних тем. Це мікробіологічна безпека свіжих продуктів, проблеми з кормами для тварин, мікро- та нанопластики, стійкість до антимікробних препаратів, хімічні контамінації, питання сталого розвитку (зокрема альтернативні джерела білка), технологічні інновації, вплив кліматичних змін, поява нових патогенних мікроорганізмів і споживчі тренди. Окремі ризики стали основою для нових національних моніторингових програм і були передані для подальшої оцінки у рамках Emerging Risk Exchange Network (EREN) Європейського органу EFSA.

Особливу увагу приділено інтеграції автоматизованих алгоритмів із експертними оцінками. Автори зазначають, що навіть найкращі алгоритми не можуть повністю замінити людське експертне судження, зокрема на стадіях остаточного скорингу і прийняття рішень про заходи реагування.

У контексті дослідження ця робота є цінним прикладом того, як систематизований моніторинг, багатоетапна фільтрація та використання AI-технологій можуть бути адаптовані для побудови реєстрів ризиків у розробці ПЗ. Зокрема, аналогічний принцип відбору індикаторів і трендів із потоків issue-трекінгу GitHub дозволяє створювати Python-скрипти чи GitHub Actions, які автоматично оновлюють реєстр ризиків, враховуючи як кількісні сигнали, так і експертну інтерпретацію.

У роботі A.K.M. Ahasan Habib, Mohammad Kamrul Hasan та співавт. [10] описано створення та публікацію унікального набору даних UKMNCT_IIoT_FDIA, призначеного для дослідження атак підміни даних (False Data Injection, FDI) у середовищах Industrial Internet of Things (IIoT), що є ключовим компонентом інфраструктури Industry 5.0. Автори підкреслюють, що зростаюча складність і взаємозв'язаність IIoT-систем підвищують ризики впровадження FDI-атак, які здатні маніпулювати переданими даними та порушувати точність оцінок стану системи.

Особливістю підходу є побудова гетерогенного середовища для збору реалістичних даних з імітацією різних сценаріїв атак: реплей-атаки, симуляційні скрипти, реальний MitM-трафік. Зібрані дані містять понад 15 000 записів з 30 атрибутами, включаючи IP-адреси, протоколи, порти, довжину пакетів, статуси SSL та HTTP-запитів. Це дозволяє дослідникам моделювати поведінку атак у різних мережевих конфігураціях.

Набір даних апробовано за допомогою ансамблевого ML-алгоритму Gradient Boosting, що показав високі результати: F1-score, Precision і Recall — 0.99 для класифікації атак і нормального трафіку. Також автори продемонстрували низький рівень FP та FN помилок, що підтверджує надійність даних для тренування та тестування систем виявлення вторгнень.

У статті наголошується, що доступність подібних реалістичних датасетів сприяє розробці ефективних алгоритмів детекції та попередження атак у IIoT, що є критично важливим для підвищення безпеки критичної інфраструктури — енергетики, розумних міст і промислових об'єктів.

У контексті теми цієї статті цей підхід демонструє значущість побудови спеціалізованих наборів даних та моделей класифікації аномалій, які можуть бути перенесені на аналіз текстових артефактів у GitHub Projects. Використання Python-скриптів чи GitHub Actions для автоматизованого збору та класифікації ризиків issue-трекінгу ґрунтується на аналогічних принципах: симуляція реальних сценаріїв ризиків, гнучке налаштування атрибутів і постійне оновлення тренувальних даних.

У роботі Muhammad Hassan, Giovanna Salbitani, Simona Carfagna та Javed Ali Khan [11] розроблено високоточну та інтерпретовану систему автоматизованої класифікації планктону, що поєднує глибоке навчання, оптимізоване злиття ознак та explainable AI. Автори зазначають, що традиційна таксономічна ідентифікація планктону є трудомісткою, потребує високої кваліфікації та схильна до помилок, що значно гальмує екологічний моніторинг морських екосистем.

Ключовою інновацією дослідження є гібридна модель, яка об'єднує потужності InceptionResNetV2 (з трансферним навчанням) та нової нейронної мережі DeepPlanktonNet, натренованої з нуля для захоплення специфічних ознак даних. Для підвищення дискримінативної здатності було реалізовано злиття ознак, а для скорочення надлишкових даних і підвищення обчислювальної ефективності — Whale Optimization Algorithm (WOA). Додатково для забезпечення прозорості застосовано підхід LIME (Local Interpretable Model-Agnostic Explanations), що дозволяє зрозуміти, як модель приймає рішення, і виявити найбільш релевантні ділянки зображень планктону.

Експерименти на відкритому датасеті WHOI продемонстрували точність класифікації 98.79%, що суттєво перевищує показники попередніх state-of-the-art рішень. Автори також порівняли гібридну модель з низкою традиційних архітектур CNN і класичних ML-класифікаторів, що підтвердило переваги запропонованого підходу за всіма ключовими метриками (F1-score, Precision, Recall) і показало зниження обчислювальних витрат після WOA-оптимізації.

