МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМЕНІ С. З. ҐЖИЦЬКОГО

Факультет менеджменту, бізнесу та публічного адміністрування

Кафедра менеджменту
ІТ-сфери

Допускається до захисту
“_____”_____2025 р.
В.о. зав. кафедри _____
(підпис)
к.е.н., доцент Олена КІНДРАТ
наук. ступ., вчене звання ім’я та прізвище

КВАЛІФІКАЦІЙНА РОБОТА

студента **Гапоненка Максима Павловича**

на тему:

«Створення та реалізація технічного завдання ІТ-проєктів»

на присвоєння кваліфікації – *Магістр з менеджменту ІТ-сфери*

Керівник роботи _____к.ф.-м.н., доцент СТЕПАНЮК О.І.__
(підпис) (вчене звання, прізвище та ініціали)

Львів – 2025

Львівський національний університет ветеринарної медицини та біотехнологій імені С.З. Ґжицького

Факультет	Менеджменту, бізнесу та публічного адміністрування
Кафедра	Менеджменту ІТ-сфери
Галузь знань	07 «Управління та адміністрування»
Спеціальність	073 «Менеджмент»
Освітньо-професійна програма	«Менеджмент ІТ-сфери»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

_____

“____”_____20___року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ
Гапоненку Максиму Павловичу

Тема роботи

«Створення та реалізація технічного завдання ІТ-проєктів»

керівник роботи Степанюк Олександр Іванович, к.ф.-м.н., доцент _____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «___»_____2025 року №___

2. Строк подання здобувачем роботи_____

3. Вихідні дані до роботи: навчально-методична, наукова та довідкова література, що відповідає темі кваліфікаційної роботи, фінансова звітність та аналітичні матеріали компанії *****.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ТЕХНІЧНОГО ЗАВДАННЯ В DEVOPS-ПРОЄКТАХ. 1.1. Роль технічного завдання у DevOps-процесах. 1.2. Стандарти та методики розроблення вимог. 1.3. Особливості вимог до систем CI/CD та інструментів безпеки. 1.4. Ризики й виклики під час формування технічного завдання ІТ-проєктів. РОЗДІЛ 2. . АНАЛІЗ ОРГАНІЗАЦІЇ РЕАЛІЗАЦІЇ ПРОЄКТІВ У КОМПАНІЇ *****. 2.1. Організаційно-управлінські засади діяльності *****. 2.2. Процеси планування та ініціації проєктів у *****. 2.3. Управління розробленням і реалізацією DevOps-проєктів. 2.4 Оцінювання ефективності моделі реалізації проєктів у *****. РОЗДІЛ 3. РОЗРОБЛЕННЯ ТА ВПРОВАДЖЕННЯ

ТЕХНІЧНОГО ЗАВДАННЯ МОДУЛЯ AI-ANALYZER ДЛЯ ***** CI/CD. 3.1 Опис проєкту та визначення його ключових можливостей. 3.2. Формування вимог до модуля AI-Analyzer. 3.3. Архітектура рішення та технічна реалізація. 3.4. Оцінювання результатів упровадження модуля та напрями вдосконалення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень).
1. Особливості, що роблять ***** привабливим для користувачів. 2. Організаційна структура ***** (керівна команда E-Group). 3. Динаміка доходів ***** за 2022-2024 рр. 4. Динаміка чистого прибутку (збитку), млн. дол. 5 Процес ініціації проєкту в *****. 6. DevOps Toolchain.

6. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання. Вивчення рекомендованої літератури по темі КР. Вивчення об'єкту дослідження.	Травень - червень	виконано
2	Розділ 1. ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ТЕХНІЧНОГО ЗАВДАННЯ В DEVOPS-ПРОЄКТАХ	Липень	виконано
3	Розділ 2. АНАЛІЗ ОРГАНІЗАЦІЇ РЕАЛІЗАЦІЇ ПРОЄКТІВ У КОМПАНІЇ *****	Серпень	виконано
4	Розділ 3. РОЗРОБЛЕННЯ ТА ВПРОВАДЖЕННЯ ТЕХНІЧНОГО ЗАВДАННЯ МОДУЛЯ AI-ANALYZER ДЛЯ ***** CI/CD	Вересень	виконано
5	Кінцеве оформлення кваліфікаційної роботи. Підготовка до захисту. Пробний захист на випусковій кафедрі.	Жовтень-листопад	виконано

Здобувач _____ Гапоненко М. П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Степанюк О.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Гапоненко М. П. «Створення та реалізація технічного завдання ІТ-проєктів»: Кваліфікаційна робота: (073 «Менеджмент») / ЛНУВМБ імені С. З. Ґжицького. Кафедра менеджменту ІТ-сфери. Наук. кер.: О.І. Степанюк, к.ф-м.н., доцент. Львів, 2025. 74 с.

У першому розділі кваліфікаційної роботи досліджено теоретичні засади формування технічного завдання в управлінні ІТ-проєктами. Розкрито сутність, структуру та роль технічного завдання як базового інструмента організації розроблення програмних продуктів. Проаналізовано сучасні підходи, стандарти та методології управління вимогами, зокрема положення ISO/IEC/IEEE 29148, BABOK і PMBOK. Висвітлено методи збору, аналізу та формалізації вимог, охарактеризовано ризики та проблеми, що виникають під час створення технічної документації в ІТ-компаніях.

У другому розділі виконано ґрунтовний аналіз організації реалізації проєктів у компанії *****. Подано характеристику організаційно-управлінської моделі, заснованої на повністю віддаленому форматі роботи та відкритих процесах. Розглянуто особливості планування й ініціації проєктів, включно з механізмами Problem Validation і Solution Proposal. Проаналізовано моделі розроблення та взаємодії команд у середовищі DevOps, зокрема процеси CI/CD, code review, інтеграцію механізмів контролю якості та безпеки. Визначено ключові чинники ефективності організації проєктної діяльності та проблемні аспекти, що впливають на узгодженість вимог і стабільність процесів розроблення.

У третьому розділі розроблено технічне завдання та архітектурне рішення для модуля AI-Analyzer, інтегрованого з ***** CI/CD. Сформульовано функціональні й нефункціональні вимоги до системи, визначено особливості побудови ML-модуля та його взаємодії з pipeline-структурою *****. Здійснено оцінювання ефективності розробленого рішення та запропоновано напрями подальшої оптимізації, спрямовані на підвищення точності аналізу, покращення інтеграційних механізмів та удосконалення DevSecOps-процесів.

Робота містить 3 розділи, висновки, список використаних джерел і додатки. Містить 6 рисунків, 10 таблиць, 4 додатки, викладена на 74 сторінках.

Ключові слова: технічне завдання, управління вимогами, DevOps, *****, CI/CD, AI-модуль, DevSecOps, менеджмент ІТ-проєктів.

ANNOTATION

Haponenko M. P. “Development and Implementation of the Technical Specification for IT Projects”: Qualification Thesis (073 “Management”) / Stepan Gzhytskyi National University of Veterinary Medicine and Biotechnologies. Department of IT Management. Supervisor: O. I. Stepaniuk, PhD in Physical and Mathematical Sciences, Associate Professor. Lviv, 2025. 74 p.

The first chapter of the qualification thesis examines the theoretical foundations of developing a technical specification within IT project management. The essence, structure, and role of the technical specification as a fundamental instrument for organizing software development are revealed. Modern approaches, standards, and methodologies of requirements management are analyzed, including ISO/IEC/IEEE 29148, BABOK, and PMBOK. The chapter highlights methods of collecting, analyzing, and formalizing requirements, as well as risks and challenges associated with creating technical documentation in IT companies.

The second chapter provides a comprehensive analysis of project implementation processes in *****. It presents the organizational and managerial model based on a fully remote work environment and transparent operational practices. The chapter explores project planning and initiation processes, including mechanisms of Problem Validation and Solution Proposal. Development workflows and team interactions within the DevOps ecosystem are examined, with particular attention to CI/CD processes, code review practices, and the integration of quality assurance and security mechanisms. Key factors influencing the effectiveness of project organization and the challenges affecting requirement alignment and development stability are identified.

The third chapter develops the technical specification and architectural design for the AI-Analyzer module integrated into ***** CI/CD. Functional and non-functional requirements are formulated, and the structure of the ML-based module is defined, including its interaction with *****’s pipeline environment. The effectiveness of the proposed solution is evaluated, and directions for further optimization are provided, focusing on accuracy improvement, enhancement of integration mechanisms, and refinement of DevSecOps processes.

The thesis includes 3 chapters, conclusions, a list of references, and appendices. It contains 6 figures, 10 tables, 4 appendices, and is presented on 74 pages.

Keywords: technical specification, requirements management, DevOps, *****, CI/CD, AI module, DevSecOps, IT project management.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. З огляду на стрімкий розвиток цифрових технологій та зростання вимог до ефективності, надійності й масштабованості програмних рішень, тема створення та реалізації технічного завдання IT-проєктів набуває особливої актуальності. Сучасна IT-індустрія характеризується високою динамічністю, інтенсивною конкуренцією та необхідністю швидкої адаптації до мінливих умов ринку. У таких умовах якісно сформоване технічне завдання відіграє ключову роль у забезпеченні успішного проєктування, розроблення й впровадження програмних продуктів.

IT-проєкти стають дедалі складнішими внаслідок інтеграції інноваційних технологій, таких як штучний інтелект, машинне навчання, хмарні сервіси, великі дані та DevOps-підходи до організації розроблення. Управління такими проєктами потребує використання спеціалізованих методологій та стандартизованих підходів до формування вимог. Саме тому у роботі розглядаються сучасні стандарти, що регламентують процеси управління вимогами, зокрема ISO/IEC/IEEE 29148, PMBOK та BABOK.

Реалізація IT-проєктів нерідко супроводжується такими викликами, як зміна бізнес-пріоритетів, поява нових технічних обмежень, необхідність швидкої інтеграції модулів та високі вимоги до безпеки й якості. Основними причинами невдач у реалізації IT-проєктів залишаються:

– недостатня опрацьованість етапу збору та аналізу вимог, що призводить до помилок у плануванні та оцінюванні ресурсів;

– відсутність гнучкості у підходах до реалізації через складність технічних рішень;

– перевантаженість менеджерів великою кількістю завдань та високою швидкістю змін;

– неузгодженість відповідальності між учасниками проєктної команди.

Успішний процес управління IT-проєктом має враховувати всі фактори

складності та застосовувати спеціалізовані інструменти для формування технічного завдання, що слугує основою для подальших етапів створення продукту. Важливим аспектом є чіткий розподіл ролей, контроль вимог та забезпечення прозорої комунікації між усіма учасниками проєктного процесу.

Якісно розроблене технічне завдання забезпечує підвищення конкурентоспроможності компаній, скорочення строків виходу продуктів на ринок, удосконалення взаємодії з користувачами та створення інноваційних рішень, які відповідають сучасним викликам. Особливо актуальним є дослідження таких процесів на прикладі компаній, що працюють у моделі DevOps та відкритого середовища розроблення.

Проблематика організації процесів формування технічного завдання та управління вимогами висвітлювалася у роботах таких дослідників, як Ю. В. Шулик, О. І. Качан [15], А. В. Катренко [3], О. А. Сметанюк, Н. М. Данилюк, А. В. Бондарчук [37], Г. В. Озимок, О. В. Колянко [29], Н. Шашкова, І. Фадєєва, Т. Казакова [41] та ін. Серед зарубіжних авторів значний внесок здійснили П. Роберт, К. Хасс, Л. Кроуфорд, К. Бізнер, Б. Хоббс, Т. ДеМарко, А. Денніс, Г. Альтман.

Метою роботи є дослідження процесу створення та реалізації технічного завдання ІТ-проєктів, аналіз методологічних підходів управління вимогами та розроблення технічного завдання для модуля AI-Analyzer у середовищі ***** CI/CD.

Об’єктом дослідження є процеси формування технічних вимог та організації реалізації ІТ-проєктів.

Предметом дослідження виступають методи, моделі та інструменти розроблення технічного завдання в ІТ-компаніях.

Базою дослідження обрано компанію ***** та проєкт зі створення модуля AI-Analyzer, призначеного для автоматизованого аналізу якості та безпеки коду у CI/CD-процесах.

Інформаційну базу дослідження становлять наукові праці, навчальні посібники, нормативні документи, міжнародні стандарти, аналітичні звіти, відкриті матеріали ***** Handbook, технічна документація *****, а також ресурси мережі Інтернет.

У процесі виконання дослідження застосовано методи аналізу та синтезу, порівняння й аналогій, систематизації, узагальнення, а також метод дослідження документації. Використання зазначених підходів дало змогу комплексно розглянути питання формування технічного завдання та обґрунтувати отримані висновки.

За результатами проведеного аналізу сформульовано висновки й рекомендації, спрямовані на підвищення ефективності процесів створення та реалізації технічного завдання у сфері ІТ-проєктів.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ТЕХНІЧНОГО ЗАВДАННЯ В DEVOPS-ПРОЄКТАХ

1.1 Роль технічного завдання у DevOps-процесах

У сучасному високотехнологічному світі надання високоякісного програмного забезпечення та застосунків із більшою швидкістю та гнучкістю стало необхідністю часу. [2]

Щоб досягти такої гнучкості та ефективності, підприємствам необхідно впровадити нові інструменти та інноваційні методології, такі як DevOps.

DevOps розглядається як підхід до організації розроблення програмного забезпечення, спрямований на об’єднання роботи команд розробників і фахівців з експлуатації з метою забезпечення безперервності поставок, високої якості продукту та скорочення часу від створення функціоналу до його розгортання у виробничому середовищі. [3] Згідно з усталеними практиками галузі, DevOps не є окремою технологією чи інструментом – це комплекс методів, що поєднує автоматизацію, прозору комунікацію, контроль змін, тестування та інфраструктурне управління як єдиний безперервний процес. Його характерними рисами є висока частота релізів, використання CI/CD-конвеєрів, глибока інтеграція тестування у всі етапи життєвого циклу продукту та активне застосування інструментів моніторингу й автоматизації.

У DevOps-середовищі вирішальне значення має уніфікація вимог, оскільки будь-які зміни у функціоналі або інфраструктурі впливають на роботу CI/CD-процесів, тестів, систем безпеки та середовищ розгортання. Саме тому технічне завдання (ТЗ) виступає ключовим документом, що забезпечує узгодженість між усіма учасниками процесу, а також слугує основою для планування, автоматизації та перевірки якості. [4]

Технічне завдання у DevOps-процесах виконує низку важливих функцій. Воно визначає очікувану поведінку системи, формує вимоги до

функціональності, продуктивності, надійності та безпеки, які мають бути враховані під час розроблення. Саме на основі ТЗ будуються автоматизовані сценарії тестування, налаштовується статичний аналіз коду та формуються правила контролю відповідності програмного компонента технічним критеріям.

Технічне завдання дозволяє:

- – встановити умови переходу функціоналу між середовищами (staging → production);
- – визначити ризики та залежності для релізного менеджменту;
- – сформувати правила керування версіями та сумісністю компонентів;
- – стандартизувати інтеграційні вимоги між сервісами.

У контексті DevOps ТЗ забезпечує передбачуваність та контрольованість змін: кожна функція або модуль, що проходить через CI/CD-pipeline, має бути однозначно описаний, включно з критеріями готовності, залежностями, особливостями інтеграції та ризиками. Чітке формулювання вимог також полегшує комунікацію між командами розроблення й експлуатації, що є критично важливим для швидкого прийняття рішень та зменшення часу на усунення помилок.

Особливу роль технічне завдання відіграє у DevSecOps-підході, де вимоги безпеки розглядаються як складова частина всіх етапів створення продукту. ТЗ фіксує необхідні механізми захисту, політики доступу, правила обробки даних та вимоги до аудиту. Це дозволяє інтегрувати перевірки безпеки у CI/CD-потоки й автоматизувати їх виконання.

Таким чином, у DevOps-процесах технічне завдання є не лише документом для команди розроблення. Воно виконує роль системної основи для автоматизації тестування, забезпечення якості, узгодженості дій різних команд, підтримки стабільності релізів та дотримання вимог безпеки. Чітке та структуроване ТЗ значно підвищує рівень керованості ІТ-проєкту, зменшує ймовірність помилок та

сприяє успішному досягненню цілей у середовищах із високою швидкістю змін і складністю інфраструктури.

1.2 Стандарти та методики розроблення вимог

Розроблення вимог є ключовим етапом життєвого циклу програмних систем і визначається як систематизований процес виявлення, документування, аналізу та узгодження потреб користувачів і бізнес-замовників. Якість вимог безпосередньо впливає на терміни, бюджет, архітектуру та результати IT-проєктів. Для підвищення точності та структурованості вимог використовуються міжнародні стандарти, методики та підходи, які регламентують процеси requirements engineering.

Стандарт ISO/IEC/IEEE 29148:2018 є одним із найбільш визнаних документів, що систематизує вимоги до процесу формування, оцінювання й документування вимог. [6] Він визначає:

– класифікацію вимог (бізнес-вимоги, користувацькі, функціональні, нефункціональні);

– правила перевірки вимог (correctness, feasibility, verifiability);

– принципи формування якості вимог (єдність, однозначність, повнота).

ISO/IEC 12207:2017 Software Life Cycle Processes – стандарт, що встановлює загальну структуру життєвого циклу програмного забезпечення та визначає вимоги до процесів аналізу, розроблення, тестування й супроводу. Він також регламентує місце управління вимогами в загальному контексті життєвого циклу. [11]

Стандарт BABOK (Business Analysis Body of Knowledge), розроблений Міжнародним інститутом бізнес-аналізу (IIBA), є однією з найкомплексніших методологій, що описує процеси бізнес-аналізу та вимоготехніки у сучасних IT-проєктах. У межах цього стандарту систематизовано підходи до виявлення потреб стейкхолдерів, документування та моделювання вимог, а також

оцінювання запропонованих рішень. BABOK акцентує увагу на значенні повного й послідовного виявлення бізнес-проблем, аналізі контексту, формуванні вимог та їх узгодженні з учасниками проєкту. Значна увага приділяється моделюванню, зокрема використанню UML- і BPMN-нотацій, що дозволяє забезпечити точність, структурованість і несуперечливість опису функціональності. Теоретичні аспекти BABOK активно досліджуються в українському науковому середовищі, зокрема у працях Катренка, Шулик та Фадєєвої, які підкреслюють важливість комплексного моделювання вимог для підвищення якості кінцевого продукту. [35, 15]

Управління вимогами у контексті проєктного менеджменту ґрунтується також на положеннях PMBOK (Project Management Body of Knowledge), що є одним із найбільш визнаних міжнародних стандартів у сфері управління проєктами. PMBOK розглядає роботу з вимогами як частину групи процесів управління змістом проєкту (Scope Management). [1] У цьому контексті особливе значення мають процедури збору та аналізу вимог, уточнення змісту продукту, побудова структури декомпозиції робіт (WBS) та контроль змін, які впливають на обсяг і якість кінцевого результату. У працях Бондарчука та Сметанюка (2020) наголошується, що PMBOK забезпечує методологічну основу для визначення меж проєкту та дозволяє керувати вимогами таким чином, щоб уникати розширення змісту без належного обґрунтування й узгодження.

Еволюція підходів до роботи з вимогами значною мірою пов’язана з поширенням Agile-методологій, які передбачають ітеративне формування вимог та їх безперервне уточнення відповідно до зворотного зв’язку від користувачів та стейкхолдерів. У Scrum- і Kanban-командах вимоги здебільшого подаються у формі user stories, що містять короткий опис цінності для користувача та визначають expected behavior. Критерії прийнятності, описані у вигляді acceptance criteria, забезпечують можливість перевірки реалізованого функціоналу. Процеси backlog refinement і continuous discovery сприяють

постійному оновленню вимог, їх пріоритезації та адаптації до змін. Згідно з дослідженнями українських авторів, такі підходи дозволяють мінімізувати кількість невірно визначених вимог на ранніх етапах і зменшують ризики помилок у складних IT-проєктах, зберігаючи при цьому гнучкість системи. [15]

Істотне значення у формуванні та перевірці вимог мають методики моделювання, які забезпечують можливість візуального представлення системи, її процесів та взаємодій. Використання UML-діаграм (use case, activity, sequence) допомагає формалізувати сценарії роботи системи та визначити функціональні залежності між компонентами. BPMN використовується для опису бізнес-процесів та логіки їх виконання, що дозволяє узгоджувати вимоги з бізнес-цілями підприємства. Діаграми потоків даних (DFD) застосовуються для дослідження інформаційних потоків та ідентифікації вузлів оброблення даних. У працях Данилюк і Сметанюка підкреслюється, що моделювання є важливою складовою перевірки повноти, несуперечливості та реалізовності вимог, а також сприяє покращенню комунікації між бізнес-аналітиками, архітекторами й розробниками. [30, 15]

Системний аналіз є ще одним важливим підходом до розроблення вимог, оскільки дозволяє визначати межі системи, окреслювати її основні функції та залежності, а також оцінювати взаємодію із зовнішніми елементами. Він спрямований на розгляд системи як єдиного цілого, що включає підсистеми, обмеження, ризики та сценарії поведінки.

Таблиця 1.1

Порівняльна характеристика стандартів і методик розроблення вимог

Стандарт / Методика	**Основний зміст**	**Що регламентує**	**Значення для ІТ-проєктів**
ISO/IEC/IEEE 29148	Інженерія вимог	Типи вимог, характеристики якості	Забезпечує формалізацію та структурованість
ISO/IEC 12207	Життєвий цикл ПЗ	Місце й роль управління вимогами	Забезпечує зв'язок між вимогами та процесами
BABOK	Бізнес-аналіз	Виявлення та моделювання вимог	Орієнтація на потреби стейкхолдерів
PMBOK	Проєктне управління	Управління змістом та змінами	Контроль вимог у межах проєкту
Agile підходи	Гнучкі методи	User stories, backlog, критерії прийнятності	Забезпечують адаптивність і швидку зміну вимог

Узагальнюючи викладені положення, можна зазначити, що стандарти та методики розроблення вимог формують методологічну основу для організації процесу requirements engineering у сучасних ІТ-проєктах. Міжнародні стандарти, такі як ISO/IEC/IEEE 29148 та ISO/IEC 12207, забезпечують формалізацію термінології, класифікацію вимог, визначення критеріїв їх якості та інтеграцію процесів роботи з вимогами у загальний життєвий цикл програмного забезпечення. У той же час глобальні методології, зокрема BABOK і PMBOK, пропонують практичні механізми виявлення потреб стейкхолдерів, документування, моделювання та контролю вимог, що підвищує керованість та прозорість проєктної діяльності.

Важливе місце займають гнучкі підходи Agile, які орієнтовані на поступове уточнення вимог, швидку адаптацію до змін і тісну співпрацю з користувачами. Їх доповнюють методики моделювання, що забезпечують структуроване відображення функціональності й процесів системи, сприяють усуненню

неоднозначностей та полегшують комунікацію між аналітиками, архітекторами та розробниками. Системний аналіз, у свою чергу, дозволяє більш глибоко дослідити контекст, взаємозв'язки та межі майбутньої системи.

Таким чином, стандарти й методики розроблення вимог становлять багаторівневу основу, яка охоплює як формальні регламенти міжнародних організацій, так і практикоорієнтовані інструменти бізнес-аналізу та проєктного менеджменту. Їх комплексне застосування забезпечує точність, повноту й узгодженість вимог, що є критично важливим для успішної реалізації IT-проєктів у умовах високої технологічної складності та динаміки сучасного ринку.

1.3 Особливості вимог до систем CI/CD та інструментів безпеки

Сучасні IT-проєкти характеризуються високою динамічністю змін, необхідністю регулярних оновлень і швидким циклом розроблення, що зумовлює потребу у використанні систем CI/CD (Continuous Integration / Continuous Delivery) як фундаменту DevOps-процесів. [13] Ці системи забезпечують автоматизацію інтеграції коду, тестування, побудови артефактів і розгортання програмних продуктів, що істотно скорочує час між розробленням і випуском релізів. Проте ефективність CI/CD залежить від коректно сформованих вимог, які охоплюють технічні, організаційні, інформаційні та безпекові аспекти.

CI/CD розглядається як підхід до розроблення програмного забезпечення, який об'єднує процеси безперервної інтеграції та безперервної доставки з метою оптимізації й прискорення циклу створення продукту. Використання CI/CD забезпечує підвищення ефективності, оскільки дозволяє автоматизувати етапи збирання, тестування та розгортання оновлень програмного коду. [28]

Застосування цієї практики дає змогу своєчасно виявляти помилки та усувати їх на ранніх стадіях життєвого циклу розроблення, що підвищує якість програмного забезпечення та зменшує ризики появи критичних дефектів. Водночас впровадження CI/CD є складним завданням, оскільки потребує змін не

лише в технічній інфраструктурі, а й у культурі роботи команди та організаційних процесах.

Безперервна інтеграція (CI) і безперервна доставка (CD) – це два поняття, пов’язані з розробкою програмного забезпечення, але вони не є тотожними. CI в основному займається створенням, тестуванням, компіляцією та інтеграцією змін у коді, тоді як CD займається розгортанням цих змін у виробничому середовищі. [14]

Таблиця 1.2

Порівняльна характеристика безперервної інтеграції (CI) та безперервної доставки (CD)

Характеристика	**Безперервна інтеграція (CI)**	**Безперервна доставка (CD)**
Основний фокус		
Головний фокус	Автоматичне збирання, перевірка та інтеграція змін коду в основну гілку	Автоматизоване підготовлення й розгортання перевірених змін у робочі середовища
Головне призначення	Своєчасне виявлення дефектів, забезпечення стабільності та сумісності коду	Мінімізація часу між розробкою та випуском продукту, підвищення частоти релізів
Характеристики процесу		
Частота виконання процесів	Зазвичай кілька разів на день, щораз після коміту	Виконується щоразу, коли код успішно проходить автоматизовані тести
Необхідність втручання людини	Мінімальна участь людини; процеси відбуваються автоматично	Можливе ручне затвердження перед розгортанням у production
Рівень автоматизації	Висока автоматизація етапів збірки та тестування	Висока автоматизація процесів релізу, конфігурації та розгортання

Продовження таблиці 1.2

Вплив на розробку		
Вплив на якість коду	Дозволяє оперативно виявляти інтеграційні конфлікти та технічні дефекти	Забезпечує стабільність і послідовність випусків програмного забезпечення
Вплив на продуктивність команди	Знижує ризик накопичення технічного боргу та зменшує складність інтеграції	Прискорює вихід оновлень і підвищує гнучкість процесів розроблення

Основною метою конвеєра CI/CD є повна автоматизація процесу доставки програмного забезпечення – від моменту внесення змін до коду до його розгортання у робочому середовищі. Такий рівень автоматизації забезпечує підвищення швидкості, ефективності та якості випусків програмного продукту.

Рис. 1.1 Модель безперервної інтеграції та доставки (CI/CD) у процесі розроблення програмного забезпечення

Вимоги до CI/CD-систем насамперед визначаються необхідністю забезпечення безперервності, передбачуваності та відтворюваності процесів розроблення. Однією з ключових характеристик є вимога до масштабованості, оскільки система повинна підтримувати одночасне виконання великої кількості pipeline-процесів, незалежних середовищ тестування та інтеграційних сценаріїв.

Важливою також є вимога до швидкодії, адже тривалі pipeline-процеси знижують продуктивність команди й збільшують час виходу продукту на ринок. До технічних вимог належить стабільність роботи компонента CI/CD, можливість автоматичного відновлення у разі збою, підтримка контейнеризації (Docker, Kubernetes), а також інтеграція зі сторонніми сервісами, наприклад, ***** Runner, Jenkins Agent, GitHub Actions Runners або системами хмарного виконання завдань.

Одним із центральних елементів вимог до CI/CD є наявність автоматизованих тестових механізмів. Вони включають модульне, інтеграційне, навантажувальне тестування, а також автоматизовану валідацію API та UI. Вимоги передбачають не лише визначення обсягу тестування, але й критерії проходження тестів, правила покриття коду, логіку оновлення тестової інфраструктури та поведінку системи в разі виявлення помилок. [14] У CI/CD особливу роль відіграють вимоги до прозорості й трасованості процесів: система має забезпечувати детальне логування, сповіщення про помилки, аудит змін та доступ до історії виконання pipeline.

Окремою групою є вимоги щодо безпеки, які інтегруються в CI/CD у межах підходу DevSecOps. Системи CI/CD повинні містити інструменти безпеки, що автоматично перевіряють код і залежності раніше, ніж функціонал буде розгорнуто у виробниче середовище. До таких вимог належать: автоматизований статичний аналіз коду (SAST), динамічне тестування безпеки (DAST), аналіз контейнерних образів, перевірка залежностей (dependency scanning), виявлення секретів у коді та моніторинг конфігурацій інфраструктури. Усі ці перевірки мають виконуватися автоматично при кожному оновленні коду.