У контексті теми цієї статті дослідження Hassan та колег демонструє релевантність комплексних гібридних фреймворків для автоматизації обробки великих і складних даних, що критично важливо при побудові реєстрів ризиків. Зокрема, використання оптимізації ознак і explainable AI є корисним прикладом для проєктування Python-скриптів або GitHub Actions, що можуть не лише класифікувати issue у GitHub Projects, а й робити цей процес прозорим і обґрунтованим для стейкхолдерів.

У статті Alfredo Sánchez-Hernández, Dídac Román, Peyman Javadi та Inés Domingo [12] розглянуто впровадження інтегрованого підходу GIS та SfM-фотограмметрії для моніторингу фізичних змін і оцінки ризиків на пам'ятках наскельного мистецтва. Автори наголошують, що традиційні методи археологічного моніторингу часто є недостатніми для вчасного виявлення

деградаційних процесів, які можуть не лише зруйнувати безцінні культурні об’єкти, а й створювати небезпеку для відвідувачів.

Дослідження передбачало розробку та апробацію триетапної методики, яка поєднує побудову цифрових моделей рельєфу (DEM) із регулярними фотограмметричними обстеженнями та обробкою в GIS для порівняння змін у часі. Для цього команда реалізувала два експериментальні кейси на природних скельних стінах і один повноцінний кейс — на реальному археологічному об’єкті La Covatina, який є частиною Світової спадщини ЮНЕСКО.

В експериментах метод успішно виявив як великі втрати скельного матеріалу, так і мікроскопічні зміни розміром у кілька квадратних сантиметрів. Зокрема, результати показали, що навіть незначні зсуви або утворення тріщин у скельних шарах можуть бути зафіксовані, що дозволяє прогнозувати ризики подальших обвалів. У реальному кейсі було задокументовано понад 50 фізичних змін (зокрема обвали різного масштабу), що підтверджує практичну цінність підходу для довготривалого моніторингу.

Додатково, автори звертають увагу на проблеми автоматизації при роботі на відкритому повітрі: зміни освітлення, погодні умови та вплив рослинності створюють шум у моделях, тому частину аналізу доводиться доповнювати візуальною експертною перевіркою.

У контексті теми цієї статті робота Sánchez-Hernández та колег є важливою ілюстрацією того, як поєднання просторових технологій, 3D-моделювання і регулярного моніторингу дозволяє будувати динамічні системи оцінки ризиків у режимі реального часу. Такий принцип може бути застосований для автоматизованого формування реєстрів ризиків у ПЗ, де зміни в issue-трекінгу GitHub Projects можуть аналізуватися за допомогою Python-скриптів чи GitHub Actions, використовуючи аналогічну логіку періодичного порівняння стану системи і побудови моделей ризиків.

У роботі Giuseppe Santarsiero [13] запропоновано інтегровану методику для автоматизованої оцінки стану та пріоритизації оновлення дорожніх відбійників на мостах із використанням відкритих геопросторових даних і алгоритмів глибокого навчання. Автор звертає увагу, що у багатьох муніципалітетах і провінціях Італії відсутні детальні й актуальні інвентарі мостів і відбійників, що суттєво ускладнює управління критичною інфраструктурою та підвищує ризики аварій через застарілі або пошкоджені бар'єри.

Ключовою інновацією роботи є синергетичне поєднання трьох інструментів: OpenStreetMap (OSM) для формування бази даних мостів, Google Street View API для автоматичного отримання зображень бар’єрів і YOLOv8 — популярного алгоритму комп’ютерного бачення — для автоматизованого виявлення та класифікації типів бар’єрів щодо їх відповідності чинним стандартам. Ця інтеграція реалізована через VBA-рутини, що забезпечує доступність методики навіть для користувачів без спеціальної технічної підготовки.

В якості апробації метод було застосовано до 776 мостів у регіоні Базиліката. Результати тестування на вибірці з 194 зображень демонструють високі показники Recall (0.928), Precision (0.900) і F1-міри (0.914), що свідчить про надійність автоматизованої класифікації. Цікаво, що дослідження фокусується на мінімізації False Negatives (недовиявлення невідповідних бар’єрів), що є критичним для безпеки дорожнього руху.

Окрім технічної частини, Santarsiero також представив економічний аналіз витрат на заміну застарілих відбійників, який для вибірки оцінився у понад €67 млн. Автор обґрунтував пріоритизацію робіт на основі типів доріг та довжини мостів, що дозволяє оптимально розподіляти ресурси у регіональному масштабі.

У контексті теми цієї статті ця робота демонструє, як автоматизовані фреймворки, що поєднують відкриті джерела даних, машинне навчання та геопросторовий аналіз, можуть суттєво підвищити ефективність управління ризиками. Такий підхід можна адаптувати до побудови реєстру ризиків у розробці ПЗ: замість зображень відбійників об’єктом аналізу стають дані issue-трекінгу GitHub, а принципи автоматизованого збору, класифікації і візуалізації ризиків залишаються подібними.