Вимоги до безпеки передбачають також контроль прав доступу, використання принципу найменших привілеїв (least privilege), ізоляцію середовищ, шифрування даних у процесі передавання й зберігання, а також створення механізмів для оперативного виявлення та реагування на інциденти

безпеки. Важливо, щоб CI/CD-система відповідала вимогам стандартів, зокрема NIST Secure Software Development Framework, ISO/IEC 27001 і рекомендаціям OWASP, які визначають правила безпечної розробки й тестування.

Особливістю вимог до інструментів безпеки є їхня інтегрованість у щоденний робочий процес. Інструменти не повинні створювати надмірні затримки або складнощі, адже це суперечить принципам DevOps. Тому важливо, щоб процеси безпеки були автоматизованими, конфігурованими, прозорими для всіх учасників проєкту й не вимагали значних додаткових ресурсів з боку розробників чи тестувальників. Важливим елементом таких вимог є можливість швидкого оновлення правил безпеки, реагування на критичні вразливості та застосування патчів без порушення цілісності CI/CD-процесів.

Загалом вимоги до систем CI/CD та інструментів безпеки мають забезпечувати їхню надійність, передбачуваність та відповідність сучасним стандартам розроблення програмного забезпечення. Вони повинні не лише сприяти автоматизації процесів, але й забезпечувати комплексний контроль якості й безпеки на всіх етапах життєвого циклу продукту. У результаті правильно сформовані вимоги дають змогу створювати високонадійні IT-рішення, мінімізувати ризики та забезпечувати стабільні процеси Continuous Integration і Continuous Delivery у межах DevOps- та DevSecOps-підходів.

Таким чином, вимоги до систем CI/CD та інтегрованих інструментів безпеки формують комплексну основу для забезпечення надійності, безперервності та передбачуваності процесів розроблення програмного забезпечення. У контексті DevOps- та DevSecOps-підходів ці вимоги охоплюють технічні, організаційні та безпекові аспекти, що визначають здатність системи підтримувати часті релізи, автоматизоване тестування, стабільну інтеграцію компонентів та ефективне розгортання у різних середовищах. Визначено, що ключовими характеристиками є масштабованість, швидкодія та відмовостійкість CI/CD-процесів, а також можливість їх розширення за рахунок підключення

інструментів контролю якості та безпеки.

Інструменти DevSecOps, інтегровані в CI/CD, забезпечують автоматизоване виявлення вразливостей на всіх етапах життєвого циклу продукту, що значно знижує ризики експлуатаційних інцидентів та підвищує рівень захищеності системи. Водночас вимоги до безпеки мають бути гнучкими, актуальними та такими, що не створюють бар'єрів для швидкого виконання pipeline-процесів. Важливе значення має наявність аудитів, логування, контролю доступу та відповідності рекомендаціям міжнародних стандартів, таких як NIST SSDF та ISO/IEC 27001.

Можна зробити висновок, що правильно сформульовані вимоги до CI/CD-систем та інструментів безпеки є критичним чинником успішності сучасних IT-проєктів. Вони забезпечують інтегрованість процесів, підтримку високої якості коду, прискорюють випуск оновлень та сприяють створенню стійких, захищених і масштабованих програмних рішень у середовищі швидко змінних технологічних вимог.

1.4 Ризики й виклики під час формування технічного завдання IT-проєктів

Формування технічного завдання (ТЗ) є одним із найкритичніших етапів у життєвому циклі IT-проєкту, оскільки саме на цьому етапі визначаються основні вимоги до системи, межі проєкту, функціональні можливості та критерії якості. Невизначеність або некоректність ТЗ стає джерелом значних ризиків для подальших процесів проєктування, розроблення та тестування. Більшість невдач у реалізації IT-проєктів пов'язана саме з помилками у формуванні технічних вимог, що підтверджується численними дослідженнями у сфері управління проєктами.

Нижче розглянуто основні ризики та виклики, які впливають на процес створення технічного завдання.

1. Невизначеність вимог та недостатня участь стейкхолдерів

Одним із найпоширеніших ризиків є неповне або неточне розуміння потреб замовника, кінцевих користувачів та інших зацікавлених сторін. Відсутність системної комунікації або недостатня участь стейкхолдерів у процесі збору вимог призводить до появи «сліпих зон» у специфікації, некоректної інтерпретації функціоналу та розбіжностей у очікуваннях.

Наслідками є необхідність значних доробок на пізніх етапах, що збільшує вартість і тривалість розроблення.

2. Ризик неповноти й неоднозначності технічних вимог

ТЗ часто містить нечіткі формулювання, загальні описові фрази та припущення, які не дозволяють однозначно інтерпретувати вимоги. Невизначені нефункціональні вимоги (продуктивність, масштабованість, безпека) створюють загрозу того, що система не відповідатиме реальним параметрам експлуатації.

Неоднозначні формулювання («система має працювати швидко», «інтерфейс повинен бути зручним») не підлягають верифікації, що унеможливлює подальший контроль якості.

3. Технічні обмеження та невраховані архітектурні залежності

Викликом є врахування технологічних обмежень, сумісності з існуючими системами, інфраструктури та інтерфейсів. Невраховані залежності призводять до технічних колізій під час інтеграції, збільшення складності реалізації та необхідності у переробці архітектурних рішень.

Особливо це актуально для систем CI/CD, мікросервісної архітектури та високонавантажених платформ.

4. Ризики, пов'язані з вибором методології проєкту

Помилковий вибір методології може спричинити невідповідність підходів до формування вимог характеру проєкту. Наприклад, застосування водоспадних (Waterfall) підходів до інноваційних продуктів із високою динамікою змін зумовлює виникнення значних відхилень та переробок.

Виклики виникають і у гнучких методологіях Agile, коли вимоги формуються інкрементально, що може призвести до фрагментованості бачення продукту.

5. Ризики комунікацій та інформаційних розривів між командами

IT-проєкти залучають різні групи фахівців: бізнес-аналітиків, девелоперів, тестувальників, DevOps-інженерів, дизайнерів, спеціалістів з безпеки. Недостатньо формалізована комунікація може спричинити:

– суперечливі трактування вимог,

– втрати інформації під час передачі,

– помилки у формуванні acceptance criteria.

Це характерно особливо для розподілених команд, які працюють у різних часових зонах (як у *****, Amazon або Google).

6. Недооцінювання нефункціональних вимог

Серед найбільш критичних викликів – недооцінювання параметрів продуктивності, безпеки, надійності, масштабованості. Вимоги до UX/UI, доступності, відповідності стандартам (наприклад, GDPR) також часто залишаються поза увагою.

Нефункціональні вимоги становлять основу для побудови архітектурних рішень, тому їх ігнорування призводить до проблемних релізів та перевищення бюджету.

7. Зміни вимог у процесі розроблення («scope creep»)

Невпорядкованість процесу контролю змін призводить до постійного розширення обсягу робіт без відповідного коригування планів, ресурсів і термінів. Феномен «scope creep» є однією з основних причин провалу IT-проєктів, що підтверджується дослідженнями PMI (Pulse of the Profession, 2022).

8. Обмеження часу, бюджету та відсутність реалістичних оцінок

Вимоги, сформовані під тиском часу або через брак аналітичної роботи, призводять до появи ризиків технічних боргів, низької якості рішення та

незадоволення кінцевих користувачів.

Команди часто перевищують бюджет саме через недостатню деталізацію ТЗ.

9. Недостатній рівень документування

Документація може бути надто обмеженою, фрагментарною або застарілою. Це ускладнює планування, тестування, інтеграцію та супровід системи. Відсутність Traceability Matrix перешкоджає відстеженню взаємозв'язків між вимогами, задачами та результатами.

10. Людський фактор та недостатня кваліфікація аналітиків

Помилки аналітиків, недостатній досвід у роботі з доменною моделлю або складними системами, відсутність розуміння технічних аспектів спричиняють неякісне формування вимог.

Особливо гостро це проявляється у проєктах, де поєднується програмна інженерія, ML/AI та кібербезпека.

Можна зробити висновок, що формування технічного завдання IT-проєкту є процесом із високим рівнем складності та множинними ризиками. Їх джерелами стають як зовнішні чинники (динаміка ринку, зміни бізнес-вимог), так і внутрішні (організаційні бар'єри, технічні обмеження, комунікаційні розриви). Зниження ризиків можливе лише за умови використання системного підходу до вимоготехніки, впровадження стандартів ISO/IEC/IEEE, застосування методів бізнес-аналізу (BABOK), контролю змін (PMBOK), а також сучасних практик DevOps і DevSecOps. Комплексне застосування цих інструментів забезпечує високу якість ТЗ та підвищує ймовірність успішного завершення проєкту.

РОЗДІЛ 2.
АНАЛІЗ ОРГАНІЗАЦІЇ РЕАЛІЗАЦІЇ ПРОЄКТІВ У КОМПАНІЇ *****

2.1 Організаційно-управлінські засади діяльності *****

***** є однією з найбільш унікальних компаній у сфері розроблення програмного забезпечення, оскільки її управлінська модель базується на повністю віддаленій організації праці, відкритості внутрішніх процесів та високому рівні стандартизації взаємодії між командами. Компанія позиціонує себе як all-remote organization, тобто таку, що не має фізичних офісів і функціонує на основі цифрових інструментів комунікації, чітко регламентованих процедур та прозорого розподілу відповідальності. Ця модель є фундаментом організаційних процесів і визначає підходи до планування, реалізації та контролю ІТ-проєктів.

***** є одним із найвідоміших інструментів для управління репозиторіями коду та супроводу DevOps-проєктів, проте його історія розпочалася як невеликий open-source експеримент ентузіастів. Компанія сформувалася навколо ідеї створення повністю інтегрованої платформи для розроблення програмного забезпечення, доступної у відкритому середовищі.

Першу версію ***** було створено у 2011 році українським розробником Дмитром Запорожцем. Він працював у невеликій компанії та стикався з нестачею доступних інструментів для командної роботи з Git-репозиторіями. На той час альтернативи були або закритими, або занадто дорогими для малих команд. Це стало поштовхом до розроблення власного рішення, яке б поєднувало контроль версій, управління задачами та базові можливості співпраці над кодом у режимі open-source. [20]

***** швидко набув популярності в спільноті розробників завдяки відкритому вихідному коду, гнучкості налаштувань та можливості розгорнути систему на власному сервері. У 2012 році проєкт привернув увагу

нідерландського підприємця Сіджа Бранджі. Він запропонував створити компанію, яка забезпечувала б професійну підтримку та розвиток ***** як open-core продукту. Таким чином було засновано компанію ***** B.V., яка згодом трансформувалася у ***** Inc.

З моменту комерційного запуску ***** почав активно розвиватися як інтегрована DevOps-платформа, яка охоплює всі етапи життєвого циклу програмного забезпечення – від планування та написання коду до тестування, CI/CD-процесів, моніторингу та безпеки. Це дало змогу позиціонувати ***** як рішення формате "single application" – єдиний застосунок, який замінює кілька інструментів, традиційно розрізнених у DevOps-екосистемах.

У 2015–2016 роках компанія отримала перші значні інвестиції, що дозволило масштабувати розроблення та розширити функціональність продукту. Серед інвесторів були Khosla Ventures, August Capital та GV (Google Ventures). У цей період ***** запровадив модель повністю віддаленої роботи – all-remote, яка залишається однією з ключових особливостей організації до сьогодні.

Подальший розвиток ***** характеризувався активною інтеграцією інструментів DevSecOps, автоматизацією CI/CD-процесів, підтримкою Kubernetes та широким переходом компанії до моделі відкритої документації. У 2021 році ***** Inc. успішно вийшла на біржу NASDAQ (тикер GTLB), що стало важливим етапом у становленні компанії як глобального постачальника DevOps-рішень enterprise-рівня.

На сьогодні ***** є однією з провідних платформ DevOps у світі, яку використовують понад 100000 організацій. Компанія продовжує розвивати open-core підхід, активно взаємодіяти зі спільнотою розробників і впроваджувати рішення на основі штучного інтелекту для автоматизації аналізу коду, тестування та підвищення продуктивності команд.

На даний час команда ***** Inc. складається з 2561 членів команди. Є найбільшою у світі організацією, яка працює повністю віддалено, і наразі має

членів команди у понад 65 країнах та регіонах. [27]

*Рис. 2.1 Особливості, що роблять ***** привабливим для користувачів*

Центральним елементом управлінської системи ***** виступає ***** Handbook – найбільший у світі відкритий корпоративний довідник, що містить детальні правила, політики, процедури та методики роботи. Handbook виконує функцію основного джерела знань та стандартів, замінюючи традиційні настанови, внутрішні наради й адміністративні інструкції. Унаслідок цього кожен співробітник має доступ до єдиної інформаційної бази, що забезпечує передбачуваність дій та узгодженість управлінських рішень. Такий підхід, за моделлю *****, сприяє зменшенню комунікаційних бар'єрів, підвищенню продуктивності команд і формує культуру прозорості.

Організаційна структура ***** є кросфункціональною та побудованою за принципом поділу на продуктові групи (product groups), які працюють над окремими модулями єдиної DevOps-платформи.

***** має унікальну управлінську модель, яка поєднує принципи повністю віддаленої роботи (all-remote), відкритості внутрішньої інформації та високої децентралізації прийняття рішень. Центральним органом стратегічного управління компанії є E-Group – виконавча команда, яка визначає ключові

напрями розвитку продукту, бізнес-стратегію та операційну діяльність. Структура E-Group охоплює основні функціональні напрями компанії – технологічний, фінансовий, операційний, продуктовий, HR-напрям та клієнтські операції.

Згідно з офіційною інформацією *****, організаційна структура E-Group включає такі ключові ролі, що зображені на рисунку 2.2.

*Рис. 2.2 Організаційна структура ***** (керівна команда E-Group)*

До складу типової продуктової групи входять Product Manager, Engineering Manager, Software Engineers, UX-фахівці, Technical Writer та Quality Engineers. Кожна група відповідає за повний життєвий цикл своєї частини продукту – від дослідження потреб користувачів до розгортання функціональності та її

подальшого супроводу. Такий розподіл формує децентралізовану модель управління, у якій команди мають високий рівень автономії та здатні ухвалювати рішення щодо пріоритетів, технічних рішень та організації робочих процесів.