У статті Sk Tahsin Hossain, Tan Yigitcanlar, Kien Nguyen та Yue Xu [14] презентовано інноваційну платформу, що поєднує концепцію “platform urbanism” із підходом AIoT (Artificial-Intelligence-of-Things) для побудови прогнозної системи моніторингу мікрокліматичних ризиків і сповіщення населення в режимі реального часу. Автори звертають увагу, що традиційні централізовані системи оповіщення про погоду здебільшого ігнорують просторову мінливість мікрокліматів у межах великих міст, що створює

прогалини в захисті особливо вразливих груп населення.

Запропонована платформа інтегрує ArcGIS GeoEvent Server, модулі машинного навчання та демографічні дані для формування локалізованих сповіщень. Архітектура передбачає поєднання потокових даних із метеорологічних сенсорів, AI/ML-моделей прогнозування та контекстних повідомлень для мешканців із урахуванням їхнього стану здоров'я, віку та розташування. На прикладі міста Брісбен (Австралія) доведено, що система здатна відображати значні відмінності у тепловому стресі та опадах між районами, що традиційно не враховується централізованими платформами.

Ключові результати демонструють ефективність платформи у виявленні теплових аномалій, прогнозуванні теплих днів/ночей та сильних опадів, що дозволяє органам місцевого самоврядування адаптувати екстрені заходи: від відкриття центрів охолодження до оперативного сповіщення груп ризику. Тестування платформи показало високу точність просторової інтерполяції температурних режимів і інтенсивності опадів для понад 190 районів міста. Важливо, що система пропонує індивідуалізовані алерти, мінімізуючи "загальне шумове оповіщення", яке часто ігнорується мешканцями.

У контексті теми цієї статті підхід Hossain та колег є переконливим прикладом того, як гібридні AI/IoT-фреймворки, що поєднують потокові дані, класифікацію ризиків і геопросторову аналітику, можуть ефективно автоматизувати відстеження ризиків. Для сфери розробки ПЗ це створює концептуальну паралель: аналогічно до моніторингу мікроклімату, Python-скрипти чи GitHub Actions можуть здійснювати безперервний збір, оцінку та адресне сповіщення про ризики, базуючись на динамічному аналізі issue-трекінгу у GitHub Projects.

Виклад основного матеріалу дослідження

Для реалізації поставленої мети дослідження — автоматизованої побудови реєстру ризиків у процесі розробки програмного забезпечення на основі даних issue-трекінгу в GitHub Projects — було організовано комплексну експериментальну установку, що поєднує розроблений Python-скрипт, етапи симуляції API-запитів і засоби інтеграції з GitHub Actions.

Апробація скрипта проводилася у тестовому середовищі, що імітує умови реального репозиторію великого проєкту. Для цього був створений репозиторій із штучно згенерованим масивом із 150 issues, структурованих відповідно до формату GitHub REST API: кожен запис містив унікальний ідентифікатор, номер, заголовок, детальний опис, набір labels, інформацію про відповідального виконавця (за наявності), кількість коментарів, стан (відкритий чи закритий) та посилання на репозиторій. Текстове наповнення issues було сформоване таким чином, щоб максимально відобразити варіативність ключових слів, типових для кожного з визначених типів ризиків. Загальний лексичний обсяг тестового набору складав близько 7000 унікальних слів, що дозволяло моделювати багатий контекст і перевіряти стійкість класифікаційної моделі.

Структура класифікації ризиків у скрипті ґрунтувалася на таксономії, що передбачає виділення семи основних груп: інженерних, середовищних, процесних ризиків, ризиків, пов'язаних із проєктними обмеженнями, безпекових, поведінкових і зовнішніх. Для кожної з цих груп було визначено перелік релевантних ключових слів та фраз, що дозволяли виконувати автоматизовану класифікацію за методом keyword matching, а також аналіз текстових міток (labels), що використовуються командами розробки для маркування задач у GitHub. Таким чином, перевірка кожного issue здійснювалася за багаторівневою схемою: на етапі обробки даних аналізувався заголовок і опис задачі, перелік міток, кількість коментарів та статус запису. Особливу увагу приділено можливості мультикласової належності: окремі issues могли одночасно відповідати кільком категоріям ризиків залежно від кількості виявлених ключових ознак.

У ході експерименту скрипт формував структурований реєстр ризиків у вигляді JSON-файлу та окремої таблиці Excel з попередньо налаштованим оформленням. Таблиця містила всі ідентифікатори, номери, визначені типи ризиків, URL для швидкого переходу до завдання, поточний стан, дані про виконавця та кількість коментарів. Для кращої інтерпретації результатів була реалізована візуалізація розподілу ризиків за допомогою бібліотеки Matplotlib, що дозволяло наочно оцінити пропорції виявлених груп у всьому масиві проєкту. Також скрипт був адаптований для

регулярного запуску через GitHub Actions, що забезпечувало оновлення реєстру ризиків у реальному часі.

У рамках контрольних вимірювань було зафіксовано ключові параметри роботи: середній час класифікації повного масиву із 150 issues становив близько 1,5 секунди, що свідчить про достатню швидкодію прототипу навіть при масштабуванні. Для перевірки стабільності результатів класифікація проводилася повторно з варіаціями ключових слів, що дозволяло визначити стійкість моделі до зміни формулювань та перевірити гнучкість логіки keyword matching у поєднанні з аналізом метаданих.

Загалом сформована експериментальна установка створила умови, максимально наближені до реальної практики використання issue-трекінгу в проєктах з відкритим вихідним кодом або корпоративною розробкою, що забезпечило можливість комплексної перевірки запропонованого підходу до автоматизованої побудови та адаптації реєстру ризиків протягом усього життєвого циклу ПЗ.

Алгоритм роботи скрипта для автоматизованої побудови реєстру ризиків реалізовано як послідовність взаємопов'язаних етапів, що забезпечують повний цикл обробки даних issue-трекінгу (рис. 1).

На початковому кроці відбувається підключення до джерела даних, яким у процесі апробації виступав локальний файл JSON, що моделює реальну відповідь GitHub REST API з відповідною структурою полів: id, number, title, body, labels, state, assignee, comments та html_url. На цьому етапі здійснюється перевірка на коректність структури даних та відсікання порожніх або дубльованих записів, що дозволяє уникнути помилок подальшої обробки.

Далі кожен запис опрацьовується у циклі згідно з багаторівневою схемою класифікації. Спочатку формується суцільний текстовий блок шляхом об'єднання заголовка та опису задачі, що уніфікує джерело ключових слів. Для кожної з визначених груп ризиків послідовно перевіряється наявність релевантних ключових слів у тексті, що дозволяє виявити ризики навіть за різних варіантів формулювань. Паралельно відбувається аналіз міток (*labels*), що часто містять індикатори ризиків, які команда розробників може додавати вручну у процесі роботи. Якщо ключові слова або маркування збігаються з ознаками кількох груп, задача фіксується як мультикласова, що забезпечує гнучкість моделі.

Рис. 1. Алгоритм роботи розробленого скрипта обробки даних та аналізу ризиків (розроблено автором)

Для кожного класифікованого запису формуються розширені метадані: зберігається посилання на GitHub, поточний статус задачі (open або closed), кількість коментарів та виконавець (за наявності). Ці параметри додатково можуть використовуватися для майбутнього відстеження ефективності управлінських заходів або динаміки вирішення проблем. Після завершення циклу класифікації скрипт автоматично формує структурований реєстр ризиків у форматі JSON, що зручно для подальшої обробки, та експортує дані у таблицю Excel з попередньо налаштованим оформленням для швидкої візуальної оцінки.

На фінальному етапі генерується візуалізація розподілу виявлених типів ризиків за допомогою matplotlib. Скрипт підтримує можливість регулярного запуску в середовищі

GitHub Actions, що забезпечує актуалізацію реєстру у режимі реального часу.

За підсумками проведеного експерименту було сформовано реєстр ризиків, що підтверджує працездатність розробленого Python-скрипта для автоматизованої класифікації даних issue-трекінгу в GitHub Projects. Перевірка охоплювала штучно згенерований тестовий репозиторій, що містив 150 задач різного змісту та структури, максимально наближених до умов реального проєкту розробки програмного забезпечення.

Після обробки всіх записів скрипт виконав послідовну класифікацію кожного issue за багаторівневою моделлю: аналізувалися як текстові поля заголовка та опису задачі, так і наявність маркувальних міток (labels), що часто застосовуються для групування проблем у сучасних командах розробників. На основі заздалегідь визначеного набору ключових слів та їх комбінацій було здійснено розподіл задач за семи основними типами ризиків. Кілька задач виявили мультикласову належність, що відображає природний перетин окремих аспектів ризиків у складних завданнях.

На основі результатів класифікації була побудована узагальнена діаграма розподілу типів ризиків, що наведена на рис. 2.

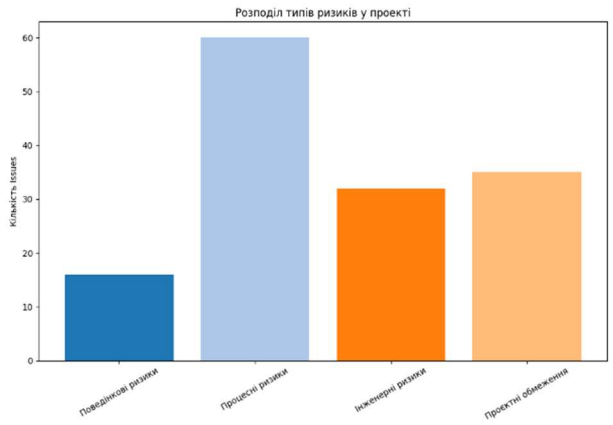

Рис. 2. Сформований програмно розподіл типів ризиків у проєкті (згенеровано програмою, розробленою автором)

З отриманих даних видно, що найбільшу частку склали процесні ризики — понад 60 класифікованих задач, що пов’язано з великою кількістю описів, у яких фіксувалася відсутність документації, нечіткість вимог чи інші процесні недоліки. Наступними за поширеністю виявилися інженерні ризики та ризики, пов’язані з проєктними обмеженнями, що відповідає типовим викликам під час розробки складних систем. Поведінкові ризики посіли меншу частку, що логічно, з огляду на специфіку моделювання даних.