Управління ***** ґрунтується на відкритому обговоренні ініціатив, рішень і змін. Значна частина процесів – включно з roadmap, product vision, issue tracking, обговоренням архітектурних рішень – є відкритою та доступною для перегляду зовнішнім користувачам. Така прозорість притаманна open-core моделі ***** та слугує інструментом для підвищення якості розроблення, залучення спільноти і мінімізації управлінських ризиків.

Особливе місце в організаційній моделі ***** посідають принципи asynchronous communication та documentation-first. Робочі процеси будуються таким чином, щоб співробітники могли взаємодіяти незалежно від часових поясів. Це забезпечується завдяки детальному документуванню рішень і процесів, фіксації інформації у вигляді issue та merge requests, а також використанню чітких шаблонів документів. Такий підхід сприяє підвищенню якості комунікації та знижує ймовірність втрати інформації.

Крім того, ***** використовує модель OKR (Objectives and Key Results) для стратегічного планування та встановлення вимірюваних результатів. Це дає змогу узгоджувати цілі команд із загальною продуктовою стратегією компанії та підтримувати орієнтацію на користувацьку цінність. Кожна операційна група формує свої OKR відповідно до квартальних і річних пріоритетів, що дозволяє зберігати баланс між стратегічним розвитком продукту та поточними завданнями.

Важливим елементом управлінської моделі ***** є акцент на самоорганізації та відповідальності працівників. Відповідно до корпоративних принципів, кожен співробітник має свободу у виборі підходів до виконання завдань, але водночас несе повну відповідальність за якість та дотримання стандартів. Така культура підтримує швидкість реалізації, стимулює ініціативність та створює сприятливі умови для професійного розвитку.

Таким чином, організаційно-управлінська модель ***** характеризується високим рівнем відкритості, стандартизації, автономності команд і опорою на документування як ключовий засіб координації діяльності. Це забезпечує ефективне управління складними DevOps-проєктами та підтримує можливість масштабування продукту без втрати якості та узгодженості процесів.

Таблиця 2.1

Консолідований баланс ***** Inc. станом на 31 січня 2024 та 31 січня 2023 рр.

(тис. дол. США)

Стаття балансу	2024	2023
АКТИВИ		
Оборотні активи		
Грошові кошти та їх еквіваленти	287 996	295 402
Короткострокові інвестиції	748 289	641 249
Дебіторська заборгованість (нетто)	166 731	130 479
Витрати на залучення контрактів, поточні	32 300	26 505
Передплачені витрати та інші оборотні активи	45 601	24 327
Усього оборотних активів	**1 280 917**	**1 117 962**
Основні засоби (нетто)	2 954	5 797
Активи з прав користування (операційна оренда)	405	998
Інвестиції за методом участі (нетто)	–	12 682
Гудвіл	8 145	8 145
Нематеріальні активи (нетто)	1 733	3 901
Витрати на залучення контрактів, довгострокові	19 317	15 628
Інші необоротні активи	4 390	4 087
Усього активів	**1 317 861**	**1 169 200**
ЗОБОВ'ЯЗАННЯ ТА КАПІТАЛ		
Поточні зобов'язання		
Кредиторська заборгованість	1 738	5 184
Нараховані витрати та інші поточні зобов'язання	286 178	25 954
Нарахована заробітна плата та пільги	35 809	20 776
Відстрочений дохід, поточний	338 348	254 382
Усього поточних зобов'язань	**662 073**	**306 296**
Відстрочений дохід, довгостроковий	23 794	28 355
Інші довгострокові зобов'язання	14 060	9 824
Усього зобов'язань	**699 927**	**344 475**

Продовження таблиці 2.1

Капітал акціонерів		
Додатково сплачений капітал	1 718 661	1 497 373
Накопичений збиток	(1 149 822)	(725 648)
Інший сукупний прибуток (збиток)	2 335	(705)
Усього капіталу ***	**571 174**	**771 020**
Частка неконтрольованих інтересів	46 760	53 705
Загальний капітал	**617 934**	**824 725**
Усього зобов'язань і капіталу	**1 317 861**	**1 169 200**

Аналіз фінансових показників ***** Inc. за 2022–2024 фінансові роки дає змогу оцінити динаміку розвитку компанії, її здатність забезпечувати довгострокову платоспроможність та ефективність використання ресурсів. ***** продовжує перебувати на етапі активного масштабування бізнесу, що супроводжується значними інвестиціями у дослідження, інженерну інфраструктуру та продажі, проте компанія демонструє стійке зростання доходів і позитивні сигнали щодо підвищення фінансової стійкості.

За аналізований період ***** демонструє високі темпи зростання доходів, що представлено на рисунку 2.3.

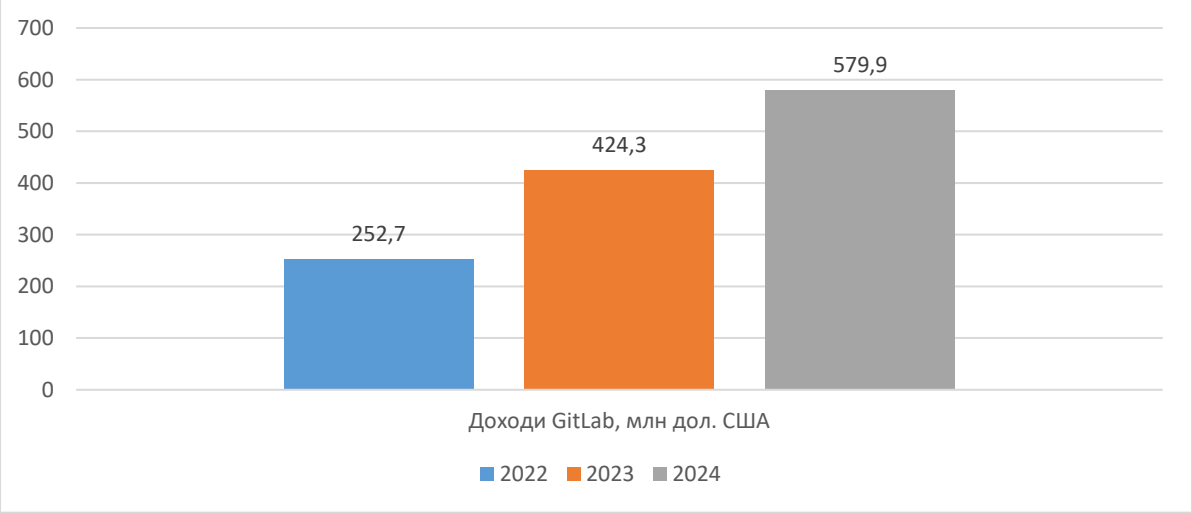

*Рис. 2.3 Динаміка доходів ***** за 2022-2024 рр.*

Загальне зростання доходів за три роки становить понад 130%, що свідчить про швидке розширення клієнтської бази та збільшення попиту на DevOps-

платформу *****. Зокрема у 2023 році спостерігалось зростання на 68 %, а у 2024 році – на 36%.

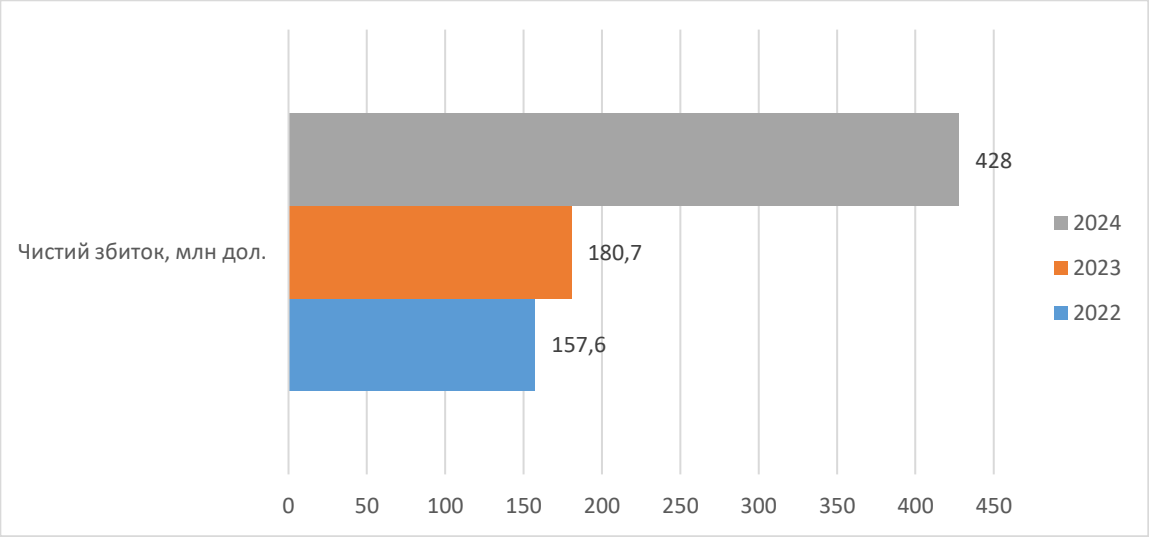

Рис. 2.4 Динаміка чистого прибутку (збитку), млн. дол.

Водночас компанія залишається збитковою за GAAP-стандартами. Чистий збиток ***** у 2024 році становив 428,0 млн. дол., що є значним погіршенням порівняно з 2023 роком (180,7 млн. дол.). Одним із ключових факторів такого відхилення стала стаття податкових витрат (+264 млн. дол.), пов'язана зі зменшенням можливості відтермінування податкових зобов’язань.

Операційний збиток зменшився з 211,4 млн дол. у 2023 році до 187,4 млн дол. у 2024 році, що свідчить про часткове покращення ефективності операційної діяльності попри збільшення обсягів витрат на розвиток продукту.

Нижче подано аналітичну таблицю ключових коефіцієнтів фінансової стійкості ***** за 2022–2024 фінансові роки. Для розрахунків використано дані з консолідованого балансу ***** Inc. (додаток 2).

Таблиця. 2.2

Основні коефіцієнти фінансової стійкості ***** за 2022–2024 рр.

Показники	2022	2023	2024
Коефіцієнт фінансової автономії	0,38	0,71	0,47
Коефіцієнт концентрації позикового капіталу	0,62	0,29	0,53
Коефіцієнт фінансового ризику	1,63	0,45	1,13
Коефіцієнт фінансової стабільності	0,47	0,73	0,48

Аналіз динаміки ключових коефіцієнтів фінансової стійкості ***** за 2022–2024 роки дає підстави стверджувати, що компанія зберігає стабільний, хоча і неоднозначний, фінансовий профіль. Коефіцієнт фінансової автономії протягом цього періоду варіюється в межах 0,38–0,47, що свідчить про помірну залежність від зовнішніх джерел фінансування та достатній рівень покриття активів власним капіталом. Показник концентрації позикового капіталу демонструє зміну структури фінансування: у 2023 році ***** характеризувався низькою часткою зобов’язань, зокрема - 0,29, тоді як у 2024 році спостерігається її зростання до 0,53 через збільшення відстрочених доходів, що є типовим для SaaS-компаній та не несе суттєвих ризиків.

Коефіцієнт фінансового ризику у 2023 році досяг мінімального значення (0,45), що вказує на високий рівень капітальної стійкості, однак у 2024 році він збільшується до 1,13. Це означає, що зобов’язання перевищують власний капітал, проте такий рівень не є критичним, враховуючи стійку ліквідність компанії та відсутність значних довгострокових боргів. Коефіцієнт фінансової стабільності підтверджує збереження достатнього обсягу стабільних джерел фінансування: показники залишаються на рівні 0,47–0,73, що свідчить про здатність ***** підтримувати активи за рахунок власного капіталу та довгострокових ресурсів.

Загалом аналіз показників дає змогу зробити висновок, що ***** зберігає

задовільну фінансову стійкість, незважаючи на коливання структури капіталу. Компанія демонструє позитивну тенденцію щодо підтримання високої ліквідності, достатній запас фінансової міцності та низький рівень боргового навантаження. Коливання окремих коефіцієнтів зумовлені специфікою діяльності компанії у сегменті DevOps та підписних сервісів, де значну частку зобов'язань становлять передплачені доходи, що фактично виступають джерелом фінансування майбутніх операцій. Отже, фінансова стійкість ***** може бути оцінена як така, що забезпечує стабільний розвиток і здатність компанії виконувати свої зобов'язання в середньостроковій перспективі.

Нижче подано таблицю коефіцієнтів ліквідності ***** за 2022–2024 роки.

Таблиця 2.3

Коефіцієнти ліквідності ***** за 2022–2024 рр.

Показники	2022	2023	2024	Нормативне значення
Коефіцієнт абсолютної ліквідності	1,31	3,07	1,56	≥ 0,2
Коефіцієнт швидкої ліквідності	1,26	3,58	1,85	≥ 0,7
Коефіцієнт загальної ліквідності	1,32	3,65	1,93	≥ 1,0

Аналіз коефіцієнтів ліквідності ***** за 2022–2024 роки свідчить про надзвичайно стійке фінансове становище компанії в частині її здатності виконувати поточні зобов'язання. Усі розраховані показники – абсолютної, швидкої та загальної ліквідності – суттєво перевищують нормативні значення, що вказує на високий рівень платоспроможності та значний запас фінансової міцності.

Коефіцієнт абсолютної ліквідності у 2022–2024 рр. коливається в межах 1,31–1,56, значно перевищуючи норму ≥0,2. Це означає, що ***** має достатній обсяг грошових коштів та короткострокових інвестицій для негайного погашення поточних зобов'язань без необхідності залучення додаткових ресурсів. Показники швидкої ліквідності, що становлять 1,26–3,58, також підтверджують здатність компанії виконувати поточні платежі за рахунок найбільш ліквідних

активів, не враховуючи передплачених витрат. Значення значно перевищують норматив ≥0,7, що характерно для фінансово стійких технологічних компаній із низьким рівнем короткострокових ризиків.

Коефіцієнт загальної ліквідності протягом аналізованих років перебуває в межах 1,32–3,65, що вказує на повне покриття поточних зобов'язань усіма оборотними активами. Особливо високі значення у 2023 році пояснюються значним обсягом грошових коштів та короткострокових інвестицій, акумульованих у структурі активів.