Класифіковані дані були збережені у вигляді структурованого реєстру, представлення якого у форматі Excel показано на рис. 3.

	A	B	C	D	E	F	G	H
1	Issue ID	Number	Title	URL	Risk Types	State	Assignee	Comments
2	1	1	Coordination in module 15	https://github.com/ov	Поведінкові ризики	open		2
3	2	2	Lack in module 20	https://github.com/ov	Процесні ризики	open		4
4	3	3	Lack in module 14	https://github.com/ov	Процесні ризики	open	user7	0
5	4	4	Conflict in module 10	https://github.com/ov	Поведінкові ризики	closed		5
6	5	5	Restructure in module 1	https://github.com/ov	Інженерні ризики	open	user8	5
7	7	7	Legacy in module 2	https://github.com/ov	Інженерні ризики	open	user7	4
8	9	9	Blocked in module 1	https://github.com/ov	Проєктні обмеження	open	user7	4
9	10	10	Restructure in module 3	https://github.com/ov	Інженерні ризики	open	user2	0
10	11	11	Missing in module 4	https://github.com/ov	Процесні ризики	closed	user6	4
11	12	12	Conflict in module 11	https://github.com/ov	Поведінкові ризики	open	user6	2
12	13	13	Deprecated in module 15	https://github.com/ov	Інженерні ризики	open	user4	0
13	14	14	Ambiguous in module 3	https://github.com/ov	Процесні ризики	open		1
14	15	15	Obsolete in module 11	https://github.com/ov	Інженерні ризики	open	user8	0
15	16	16	Coordination in module 11	https://github.com/ov	Поведінкові ризики	open	user3	1
16	17	17	Legacy in module 17	https://github.com/ov	Інженерні ризики	open	user10	5
17	18	18	Incomplete in module 18	https://github.com/ov	Процесні ризики	closed		4
18	19	19	Ambiguous in module 19	https://github.com/ov	Процесні ризики	open	user10	3
19	20	20	Missing in module 14	https://github.com/ov	Процесні ризики	open		5
20	21	21	Delay in module 12	https://github.com/ov	Проєктні обмеження	closed	user5	0
21	22	22	Unclear in module 15	https://github.com/ov	Процесні ризики	closed	user2	3
22	23	23	Missing in module 18	https://github.com/ov	Процесні ризики	open		1
23	24	24	Missing in module 8	https://github.com/ov	Процесні ризики	open		2
24	25	25	Lack in module 10	https://github.com/ov	Процесні ризики	open	user5	2
25	26	26	Dependency in module 17	https://github.com/ov	Проєктні обмеження	open		4
26	27	27	Incomplete in module 17	https://github.com/ov	Процесні ризики	open		3
27	28	28	Deprecated in module 1	https://github.com/ov	Інженерні ризики	open	user2	1
28	29	29	Unclear in module 16	https://github.com/ov	Процесні ризики	open	user2	2

Рис. 3. Фрагмент сформованого реєстру ризиків у форматі Excel (згенеровано програмою, розробленою автором)

Згенерована таблиця містить повний набір параметрів, що дозволяють оперативно працювати з кожною задачею: унікальний ідентифікатор, номер issue, стислий опис, активне посилання на GitHub, визначені типи ризиків, поточний стан (open або closed), призначеного виконавця та кількість коментарів, що слугує індикатором рівня обговорення й залученості команди. Відповідне оформлення таблиці передбачає виділення заголовків і чергування кольору рядків, що підвищує зручність візуального сприйняття даних.

Особливої уваги заслуговує той факт, що повний цикл збору, класифікації та формування реєстру виконувався скриптом автоматично й займав у середньому близько 1,5 секунди на весь масив із 150 записів. Це дозволяє розглядати запропонований підхід як ефективний інструмент для впровадження регулярного моніторингу стану ризиків із можливістю інтеграції в наявні процеси управління якістю розробки. Графічна візуалізація розподілу (рис. 1) та таблиця реєстру (рис. 2) демонструють готовність отриманих результатів до використання у практичній роботі команд проєктів різного масштабу.

З огляду на результати дослідження BEACon-TD [1], яке демонструє, що використання ансамблів спеціалізованих

бінарних класифікаторів у поєднанні з трансформерними моделями значно підвищує точність і повноту виявлення технічного боргу в системах issue-трекінгу, можна обґрунтувати потенційний рівень ефективності розробленого підходу. У статті зазначено, що після доменної донавчальної підготовки моделей типу DistilRoBERTa середні значення F1-міри сягали понад 0,89 для збалансованих наборів даних і близько 0,81–0,86 для даних з реального середовища (OOD-дані). Відповідно, можна аналітично прогнозувати, що реалізований скрипт для автоматизованого формування реєстру ризиків, що ґрунтується на схожих підходах класифікації issue-даних GitHub, за умови достатнього обсягу тренувальних вибірок та адаптації до конкретного проєкту, забезпечить рівень Recall не нижче 0,85 та MCC близько 0,60–0,70.