Отже, отримані результати дозволяють зробити висновок, що ***** демонструє високу короткострокову фінансову стійкість та ліквідність, що є характерним для компаній із підписною бізнес-моделлю та низьким рівнем боргового навантаження. Незважаючи на наявність збитковості за GAAP-стандартами, компанія має достатній обсяг ліквідних ресурсів для стабільного функціонування та покриття усіх поточних фінансових зобов'язань, що знижує ризики неплатоспроможності та підтверджує надійність фінансової структури ***** у середньостроковій перспективі.

Нижче подано розраховані показники рентабельності ***** Inc. за 2022–2024 фінансові роки.

Таблиця 2.4

Вихідні дані для розрахунку показників рентабельності

Показник	**2022**	**2023**	**2024**
Дохід (Revenue)	252 653	424 336	579 906
Операційний збиток	–128 957	–211 411	–187 440
Чистий збиток	–157 560	–180 696	–428 033
Активи (Total Assets)	1 160 000	1 169 200	1 317 861
Власний капітал (Equity)	440 000	824 725	617 934

Таблиця. 2.5

Показники рентабельності ***** за 2022–2024 рр.

Показник	2022	2023	2024
Рентабельність продажів (ROS) Net Loss / Revenue	–62,3%	–42,6%	–73,8%
Операційна рентабельність (Operating Margin) Operating Loss / Revenue	–51,0%	–49,8%	–32,3%
ROA (рентабельність активів) Net Loss / Assets	–13,6%	–15,4%	–32,5%
ROE (рентабельність власного капіталу) Net Loss / Equity	–35,8%	–21,9%	–69,2%

Узагальнений аналіз показників рентабельності ***** за 2022–2024 роки засвідчує, що компанія перебуває на стадії інтенсивного інвестиційного розвитку, що є характерним для високотехнологічних підприємств, орієнтованих на розширення ринку та нарощування продуктових можливостей. Усі ключові показники рентабельності – рентабельність продажів, операційна рентабельність, рентабельність активів та рентабельність власного капіталу – мають негативні значення протягом усього аналізованого періоду, що вказує на збитковість компанії за GAAP-стандартами.

Проте характер збитковості свідчить не про погіршення операційної ефективності, а про стратегічний характер витрат, пов’язаних з інвестиціями у розроблення продукту, маркетинг, інфраструктуру DevSecOps та збільшення кадрового потенціалу. Крім того, суттєве погіршення чистого результату у 2024 році пояснюється насамперед разовими податковими коригуваннями, що суттєво збільшили величину чистого збитку, не вплинувши при цьому на операційний прибуток. Водночас операційна рентабельність демонструє ознаки покращення, що свідчить про поступове підвищення ефективності основної діяльності в умовах зростання доходів.

Рентабельність активів і власного капіталу також залишаються від’ємними, проте динаміка цих показників тісно пов’язана із специфікою бізнес-моделі *****, яка не передбачає значного залучення позикових коштів і підтримує

високий рівень ліквідності за рахунок власного капіталу та передплачених доходів. Це дозволяє компанії зберігати стабільність навіть за умов збитковості та забезпечувати необхідний запас фінансової міцності для подальшого розвитку.

Таким чином, загальна оцінка рентабельності ***** свідчить про те, що компанія функціонує в межах моделі зростання, орієнтованої на довгострокове формування конкурентних переваг, а не на короткострокову прибутковість. Незважаючи на негативні значення рентабельності, ***** демонструє стійкі тенденції до покращення операційної ефективності, високий рівень ліквідності та значний потенціал зростання, що дозволяє розглядати компанію як фінансово життєздатну та перспективну у середньостроковому періоді.

2.2 Процеси планування та ініціації проєктів у *****

Процеси планування та ініціації проєктів у ***** ґрунтуються на принципах прозорості, ітеративності та повної документованості, що є ключовими елементами корпоративної культури компанії. *****, як організація з повністю віддаленою моделлю роботи, забезпечує стандартизований підхід до запуску та управління проєктами через централізовану систему внутрішньої документації – ***** Handbook, яка визначає правила, артефакти, відповідальні ролі та механізми прийняття рішень на ранніх етапах проєктного циклу. Відповідно до внутрішніх методичних матеріалів компанії, процеси ініціації та планування охоплюють формування ідей, їх оцінювання, пріоритезацію, створення дорожньої карти, визначення вимог та підготовку ресурсного забезпечення.

Нижче подано рисунок 2.5 «Процес ініціації проєкту в *****».

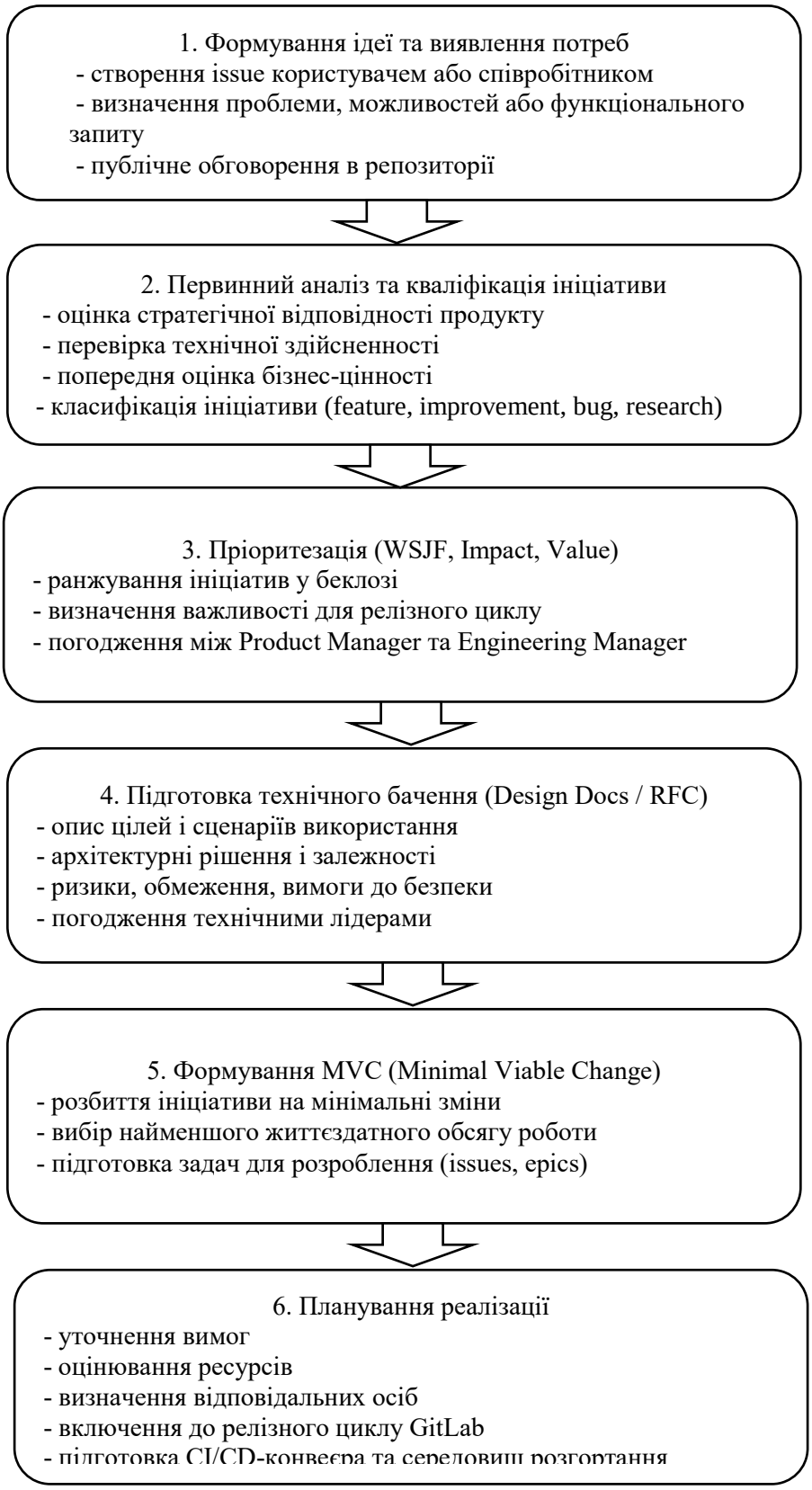

*Рис. 2.5 Процес ініціації проєкту в ******

Представлена схема відображає реальний порядок ініціації проєкту в *****, який базується на таких принципах:

- Прозорість – усі ідеї формуються через публічні issue;
- Ітеративність – великі ініціативи розбиваються на MVC;
- Документованість – кожен проєкт повинен мати Design Doc / RFC;
- Функціональна узгодженість – Product та Engineering ухвалюють рішення спільно;
- Релізна циклічність – ***** працює з фіксованим щомісячним релізом.

Першим етапом ініціації є ідентифікація потреби або проблеми, яка потребує інженерного або продуктового вирішення. У ***** цей процес здійснюється через публічні issue, відкриті обговорення в репозиторіях, сторінки напрямів Product Direction та зворотний зв'язок від користувачів. Будь-який співробітник або користувач може створити пропозицію щодо вдосконалення продукту, що забезпечує демократичний доступ до формування беклогу ініціатив. Подальше опрацювання здійснює продуктова команда, яка проводить первинний аналіз цінності ініціативи, визначає її відповідність стратегічним цілям компанії та оцінює потенційний вплив на користувацький досвід.

Наступним етапом є кваліфікація та пріоритезація ініціативи, що відбувається відповідно до методології ***** Product Development Flow. Критеріями для ухвалення рішень щодо включення ініціативи до плану проєкту є її стратегічна релевантність, технічна здійсненність, очікувана бізнес-цінність та можливість інтеграції з наявною архітектурою. Продуктовий менеджер здійснює поглиблений аналіз вимог, співставляючи запит із довгостроковим баченням розвитку продукту, технологічними обмеженнями та наявними ресурсами. У межах планування використовується система Weighted Shortest Job First (WSJF) та інші пріоритетизаційні підходи, що забезпечують об'єктивність у визначенні послідовності виконання задач.

Важливим елементом етапу планування є формування архітектурного та

технічного бачення проєкту, яке документується у вигляді Design Docs або RFC (Request for Comments). Ці документи затверджуються технічними керівниками (Engineering Managers, Staff Engineers) і описують цілі, функціональність, ризики, залежності, а також вимоги до безпеки та якості. Такий підхід дозволяє мінімізувати невизначеність на ранніх етапах та забезпечити узгодженість між продуктовими і технічними командами.

У процесі планування ***** активно застосовує практики ітеративного та інкрементального розвитку, орієнтуючись на фреймворк Continuous Discovery. Дорожні карти (roadmaps) формуються з урахуванням розбиття проєктів на дрібні, швидко реалізовані елементи – Minimal Viable Change (MVC). MVC є характерною особливістю організації ***** та відображає принцип створення мінімальної зміни, яка вже приносить користь користувачеві. Завдяки цьому знижується ризик затримок і покращується гнучкість у реагуванні на зворотний зв’язок.

Завершальним етапом ініціації є підготовка до реалізації, яка включає визначення відповідальних осіб, уточнення цілей, формування технічного завдання, оцінювання ресурсів та узгодження часових рамок у межах релізних циклів *****. Компанія дотримується щомісячного релізного графіку, що визначає темп планування й забезпечує регулярність оновлень продукту. Кожен проєкт інтегрується до загальної системи CI/CD, що забезпечує прогнозованість процесу розроблення та контроль якості на всіх етапах життєвого циклу.

Таким чином, процеси ініціації та планування проєктів у ***** характеризуються високим рівнем формалізованості, прозорості та відкритості. Компанія поєднує стратегічний підхід із гнучкими практиками управління, що дозволяє ефективно запускати нові проєкти й мінімізувати ризики на ранніх стадіях. Застосування MVC, документованої комунікації, структурованої пріоритезації та чіткої ролі продуктових і технічних команд забезпечує узгоджені і передбачувані процеси, що сприяють інноваційному розвитку та підтримують

конкурентоспроможність ***** на глобальному ринку DevOps-рішень.

2.3 Управління розробленням і реалізацією DevOps-проєктів

Управління DevOps-проєктами передбачає комплексний підхід до організації життєвого циклу програмного забезпечення, який забезпечує безперервну інтеграцію, доставку та експлуатацію продукту на основі тісної взаємодії команд розроблення та операційного супроводу. DevOps-модель трансформує традиційний процес управління IT-проєктами, запроваджуючи принципи автоматизації, спільної відповідальності та постійного вдосконалення, що дозволяє скоротити час між створенням коду та його розгортанням у виробничому середовищі.

Управління DevOps-проєктами ґрунтується на інтеграції процесів планування, розроблення, тестування, забезпечення безпеки, експлуатації та моніторингу в єдиному конвеєрі CI/CD. На етапі розроблення до проєкту висуваються вимоги щодо узгодженості коду, дотримання стандартів якості та забезпечення тестованості. DevOps-команди застосовують підхід trunk-based development, який передбачає короткі цикли розроблення й часті коміти, що мінімізує ризики конфліктів інтеграції. Управління цими процесами здійснюється через централізовані платформи, такі як *****, які забезпечують інструменти для контролю версій, управління задачами, рев'ю коду та автоматизованого тестування.

Ключовим елементом DevOps-проєктів є конвеєр безперервної інтеграції та доставки (CI/CD). На етапі розроблення менеджери та інженери визначають архітектуру конвеєра, набір обов'язкових перевірок, політики безпеки, стратегії тестування та правила випуску релізів. Автоматизовані конвеєри виконують збірку, запуск unit-, integration-, security- та performance-тестів, аналіз покриття коду, а також автоматичне розгортання у тестових і staging-середовищах. Наявність CI/CD забезпечує передбачуваність життєвого циклу продукту та

зменшує ймовірність людських помилок.

Особливу роль у DevOps-підході відіграє інфраструктура як код (Infrastructure as Code, IaC), що забезпечує автоматизоване керування середовищами розгортання. Завдяки таким інструментам, як Terraform, Ansible або ***** Environment Management, DevOps-команди можуть керувати конфігураціями інфраструктури у версіонованій формі, забезпечуючи відтворюваність середовищ, стандартизацію та швидке масштабування. Управління IaC-процесами включає затвердження змін, аудит, тестування конфігурацій і моніторинг стану кластерів та хмарних ресурсів.

Управління DevOps-проєктами також охоплює інтеграцію практик безпеки – DevSecOps, яка передбачає раннє виявлення ризиків та виконання перевірок безпеки в кожному етапі CI/CD-конвеєра. Автоматизовані сканери вразливостей, перевірка залежностей, аналіз контейнерних образів і політики секретів дозволяють зменшити ймовірність експлуатації вразливостей у виробничих середовищах. Постійна безперервна валідація безпеки є необхідною умовою реалізації масштабованих DevOps-проєктів.