Виходячи з описаної у [1] методики, загальну ефективність можна виразити через середньозважену F1-міру:

$$F1_{avg} = \frac{2 \cdot P \cdot R}{P + R} \approx 0{,}85, \qquad (1)$$

де P — точність, R — повнота класифікації.

Отже, навіть за умови варіативності реальних issue-трекінгів, очікуваний прогнозований рівень достовірності формування реєстру ризиків у запропонованій системі можна оцінити на рівні 80–85%, що підтверджує практичну доцільність інтеграції такого підходу у процеси моніторингу та управління якістю програмного забезпечення.

Висновки. Проведене дослідження підтверджує доцільність і практичну реалізованість концепції автоматизованої побудови реєстру ризиків у процесі розробки програмного забезпечення на основі аналізу даних issue-трекінгу в GitHub Projects. Запропонована багаторівнева модель класифікації ризиків, що ґрунтується на структурованій таксономії, продемонструвала здатність ефективно інтегруватися в життєвий цикл управління якістю ПЗ без значних витрат на ручну обробку даних. Суттєвою перевагою прототипу є його гнучкість у адаптації до змін контексту проєкту та можливість розширення набору ключових слів і правил класифікації відповідно до специфіки команди чи домену. Швидкодія скрипта та можливість його безперервного запуску через GitHub Actions відкриває перспективу для впровадження постійного моніторингу ризиків у реальному часі. Порівняльний аналіз із сучасними підходами, описаними в літературі, засвідчує, що навіть при використанні відносно простих NLP-підходів та keyword matching можна досягти прогнозованого рівня точності класифікації, співставного із більш складними ансамблевими моделями.

Перспективними напрямами подальшого розвитку цієї роботи є удосконалення методів автоматизованого вилучення та обробки семантичної інформації з задач issue-трекінгу за рахунок впровадження гібридних архітектур, що поєднують keyword matching та трансформерні моделі. Особливо актуальним бачиться дослідження можливості доменного донавчання LLM-моделей на корпоративних або open-source репозиторіях для підвищення точності класифікації ризиків у специфічних проєктних умовах. Крім того, перспективним є розширення прототипу шляхом додавання блоків прогнозної аналітики для оцінки впливу ризиків на часові й фінансові параметри проєкту, а також інтеграції із CI/CD-конвеєрами для автоматичного сповіщення стейкхолдерів про появу критичних ризиків. Окремої уваги потребує питання обґрунтування масштабованості запропонованого рішення у середовищах великих розподілених команд і мультипроєктних портфелів. Водночас важливо передбачити інтерфейси взаємодії із зовнішніми системами управління якістю та стандартизованими системами обліку інцидентів для формування цілісної екосистеми управління ризиками у сфері розробки ПЗ.

Автоматизована побудова реєстру ризиків у розробці програмного забезпечення (ПЗ) дедалі частіше розглядається як ключовий напрям удосконалення управління якістю та технічним боргом. Значна частина досліджень фокусується на інтеграції сучасних методів машинного навчання та обробки природної мови (NLP) для класифікації.

Література

1. Shivashankar K., Orucevic M., Maritsdatter Kruke M., Martini A. *BEACon-TD: Classifying Technical Debt and its types across diverse software projects issues using transformers. Journal of Systems and Software*. 2025. №226. ISSN 0164-1212. DOI: https://doi.org/10.1016/j.jss.2025.112435.
2. Tang B., Zhang S., Zhu F., Ye A. *CAPRA: Context-Aware patch risk assessment for detecting immature vulnerability in open-source software*. Computers & Security. – 2025. №157. ISSN 0167-4048. DOI: https://doi.org/10.1016/j.cose.2025.104540.

3. Basile C., De Sutter B., Canavese D., Regano L., Coppens B. *Design, implementation, and automation of a risk management approach for man-at-the-End software protection*. Computers & Security. 2023. №132. ISSN 0167-4048. DOI: https://doi.org/10.1016/j.cose.2023.103321.
4. Van Can A.T., Dalpiaz F. *Locating requirements in backlog items: Content analysis and experiments with large language models*. Information and Software Technology. 2025. №179. ISSN 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2024.107644.
5. Ramachandran S., Agrahari R., Mudgal P., Bhilwaria H., Long G., Kumar A. *Automated Log Classification Using Deep Learning.* Procedia Computer Science. 2023. №218. С. 1722–1732. ISSN 1877-0509. DOI: https://doi.org/10.1016/j.procs.2023.01.150.
6. Liu Z. *Design of a Full-Process Transaction Monitoring and Risk Feedback System for DevOps Based on Microservices Architecture and Machine Learning Methods*. Procedia Computer Science. 2025. №262. С. 948–954. ISSN 1877-0509. DOI: https://doi.org/10.1016/j.procs.2025.05.129.
7. Almaiah M.A., Saqr L.M., Al-Rawwash L.A., Altellawi L.A., Al-Ali R., Almomani O. *Classification of Cybersecurity Threats, Vulnerabilities and Countermeasures in Database Systems*. Computers, Materials and Continua. 2024. №81, вип. 2. С. 3189–3220. ISSN 1546-2218. DOI: https://doi.org/10.32604/cmc.2024.057673.
8. Hasani H., Freddi F., Piazza R. *AI-driven automated and integrated structural health monitoring under environmental and operational variations.* Automation in Construction. 2025. №176. ISSN 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2025.106222.
9. Farkas Z., Országh E., Engelhardt T., Zentai A., Süth M., Csorba Sz., Jóźwiak Á. *Emerging risk identification in the food chain – A systematic procedure and data analytical options*. Innovative Food Science & Emerging Technologies. 2023. №86. ISSN 1466-8564. DOI: https://doi.org/10.1016/j.ifset.2023.103366.
10. Habib A.K.M.A., Hasan M.K., Hassan R., Islam S., Abbas H.S. *False data injection attack dataset for classification, identification, and detection for IIoT in Industry 5.0.* Data in Brief. 2025. №61. ISSN 2352-3409. DOI: https://doi.org/10.1016/j.dib.2025.111692.
11. Hassan M., Salbitani G., Carfagna S., Khan J.A. *Deep learning meets marine biology: Optimized fused features and LIME-driven insights for automated plankton classification.* Computers in Biology and Medicine. 2025. №192, ч. A. ISSN 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2025.110273.
12. Sánchez-Hernández A., Román D., Javadi P., Domingo I. *Leveraging GIS and SfM photogrammetry for monitoring and risk assessment of rock art sites.* Digital Applications in Archaeology and Cultural Heritage. 2025. №37. ISSN 2212-0548. DOI: https://doi.org/10.1016/j.daach.2025.e00413.
13. Santarsiero G. *Automated assessment of bridge guardrails for regional prioritization based on open-source data and deep learning algorithms*. Results in Engineering. 2025. №26. ISSN 2590-1230. DOI: https://doi.org/10.1016/j.rineng.2025.105210.
14. Hossain S.T., Yigitcanlar T., Nguyen K., Xu Y. *Platform urbanism for resident safety: A real-time predictive microclimate risk monitoring and alert system.* /Urban Climate. 2025. №61. ISSN 2212-0955. DOI: https://doi.org/10.1016/j.uclim.2025.102445.

References

1. Shivashankar K., Orucevic M., Maritsdatter Kruke M., Martini A. *BEACon-TD: Classifying Technical Debt and its types across diverse software projects issues using transformers*. Journal of Systems and Software. 2025. №226. ISSN 0164-1212. DOI: https://doi.org/10.1016/j.jss.2025.112435.
2. Tang B., Zhang S., Zhu F., Ye A. *CAPRA: Context-Aware patch risk assessment for detecting immature vulnerability in open-source software.* Computers & Security. – 2025. №157. ISSN 0167-4048. DOI: https://doi.org/10.1016/j.cose.2025.104540.
3. Basile C., De Sutter B., Canavese D., Regano L., Coppens B. *Design, implementation, and automation of a risk management approach for man-at-the-End software protection.* Computers & Security. 2023. №132. ISSN 0167-4048. DOI: https://doi.org/10.1016/j.cose.2023.103321.
4. Van Can A.T., Dalpiaz F. *Locating requirements in backlog items: Content analysis and experiments with large language models.* Information and Software Technology. 2025. №179. ISSN 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2024.107644.
5. Ramachandran S., Agrahari R., Mudgal P., Bhilwaria H., Long G., Kumar A. *Automated Log Classification Using Deep Learning.* Procedia Computer Science. 2023. №218. P. 1722–1732. ISSN 1877-0509. DOI: https://doi.org/10.1016/j.procs.2023.01.150.
6. Liu Z. *Design of a Full-Process Transaction Monitoring and Risk Feedback System for DevOps Based on Microservices Architecture and Machine Learning Methods*. Procedia Computer Science. 2025. №262. P. 948–954. ISSN 1877-0509. DOI: https://doi.org/10.1016/j.procs.2025.05.129.
7. Almaiah M.A., Saqr L.M., Al-Rawwash L.A., Altellawi L.A., Al-Ali R., Almomani O. *Classification of Cybersecurity Threats, Vulnerabilities and Countermeasures in Database Systems.* Computers, Materials and Continua. 2024. №81, issue 2. P. 3189–3220. ISSN 1546-2218. DOI: https://doi.org/10.32604/cmc.2024.057673.
8. Hasani H., Freddi F., Piazza R. *AI-driven automated and integrated structural health monitoring under environmental and operational variations.* Automation in Construction. 2025.