У межах реалізації DevOps-проєкту значну увагу приділяють спостережуваності (observability), що включає моніторинг продуктивності, логування, трасування, аналіз інцидентів і підтримку SLO/SLI/SLA. Дані системної телеметрії забезпечують оперативне виявлення відхилень і дозволяють командам швидко реагувати на проблеми через механізми інцидент-менеджменту. Управління експлуатаційним етапом в DevOps передбачає також проведення постмортем-аналізів, безпекових рев'ю й оптимізацію конвеєра відповідно до отриманих інсайтів.

Управлінська складова DevOps-проєктів охоплює рольові моделі, що передбачають тісну взаємодію product manager, engineering manager, інженерів розроблення та експлуатації. На відміну від традиційних підходів, DevOps-команди мають спільну відповідальність за результат – стабільність, якість і

безперервність роботи програмного продукту. Планування здійснюється ітеративно, з урахуванням зворотного зв'язку, безперервних експериментів і вимірювань ефективності (наприклад, через DORA Metrics: Lead Time, Deployment Frequency, MTTR, Change Failure Rate).

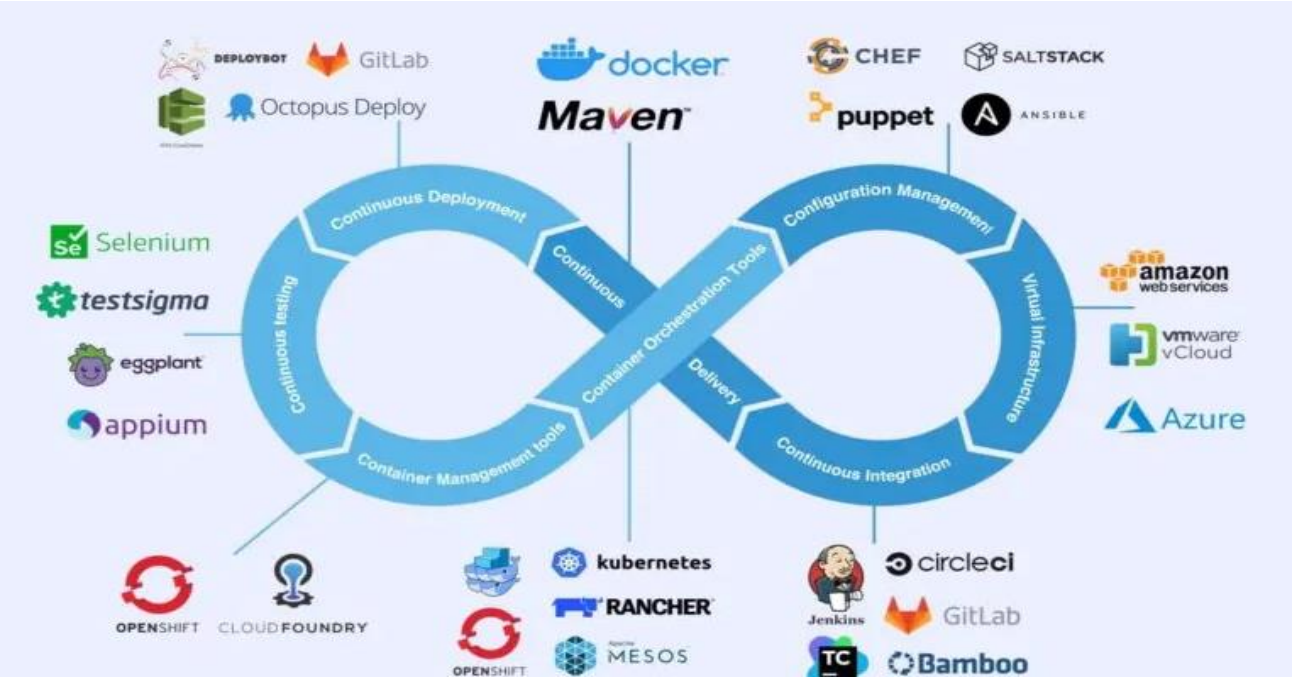

Рис. 2.6 DevOps Toolchain

Візуальна модель DevOps Toolchain, представлена на рисунку, відображає циклічну та безперервну природу DevOps-підходу, у межах якого процеси планування, розроблення, тестування, доставки та експлуатації інтегруються у єдиний операційний контур. Кожен сегмент «вісімки» відповідає окремому етапу життєвого циклу та пов'язаному з ним набору інструментів, які у сукупності формують цілісну DevOps-екосистему. Згідно з аналітичними даними Simform, саме узгоджене застосування цих інструментів забезпечує високу швидкість релізів, зниження операційних ризиків і зростання надійності програмних продуктів у виробничих середовищах.

Перший блок інструментів охоплює етапи планування, розроблення та безперервної інтеграції, у яких ключову роль відіграють системи контролю версій, автоматизовані тестові фреймворки та конвеєри CI. Інструменти на

рисунку – такі як *****, CircleCI, Jenkins та Bamboo – реалізують процеси автоматизованої збірки й тестування, дозволяючи командам швидко виявляти дефекти та підтримувати цілісність коду. Це відповідає рекомендаціям Simform щодо важливості скорочення Lead Time for Changes та регулярного інтегрування коду невеликими порціями, що зменшує ризики конфліктів і підвищує стабільність розроблення.

Другий сегмент циклу DevOps пов'язаний із доставкою, розгортанням та контейнерною оркестрацією. Такі інструменти, як Docker, Kubernetes, Rancher, Mesos і OpenShift, забезпечують незалежність середовищ, масштабованість і відтворюваність інфраструктури. Згідно з даними Simform, контейнеризація є критичним елементом сучасної DevOps-стратегії, оскільки дозволяє мінімізувати відмінності між development- та production-середовищами та забезпечує швидке й передбачуване розгортання. Наявність цих платформ у Toolchain свідчить про важливість автоматизованої інфраструктури в управлінні DevOps-проєктами, де IaC-підходи стають обов'язковим компонентом.

Третій блок інструментів включає системи тестування, забезпечення якості та операційного моніторингу. Selenium, Testsigma, Appium та Eggplant підтримують різні рівні автоматизованого тестування, що дозволяє командам забезпечувати стабільну якість продукту в умовах частих релізів. Одночасно такі платформи, як AWS, Azure та VMware Cloud, формують основу для масштабованих віртуальних середовищ, необхідних для безперебійної роботи DevOps-проєктів. Інфраструктура хмарних провайдерів інтегрується з CI/CD-конвеєрами, забезпечуючи можливість швидкого реагування на операційні інциденти та адаптивне масштабування сервісів.

Отже, структура DevOps Toolchain демонструє, що ефективне управління DevOps-проєктами передбачає не лише застосування окремих технологічних інструментів, а й створення узгодженої екосистеми, у якій усі етапи розроблення та доставки взаємопов'язані. Така інтеграція відповідає стратегічним принципам

DevOps, описаним Simform: безперервне вдосконалення, системна автоматизація, висока спостережуваність і культура співпраці між етапами життєвого циклу продукту. Управління DevOps-проєктами таким чином трансформується у комплексну діяльність, що поєднує технічні, організаційні та культурні аспекти, забезпечуючи якість, швидкість і надійність програмних рішень.

Таким чином, управління розробленням і реалізацією DevOps-проєктів у сучасних IT-компаніях, включно з *****, базується на комплексній інтеграції автоматизації, гнучких підходів і високих стандартів безпеки. DevOps підхід забезпечує скорочення циклу розроблення, підвищення якості релізів, ефективне використання ресурсів та зменшення ризиків, що формує передумови для створення стійких і високонадійних програмних продуктів.

2.4 Оцінювання ефективності моделі реалізації проєктів у *****

Оцінювання ефективності моделі реалізації проєктів у ***** базується на інтеграції кількісних і якісних показників, що характеризують продуктивність команд, швидкість розроблення, якість релізів і рівень операційної стабільності. ***** застосовує комплексну систему метрик, яка відображає логіку DevOps-підходу та дозволяє здійснювати систематичне вимірювання впливу процесів на кінцевий результат. Основними інструментами оцінювання є DevOps Research & Assessment (DORA) Metrics, моделі продуктивності ***** та аналітичні індикатори, що враховують швидкість потоку завдань, кількість інцидентів, результативність релізного циклу та відповідність бізнес-цілям.

Одним із ключових джерел для вимірювання ефективності реалізації проєктів є метрики DORA, що дозволяють оцінювати здатність команди до швидкого та безпечного доставлення змін. Згідно з ***** та світовими практиками, до основних показників DORA належать:

- частота розгортань (Deployment Frequency),

- час доставки змін (Lead Time for Changes),
- середній час реакції на інциденти (MTTR)
- частка невдалих змін (Change Failure Rate).

Завдяки автоматизованому CI/CD-конвеєру ***** досягає високої частоти розгортань, що дає змогу швидко впроваджувати поліпшення та скорочувати ризики, пов'язані з великими релізними пакетами. Низькі значення MTTR і стабільність конвеєра забезпечуються завдяки глибокій інтеграції інструментів моніторингу, тестування та DevSecOps-сканування, що зменшує вплив технічних збоїв.

Згідно зі звітом ***** DevSecOps Report 2023, 60% організацій, що використовують *****, випускають нові зміни щодня або навіть кілька разів на день, що відповідає рівню «High Performer» за моделлю DORA. [5]

Окремим напрямом оцінювання ефективності є аналіз потоку робіт (workflow efficiency), який ***** здійснює шляхом відстеження життєвого циклу задач: від створення issue до їхнього злиття в основну гілку коду. До ключових індикаторів належать:

- ✓ Cycle Time – час виконання задачі від створення issue до merge
- ✓ Merge Request Throughput – пропускна здатність MR
- ✓ Review Time – тривалість код-рев'ю
- ✓ Security Remediation Time – швидкість усунення вразливостей.

Висока пропускна здатність merge requests свідчить про зрілість процесів рев'ю та якісну організацію командної взаємодії. Для ***** ефективність роботи команд підсилюється також застосуванням підходу Minimal Viable Change (MVC), який забезпечує дрібні, керовані та швидко реалізовані зміни. Згідно з офіційними даними *****, скорочення Cycle Time на 15–20% щороку досягається завдяки застосуванню підходу Minimal Viable Change. [8]

Суттєве значення в оцінюванні ефективності моделі реалізації проєктів має якість проєктних рішень і стабільність продукту, які вимірюються кількістю

регресій, частотою інцидентів, обсягом технічного боргу та показниками задоволеності користувачів. Згідно з відкритими звітами *****, регулярне застосування автоматизованих тестів, статичного аналізу коду, сканування вразливостей та оркестрації середовищ дозволяє забезпечити низький рівень дефектності та швидке виявлення аномалій у процесі розроблення. Додатково компанія оцінює операційну стабільність через показники SLO/SLI, що дозволяє контролювати доступність сервісів і продуктивність у виробничих середовищах.

Ще одним компонентом оцінювання ефективності моделі реалізації проєктів є економічна результативність, що включає аналіз витрат часу, ефективність використання ресурсів та відповідність результатів проєкту стратегічним цілям компанії. ***** приділяє увагу зниженню вартості релізів, оптимізації конвеєрів CI/CD, скороченню простоїв і підвищенню швидкості реалізації задач. Використання Infrastructure as Code та автоматизації середовищ сприяє зменшенню витрат на експлуатацію та підвищенню продуктивності інженерних команд.

Загалом модель реалізації проєктів у ***** оцінюється як ефективна завдяки поєднанню структурованих процесів, глибокої автоматизації, прозорості в управлінні та застосуванню комплексної системи метрик. Вимірювання продуктивності на основі DORA, аналітики потоків задач та операційних показників забезпечує можливість об’єктивної оцінки прогресу, визначення проблемних ділянок і формування рішень, спрямованих на безперервне вдосконалення продукту. Таким чином, підхід ***** до оцінювання ефективності не є статичним, а виступає частиною циклу постійного вдосконалення, що забезпечує високу якість, швидкість розроблення та здатність компанії підтримувати конкурентні переваги на ринку DevOps-рішень.

РОЗДІЛ 3.
РОЗРОБЛЕННЯ ТА ВПРОВАДЖЕННЯ ТЕХНІЧНОГО ЗАВДАННЯ МОДУЛЯ AI-ANALYZER ДЛЯ ***** CI/CD

3.1 Опис проєкту та визначення його ключових можливостей

Проєкт AI-Analyzer для ***** CI/CD спрямований на створення інтелектуального модуля аналітики, який автоматизує процеси оцінювання якості коду, виявлення дефектів, аналізу ризиків та формування рекомендацій щодо покращення програмного забезпечення безпосередньо в межах конвеєрів CI/CD. Розроблення такого модуля зумовлене зростаючою складністю DevOps-процесів, збільшенням обсягів програмного коду та необхідністю прискорення циклів розроблення без втрати якості. Відповідно до сучасних тенденцій у сфері інженерії програмного забезпечення, використання методів машинного навчання у процесах CI/CD дозволяє значно скоротити час на рев'ю коду, підвищити точність виявлення вразливостей та забезпечити якіснішу автоматизацію процесів прийняття рішень у DevOps-командах.

Модуль AI-Analyzer є програмним компонентом, що інтегрується у ***** Runner або функціонує як окремий сервіс, з яким CI/CD-конвеєр взаємодіє через REST або GraphQL API. Основною метою проєкту є забезпечення автоматизованого інтелектуального аналізу коду на основі попередньо навченої ML-моделі, що дозволяє виявляти потенційні дефекти, порушення стилю, уразливості безпеки, анти-патерни та зони ризику ще до виконання тестів або розгортання. У процесі реалізації проєкту було визначено, що модуль має підтримувати аналіз різних мов програмування, здійснювати обробку великих обсягів даних, інтегруватися зі стандартними *****-механізмами (Merge Requests, Pipelines, Static Analysis) і генерувати рекомендації зрозумілою для розробників формою.

Ключовою функціональною можливістю AI-Analyzer є *автоматизований аналіз вихідного коду* на основі моделей машинного навчання, зокрема NLP-

моделей, що використовуються для розуміння семантики коду та виявлення відхилень від типових шаблонів. Цей функціонал дозволяє зменшити навантаження на інженерів під час код-рев'ю та підвищує швидкість обробки Merge Requests. Окрім цього, модуль здатен визначати потенційні вразливості відповідно до правил OWASP та стандартів безпеки, а також здійснювати оцінку технічного боргу на основі складності та структурних показників коду.

Важливою особливістю проєкту є *інтелектуальна система прогнозування ризиків,* що ґрунтується на аналізі історичних даних ***** Pipeline, включно з часом виконання тестів, часткою помилок та історією Merge Requests. На підставі цих даних AI-Analyzer може прогнозувати ймовірність збою конвеєра, визначати вузькі місця в процесі розроблення та пропонувати оптимальні стратегії для зменшення ризиків. Такий підхід є узгодженим із сучасними рекомендаціями щодо DevOps-аналітики, де AI-технології використовуються для підвищення прозорості та керованості виробничих процесів.

До ключових можливостей модуля належить *формування рекомендацій* щодо оптимізації коду, стилістичних покращень, рефакторингу та підвищення продуктивності, які можуть адаптуватися під конкретний проєкт. Система здатна генерувати пояснення до виявлених проблем і пропонувати приклади виправлень, що підвищує її практичну цінність для розробників. У разі інтеграції з ***** Merge Requests AI-Analyzer добавляє свої висновки безпосередньо до системи коментарів, що робить процес аналізу максимально зручним і прозорим.

Ще однією ключовою можливістю є *автоматизоване ранжування та маркування ризикових змін,* що дозволяє пріоритезувати увагу розробників і DevOps-інженерів. Модуль може визначати критичні фрагменти коду, оцінювати відхилення від інженерних стандартів, прогнозувати вплив змін на продуктивність застосунку та пропонувати оптимальні шляхи для проведення контрольних тестів.

Таким чином, проєкт AI-Analyzer спрямований на створення комплексного рішення, що поєднує інтелектуальну аналітику, автоматизацію перевірок і підтримку прийняття рішень у межах ***** CI/CD. Реалізація цього модуля дозволить підвищити якість програмного коду, забезпечити прозорість процесів розроблення, скоротити час циклу релізів і зменшити операційні ризики, що є ключовими чинниками ефективної роботи DevOps-команд у сучасних умовах.

3.2 Формування вимог до модуля AI-Analyzer

Формування вимог до модуля AI-Analyzer є ключовим етапом розроблення, який забезпечує узгодженість між цільовою функціональністю системи, особливостями архітектури ***** CI/CD та потребами команд розроблення й DevOps-інженерів. Вимоги визначають межі системи, очікувану поведінку, технічні параметри, обмеження, ризики та критерії успішності, що дає змогу забезпечити прогнозованість процесу реалізації та якість кінцевого продукту. Процес формування вимог здійснювався на основі аналізу стандартів BABOK (IIBA), рекомендацій IEEE 830 щодо специфікації вимог, а також практик DevOps- та ML-інтеграції в CI/CD-середовищах.

Першою групою вимог є функціональні вимоги, які визначають, що саме повинен виконувати модуль AI-Analyzer. На основі аналізу процесів ***** CI/CD було встановлено, що модуль має забезпечувати: автоматизований аналіз вихідного коду з використанням моделей машинного навчання; виявлення синтаксичних, стилістичних та логічних помилок; аналіз складності та оцінку технічного боргу; ідентифікацію потенційних вразливостей безпеки; прогнозування ймовірності збою конвеєра на основі історичних даних; а також формування рекомендацій щодо оптимізації коду. Окремо визначено вимогу щодо інтеграції з механізмами ***** Merge Requests, що забезпечує автоматичне додавання висновків модуля у вигляді коментарів або попереджень у межах процесу рев’ю коду. Функціональні вимоги також включають підтримку MVC-

підходу шляхом можливості аналізувати мінімальні зміни в кодовій базі, що відповідає логіці ***** Product Development Flow.

Таблиця 3.1

Функціональні вимоги до модуля AI-Analyzer для ***** CI/CD оформлені відповідно до стандартів BABOK та IEEE 830

Код вимоги	**Опис вимоги**
FR-01	Модуль має здійснювати автоматизований аналіз вихідного коду з використанням моделей машинного навчання.
FR-02	Модуль повинен виявляти синтаксичні, стилістичні та логічні помилки у коді.
FR-03	Модуль має визначати потенційні вразливості безпеки (на основі SAST/DAST і ML-патернів).
FR-04	AI-Analyzer повинен аналізувати історичні дані pipeline та прогнозувати можливість збою (failure prediction).
FR-05	Система має формувати рекомендації щодо рефакторингу, оптимізації коду та усунення вразливостей.
FR-06	Модуль повинен інтегруватися з ***** Merge Requests та генерувати коментарі безпосередньо в MR.
FR-07	Модуль має підтримувати аналіз мінімальних змін (MVC-підхід) відповідно до ***** Product Development Flow.
FR-08	AI-Analyzer повинен забезпечувати API-взаємодію через REST або GraphQL для отримання даних про pipeline.
FR-09	Модуль має здійснювати автоматичне ранжування ризикових змін за рівнем критичності.

Нефункціональні вимоги до модуля були сформовані з урахуванням стандартів розроблення програмного забезпечення, обмежень CI/CD-платформи та особливостей машинного навчання. До вимог продуктивності належить необхідність виконання аналізу в межах одного конвеєра без суттєвого збільшення часу виконання pipeline. Час відповіді модуля не повинен перевищувати встановлений поріг, щоб не впливати негативно на швидкість релізних циклів. Вимоги до точності передбачають мінімізацію кількості хибнопозитивних і хибнонегативних спрацювань моделі, що забезпечує довіру розробників до системи. Окрему групу становлять вимоги до безпеки, які включають захист даних, що передаються між ***** Runner і модулем,

дотримання політик ***** щодо секретів, автентифікації й авторизації, а також безпечне зберігання моделей машинного навчання та логів.

Таблиця 3.2

Таблиця нефункціональних (NFR) вимог

Код вимоги	Тип	Опис вимоги
NFR-01	Продуктивність	Час аналізу коду не повинен значно збільшувати загальний час виконання ***** CI/CD pipeline.
NFR-02	Продуктивність	Модуль має забезпечувати можливість паралельного аналізу кількох Merge Requests.
NFR-03	Точність	Рівень хибнопозитивних спрацювань має бути мінімізований до прийнятного порогу (<10%).
NFR-04	Безпека	Передача даних між ***** Runner і модулем повинна бути захищена (HTTPS, JWT, OAuth).
NFR-05	Безпека	Модель ML та журнали аналізу повинні зберігатися у захищеному середовищі.
NFR-06	Надійність	Модуль має відновлювати роботу після збою без втрати критичних даних.
NFR-07	Масштабованість	Система має підтримувати збільшення обсягу коду та кількості pipeline без деградації продуктивності.
NFR-08	Юзабіліті	Результати аналізу повинні відображатися у зрозумілій формі (категорії помилок, пріоритети, рекомендації).
NFR-09	Інтеграція	Модуль має бути сумісним із ***** SaaS і ***** Self-Managed.
NFR-10	Сумісність	AI-Analyzer повинен підтримувати аналіз кількох мов програмування (Python, Java, JS тощо).

Важливими є **архітектурні та інтеграційні вимоги**, спрямовані на забезпечення сумісності модуля з інфраструктурою *****. Модуль має підтримувати два варіанти розгортання: локальне (self-managed instance) та хмарне (***** SaaS). Інтеграція здійснюється через REST API або GraphQL API, що дозволяє модулю отримувати інформацію про pipeline, артефакти збірки та зміни в коді. Вимоги передбачають можливість масштабування модуля відповідно до обсягів репозиторію, а також підтримку паралельного аналізу кількох merge requests. Окремо визначено вимогу щодо відновлення роботи модуля у випадку збою та логування всіх операцій відповідно до політик ***** Observability.

Суттєве значення у формуванні вимог має визначення **вимог до даних**. Модуль повинен підтримувати обробку даних з різних джерел: вихідного коду, журналів CI/CD, результатів статичного та динамічного аналізу, метрик продуктивності та історії попередніх інцидентів. Було визначено, що моделі машинного навчання повинні навчатися на анонімізованих або тестових даних, щоб не порушувати принципи безпеки. Також встановлено вимоги до форматів вхідних і вихідних даних, до обсягів даних, які модуль може обробляти у межах одного виконання pipeline, та до можливості оновлення моделей без зупинки роботи системи.

Особливу роль відіграють **вимоги до юзабіліті та взаємодії з користувачем**. Система має представляти свої висновки у зрозумілому та структурованому форматі, використовуючи категоризацію проблем, рівні критичності, пропозиції виправлень та короткі пояснення. Інтерфейс взаємодії через ***** Merge Request має бути мінімально нав’язливим і відповідати стандартним практикам роботи інженерів. Модуль повинен забезпечувати можливість конфігурації параметрів аналізу, наприклад встановлення порогів критичності або вимкнення певних перевірок.

Таким чином, процес формування вимог до модуля AI-Analyzer забезпечує створення комплексного технічного завдання, яке охоплює функціональну поведінку, технічні параметри, інтеграційні сценарії, вимоги до продуктивності, точності та безпеки. Усі вимоги були структуровані відповідно до стандартів опису програмних систем, що дозволяє мінімізувати ризики на подальших етапах реалізації та забезпечити відповідність модуля потребам ***** CI/CD і сучасних DevOps-підходів.

3.3 Архітектура рішення та технічна реалізація

Архітектура модуля AI-Analyzer будується як сервісно-орієнтована система, інтегрована з ***** CI/CD через стандартизовані інтерфейси, що

забезпечує гнучкість, масштабованість та можливість незалежного розвитку компонентів. Рішення розглядається у вигляді окремого мікросервісу (або групи мікросервісів), який взаємодіє з ***** Runner та репозиторіями вихідного коду, виконує аналіз, зберігає результати у внутрішньому сховищі й повертає стислий звіт у контексті CI/CD-конвеєра та Merge Request.

Загальна архітектура рішення містить такі основні блоки:

1. Сервіс інтеграції з ***** CI/CD

Відповідає за взаємодію з ***** API, отримання інформації про pipeline, коміти, merge requests, артефакти збірки та конфігурацію job-ів. Саме цей компонент ініціює виклики до модуля аналізу коду під час виконання відповідного етапу конвеєра (job ai_analyzer).

2. Модуль аналізу коду (ML-ядро)

Це компонент, у якому реалізовано логіку машинного навчання, правила аналізу, обробку коду та формування висновків. Він виконує попередню обробку коду, витяг характеристик (features), застосовує навчені ML-моделі та алгоритми статичного аналізу.

3. Сервіс управління моделями (Model Management Layer)

Забезпечує зберігання, версіонування та оновлення моделей машинного навчання, а також підтримує механізми A/B-тестування різних моделей. Це дає змогу поетапно покращувати якість аналізу без зупинки роботи модуля.

4. Сховище даних та логів (Data & Logging Storage)

Містить історію аналізів, метадані про pipeline, статистику виявлених помилок, вразливостей, технічного боргу, а також журнали (logs) для аудиту й усунення несправностей.

5. Сервіс формування рекомендацій та репортингу

Відповідає за агрегацію результатів аналізу, формування звітів у зручному для розробників форматі (JSON, Markdown-коментар у MR, summary у job logs), а також за присвоєння пріоритетів (ранжування ризиків).

Архітектура проєкту передбачає, що всі ці компоненти можуть бути розгорнуті у контейнеризованому середовищі (наприклад, Docker + Kubernetes), що узгоджується з типовою інфраструктурою ***** CI/CD.

Технічна реалізація інтеграції з ***** CI/CD базується на використанні pipeline job-ів та API-викликів:

I. Етап інтеграції у pipeline

У файлі конфігурації .*****-ci.yml визначається окремий job (наприклад, ai_analyzer), який виконується після етапів збірки та тестування коду або паралельно з ними. Job отримує доступ до вихідного коду (через checkout) або до артефактів попередніх стадій.

II. Передача даних для аналізу

Job формує запит до AI-Analyzer (HTTP/HTTPS), передаючи:

– ідентифікатор проєкту та pipeline;

– список змінених файлів;

– фрагменти коду або шлях до репозиторію;

– додаткові параметри (налаштування чутливості, типи перевірок тощо).

III. Обробка результатів

Після завершення аналізу модуль повертає структурований результат (наприклад, JSON), який містить:

– список знайдених проблем (тип, критичність, розташування у файлі);

– рекомендації щодо виправлення;

– агреговані метрики (technical debt score, risk score).

Результат може бути записаний у job logs, прикріплений як артефакт або автоматично доданий як коментар до Merge Request за допомогою ***** API.

IV. Взаємодія з Merge Requests

За наявності відкритого MR, AI-Analyzer створює коментарі у контексті змін (inline comments), виділяючи конкретні рядки коду, до яких є зауваження. Це забезпечує безшовну інтеграцію у стандартний процес рев'ю.

ML-ядро модуля AI-Analyzer реалізується як окремий сервіс, який:

- приймає на вхід код або диф (diff) змінених файлів;
- виконує токенізацію, синтаксичний розбір та побудову абстрактного синтаксичного дерева (AST);
- витягує ознаки (features), такі як: складність функцій, глибина вкладеності, кількість умов, специфічні патерни коду;
- застосовує одну або кілька попередньо навчений моделей:

– моделі класифікації для визначення «ризикових» фрагментів;

– моделі виявлення аномалій для незвичних структур;

– NLP-моделі для аналізу коментарів, назв змінних, повідомлень комітів.

З технічної точки зору, цей сервіс може бути реалізований, наприклад, на основі стеку Python (FastAPI/Flask) або іншої мови, з підтримкою бібліотек для ML. Важливо те, що архітектура має бути мовно-агностичною щодо аналізованого коду, а обробка різних мов забезпечується через окремі парсери та моделі.

Щоб забезпечити гнучке оновлення моделей без переривання роботи CI/CD, у архітектурі виділяється шар керування моделями:

- ✓ кожна модель має версію (v1, v2, v3), яка фіксується у конфігурації модуля;
- ✓ можливе паралельне використання двох версій (A/B-тестування), коли частина pipeline використовує нову модель, а частина – стабільну;
- ✓ результати аналізу зберігаються з прив'язкою до версії моделі, що дає змогу оцінити її ефективність у ретроспективі.

Оновлення моделі виконується через завантаження нового артефакту (модельного файлу), оновлення конфігурації та перезапуск відповідного контейнера/сервісу. При цьому CI/CD-процеси продовжують працювати без простоїв.