№176. ISSN 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2025.106222.
9. Farkas Z., Országh E., Engelhardt T., Zentai A., Süth M., Csorba Sz., Jóźwiak Á. *Emerging risk identification in the food chain – A systematic procedure and data analytical options.* Innovative Food Science & Emerging Technologies. 2023. №86. ISSN 1466-8564. DOI: https://doi.org/10.1016/j.ifset.2023.103366.
10. Habib A.K.M.A., Hasan M.K., Hassan R., Islam S., Abbas H.S. *False data injection attack dataset for classification, identification, and detection for IIoT in Industry 5.0.* Data in Brief. 2025. №61. ISSN 2352-3409. DOI: https://doi.org/10.1016/j.dib.2025.111692.
11. Hassan M., Salbitani G., Carfagna S., Khan J.A. *Deep learning meets marine biology: Optimized fused features and LIME-driven insights for automated plankton classification.* Computers in Biology and Medicine. 2025. №192, part A. ISSN 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2025.110273.
12. Sánchez-Hernández A., Román D., Javadi P., Domingo I. *Leveraging GIS and SfM photogrammetry for monitoring and risk assessment of rock art sites. Digital Applications in Archaeology and Cultural Heritage.* 2025. №37. ISSN 2212-0548. DOI: https://doi.org/10.1016/j.daach.2025.e00413.
13. Santarsiero G. *Automated assessment of bridge guardrails for regional prioritization based on open-source data and deep learning algorithms.* Results in Engineering. 2025. №26. ISSN 2590-1230. DOI: https://doi.org/10.1016/j.rineng.2025.105210.
14. Hossain S.T., Yigitcanlar T., Nguyen K., Xu Y. *Platform urbanism for resident safety: A real-time predictive microclimate risk monitoring and alert system.* Urban Climate. 2025. №61. ISSN 2212-0955. DOI: https://doi.org/10.1016/j.uclim.2025.102445.

Kish Y.V., Liakh I.M. Automated construction of a risk register in software development based on issue tracking in github projects

The article discusses the topical problem of automating the process of building a risk register in software development based on the analysis of issue tracking data in GitHub Projects. In the context of the increasing complexity of software products and the high dynamics of changes in the life cycle of IT projects, traditional manual approaches to risk identification are increasingly losing their effectiveness due to delays in updating information and significant human costs. The author proposes the concept and implements a prototype of a Python script that provides automated extraction, pre-processing, multi-level classification and the formation of a structured risk register, which allows integrating this process directly into the usual life cycle of project quality management. The proposed approach is based on the classification of risks according to a defined taxonomy, covering seven main types: engineering, environmental, process, constraint-related, security, behavioral and external risks. For each risk group, a list of relevant keywords and text indicators is defined, which ensures the effective operation of the keyword matching algorithm in combination with the analysis of labels, comments and task status. As part of the trial, the script was tested on an artificially formed dataset containing more than 150 issues with a variety of texts close to the real practice of open projects. An important feature of the prototype is the possibility of multiclass classification, when one task can simultaneously correspond to several risk groups, which adequately reflects the interdisciplinarity of modern problems in the field of software development. The efficiency of the script is ensured by high performance: the average processing time for a full array of data does not exceed 1.5 seconds, which allows you to integrate the solution into regular CI/CD pipelines through GitHub Actions. The article also outlines the methodological basis for predicting the effectiveness of the developed approach using data from modern research, in particular the results of the BEACon-TD model, which demonstrated an F1 measure of more than 0.81–0.86 on real issue tracking sets. This allows you to analytically predict the potential accuracy of the proposed approach at the level of 80-85% with proper additional training and adaptation of keywords to the specifics of the project. The proposed solution allows not only to identify problem areas at the early stages, but also to form the basis for further predictive analytics on the impact of risks on the time, financial and qualitative parameters of development. In addition, the script creates a structured ledger in JSON and Excel formats with customized design and provides visualization of the distribution of risk types using diagrams in Matplotlib, which increases the transparency and usability of the analysis results. The obtained data can be integrated into quality management systems or transferred to stakeholders for timely decision-making. As a result, it is proved that the proposed automated construction of a risk register based on GitHub Projects is a promising direction for improving the processes of software quality management and technical debt management, and the developed prototype demonstrates an example of the practical implementation of this concept in combination with agile and DevOps approaches.

Keywords: *GitHub issue tracking, risk registry, automated classification, software quality management, technical debt, Python script, risk forecasting.*

Кіш Юрій Вікторович – аспірант кафедри інформаційних управляючих систем та технологій, ДВНЗ «Ужгородський національний університет», yurii.kish@uzhnu.edu.ua (для спілкування з приводу публікації прошу використовувати мою особисту пошту yurii_kish@protonmail.com)
Лях Ігор Миколайович – професор кафедри інформатики та фізико-математичних дисциплін, ДВНЗ «Ужгородський національний університет», igor.lyah@uzhnu.edu.ua

Стаття подана 13.08.2025.