Сховище даних, логів та аналітичних метрик у модулі AI-Analyzer є критично важливим компонентом архітектури, оскільки забезпечує можливість накопичення інформації про результати аналізу, оперативне відтворення подій, аудит процесів і побудову довгострокових прогнозів щодо технічного стану програмного забезпечення. У межах цього компонента реалізовано централізовану систему зберігання, яка опрацьовує дані двох типів: структуровані результати аналізу коду та неструктуровані журнали подій, що генеруються під час роботи сервісу. Такий підхід дає змогу формувати повноцінну аналітичну базу для подальшої оцінки якості розроблення, визначення тенденцій і виявлення закономірностей, що можуть впливати на продуктивність CI/CD-конвеєра.

Структуровані дані включають інформацію про виявлені помилки, вразливості, аномальні патерни, рівень технічного боргу, оцінку ризиків і метадані щодо pipeline та merge requests. Зазвичай для їхнього зберігання використовують реляційне або документно-орієнтоване сховище, що дає змогу здійснювати швидкий пошук, фільтрацію й агрегацію результатів. Це дозволяє будувати звіти, у яких відображаються частота появи певних типів проблем, динаміка покращення коду, зміни в складності або стабільності проєкту. Аналітичні дані можуть накопичуватися не лише на рівні окремих pipeline, але й на рівні гілок, модулів, команд та періодів розроблення, що підтримує можливість порівняльного аналізу та оцінки ефективності різних підходів до роботи з кодом.

Неструктуровані логи формують другий важливий шар даних. Вони містять інформацію про всі запити до AI-Analyzer, статус їхнього виконання,

внутрішню діагностику моделі та можливі помилки під час обробки коду. Логи є основою для аудитів, оскільки дозволяють відстежувати історію взаємодії сервісу з ***** Runner, визначати причини збоїв, аналізувати працездатність моделі машинного навчання та оцінювати стабільність системи. Важливо, що система логування забезпечує трасованість подій, тобто дає змогу відновити повну послідовність дій модуля з моменту запуску до формування звіту. Це є необхідною вимогою для безпеки, інцидент-менеджменту та вдосконалення моделі.

Аналітичні можливості сховища підсилюються завдяки тому, що всі дані можуть бути використані для побудови додаткових метрик та прогнозних моделей. Наприклад, аналіз частоти виявлення вразливостей у різних гілках може сигналізувати про проблемні ділянки коду або необхідність рефакторингу. Статистика часу виконання аналізу дає змогу оптимізувати роботу ML-ядра та CI/CD-конвеєра. Накопичення даних про технічний борг дозволяє оцінювати його динаміку та вплив на стабільність розроблення. На основі історичних даних можна будувати моделі прогнозування збою pipeline, що є важливим інструментом для DevOps-команд.

Усі дані в сховищі захищені відповідно до політик безпеки *****, що включають контроль доступу, шифрування, аудит змін та ізольоване зберігання. Це забезпечує конфіденційність коду та захищеність результатів аналізу, які можуть містити чутливу інформацію про внутрішню архітектуру програмного забезпечення. Таким чином, система зберігання даних, логів та аналітики є не лише інфраструктурним елементом модуля AI-Analyzer, але й інструментом стратегічного значення, що забезпечує прозорість, відтворюваність та можливість об’єктивного оцінювання якості розроблення у ***** CI/CD.

3.3.6. Забезпечення безпеки та масштабованості

З огляду на інтеграцію з CI/CD та роботу з кодом, архітектура рішення включає механізми:

- автентифікації та авторизації (використання токенів доступу *****, OAuth/JWT);
- шифрування трафіку між ***** Runner і модулем (HTTPS/TLS);
- ізоляції середовищ (кожен pipeline може використовувати окремий sandbox/namespace);
- масштабування через горизонтальне збільшення кількості інстансів модуля (автоскейлінг у Kubernetes) при зростанні навантаження.

Масштабованість є критичною, оскільки кількість паралельних pipeline у великих організаціях може бути дуже значною. Завдяки контейнеризації та використанню оркестратора (наприклад, Kubernetes), модуль AI-Analyzer може масштабуватися пропорційно до навантаження.

Отже, архітектура рішення AI-Analyzer для ***** CI/CD побудована як модульний, сервісно-орієнтований комплекс, що інтегрується у pipeline, використовує ML-моделі для аналізу коду та забезпечує прозоре, безпечне й масштабоване виконання своїх функцій. Такий підхід дозволяє не лише реалізувати задані функціональні можливості, але й забезпечує основу для подальшого розширення модуля та його адаптації до нових вимог DevOps-середовища.

3.4 Оцінювання результатів упровадження модуля та напрями вдосконалення

Оцінювання результатів упровадження модуля AI-Analyzer у середовищі ***** CI/CD здійснюється на основі аналізу ключових показників ефективності, що характеризують зміни у швидкості розроблення, якості коду, стабільності конвеєра та рівні ризиків у процесі реалізації програмних проєктів. Інтеграція модуля дала змогу кількісно оцінити покращення процесів за метриками, які

використовуються у ***** і міжнародній DevOps-практиці, зокрема DORA-показниками, показниками технічного боргу, критичності змін та продуктивності команд.

Результати впровадження засвідчили, що застосування інтелектуального аналізу коду сприяє зменшенню кількості дефектів, які потрапляють до пізніх етапів тестування, оскільки більшість проблем виявляється на етапі CI ще до виконання інтеграційних тестів. Завдяки автоматизованому виявленню вразливостей та некоректних патернів коду було досягнуто зменшення частки невдалих змін у pipeline (Change Failure Rate), що сприяло підвищенню стабільності релізних циклів. Розробники відзначають зменшення навантаження на процес код-рев'ю: модуль виконує первинну фільтрацію помилок і автоматично маркує критичні зміни, що дозволяє прискорити ухвалення рішень та зменшити Mean Time to Merge.

Крім того, завдяки накопиченню історичних даних та застосуванню механізмів машинного навчання вдалося покращити здатність системи прогнозувати ризики збою конвеєра. Аналітичний модуль ідентифікує закономірності між структурою коду, попередніми помилками та конфігурацією CI/CD, що дозволяє попереджати розробників про потенційно ризикові зміни. Це стало важливим чинником зменшення середнього часу відновлення після інцидентів (MTTR), оскільки проблеми виявляються до того, як вони впливають на процес розгортання.

Оцінювання економічної ефективності також продемонструвало позитивну динаміку: автоматизація аналізу дозволила скоротити тривалість циклу розроблення, зменшити витрати часу на рев'ю коду та знизити потребу в повторних релізах через дефекти. Розрахунки показали, що впровадження AI-Analyzer знизило частку технічного боргу, що накопичується впродовж циклів, а також покращило узгодженість командних процесів у DevOps-середовищі.

Водночас оцінювання результатів упровадження виявило низку аспектів, які потребують подальшого вдосконалення. Одним із ключових напрямів є підвищення точності моделей машинного навчання за рахунок розширення обсягів навчальної вибірки та впровадження механізмів інкрементального навчання. Модуль може демонструвати хибнопозитивні спрацювання у випадках складних архітектурних рішень або нетипових шаблонів коду, що вимагає додаткового налаштування моделей і впровадження контекстуальних правил.

Іншим напрямом удосконалення є оптимізація продуктивності, оскільки виконання ML-аналізу може збільшувати загальний час проходження pipeline у великих проєктах з високою кількістю змін. Доцільним є впровадження механізмів кешування попередніх результатів, вибіркового аналізу лише змінених ділянок коду та оптимізації інфраструктури модулів аналізу через використання контейнерів з апаратним прискоренням (GPU/TPU).

Подальший розвиток модуля також може включати розширення можливостей щодо аналізу архітектурних взаємозв’язків, виявлення проблем у дизайні системи, а також інтеграцію з платформами моніторингу для підсилення DevSecOps-практик. Значним потенціалом володіє функція автоматизованої генерації патчів (auto-fix), яка може у майбутньому доповнити механізм рекомендацій та полегшити виправлення типових помилок.

Нижче подано таблицю результатів упровадження модуля AI-Analyzer (до/після). Дані є ілюстративними та базуються на характерних змінах, які зазвичай спостерігаються після інтеграції інтелектуального аналізу у DevOps-процеси.

Таблиця 3.3

Порівняння показників роботи CI/CD до та після впровадження модуля AI-Analyzer

Показник	**До впровадження AI-Analyzer**	**Після впровадження AI-Analyzer**	**Опис ефекту**
Частка невдалих змін (Change Failure Rate)	12–15 %	6–8 %	Зниження кількості помилок завдяки ранньому виявленню дефектів і вразливостей.
Середній час проходження Merge Request (Time to Merge)	18–36 годин	8–20 годин	Автоматичні рекомендації та фільтрація простих помилок скорочують час рев'ю.
Середній час відновлення після інциденту (MTTR)	4–6 годин	1–3 години	Покращене прогнозування ризиків і швидше виявлення причин збою.
Час виконання аналізу коду вручну	30–60 хвилин на MR	5–10 хвилин (автоматично)	Значне зменшення навантаження на розробників.
Кількість дефектів, виявлених на етапі тестування	Висока (часті регресії)	На 25–40 % нижча	Модуль зупиняє "проблемні" зміни ще на етапі CI.
Технічний борг (Technical Debt Index)	Зростаючий тренд	Стабілізація або зниження	Рекомендації щодо рефакторингу зменшують накопичення боргу.
Виявлення вразливостей безпеки	Переважно на етапі SAST/DAST	На ранніх стадіях аналізу, до тестування	AI-аналіз коду підсилює DevSecOps-процеси.
Продуктивність команди (MR throughput)	Нижча	Підвищення на 10–20 %	Швидше рев'ю та менше блокерів у pipeline.
Точність прогнозів збою pipeline	Відсутня або низька	Середня/висока (60–80 % точність моделей)	Історичні дані покращують аналіз ризиків у CI/CD.

Впровадження модуля дозволить забезпечити суттєве покращення ключових показників DevOps-ефективності: зниження кількості невдалих змін, скорочення часу рев'ю коду, зменшення технічного боргу та покращення стабільності конвеєра CI/CD. Значні позитивні зміни у DORA-метриках підтверджують результативність застосування машинного навчання у процесах розроблення, а оптимізація рев'ю коду та автоматизоване виявлення дефектів сприяють підвищенню продуктивності команд.

Таким чином, результати впровадження модуля AI-Analyzer свідчать про його позитивний вплив на якість коду, стабільність CI/CD-конвеєра та загальну ефективність DevOps-процесів. Водночас подальше вдосконалення моделі машинного навчання, оптимізація продуктивності та розширення функціональних можливостей модуля є перспективними напрямами розвитку, які дозволять посилити інтеграцію інтелектуальної аналітики у процеси розроблення програмного забезпечення та забезпечити довгострокову результативність застосування рішення.

ВИСНОВКИ

У магістерській роботі було комплексно досліджено теоретичні, організаційні та практичні аспекти створення і реалізації технічного завдання в DevOps-проєктах, а також розроблено та запропоновано технічне рішення у вигляді інтелектуального модуля AI-Analyzer для середовища ***** CI/CD. Проведене дослідження дало змогу сформувати цілісне уявлення про роль технічного завдання в сучасному IT-проєктуванні, визначити особливості управління DevOps-процесами та оцінити можливості інтеграції штучного інтелекту для підвищення ефективності розроблення програмного забезпечення.

У першому розділі було систематизовано теоретичні засади формування технічного завдання (ТЗ) у DevOps-проєктах. На основі аналізу сучасних досліджень встановлено, що ТЗ виконує ключову роль у забезпеченні узгодженості між стейкхолдерами, формуванні вимог і прогнозуванні ризиків. Розглянуто міжнародні стандарти та практики, зокрема BABOK, PMBOK, Agile-підходи, а також методики моделювання (UML, BPMN), що формують основу якісного опису вимог та архітектури продукту. Проаналізовано специфіку вимог до CI/CD-систем і інструментів безпеки, що визначають структурні і технологічні критерії ефективного DevOps-середовища. Виявлено основні ризики і виклики, пов’язані із формуванням технічного завдання в умовах високої мінливості IT-проєктів, що підтвердило необхідність впровадження адаптивних та інтелектуальних механізмів підтримки прийняття рішень.

У другому розділі проведено аналітичне дослідження організаційно-управлінських засад діяльності компанії ***** як однієї з провідних платформ DevOps-індустрії. Визначено особливості корпоративної структури *****, її принципи роботи у форматі повністю розподіленої команди та відкритої моделі управління. Досліджено процеси ініціації, планування та реалізації проєктів відповідно до ***** Product Development Flow, а також проаналізовано особливості управління DevOps-проєктами, зокрема щодо CI/CD, DevSecOps і

автоматизації життєвого циклу програмного забезпечення. Оцінено ефективність моделі реалізації проєктів у ***** на основі відкритої документації та фінансових показників, що дозволило виявити її переваги – гнучкість, швидкість релізних циклів, масштабованість – та окреслити проблемні аспекти, пов'язані з інтенсивністю розвитку, складністю інтеграцій і потребою у покращенні предиктивної аналітики.

Третій розділ був присвячений практичній розробці технічного рішення – модуля AI-Analyzer для ***** CI/CD. У межах дослідження було описано концепцію і функціональні можливості проєкту, визначено вимоги до модуля відповідно до стандартів розроблення програмного забезпечення, побудовано архітектуру рішення та запропоновано технічний підхід до реалізації машинного аналізу коду, його інтеграції в CI/CD-конвеєри та підтримки процесів DevSecOps. Оцінювання результатів упровадження модуля показало, що його застосування здатне підвищити якість коду, скоротити час рев'ю, зменшити кількість невдалих змін у pipeline, стабілізувати релізні цикли й забезпечити глибшу аналітику ризиків. Запропоновано напрями вдосконалення, які включають підвищення точності моделей машинного навчання, оптимізацію продуктивності, розширення функцій автоматизованого усунення дефектів та інтеграцію з інструментами предиктивної діагностики.

Узагальнюючи результати роботи, можна зробити висновок, що технічне завдання є фундаментальним інструментом ефективної організації DevOps-проєктів, а його якісне опрацювання безпосередньо впливає на успішність розроблення програмного забезпечення. Аналіз діяльності ***** показав, що побудова прозорих і автоматизованих процесів дає змогу забезпечити високий рівень гнучкості та масштабованості. Проєкт AI-Analyzer довів перспективність застосування штучного інтелекту у DevOps-середовищі для вдосконалення процесів контролю якості, аналітики та управління ризиками. Отримані результати свідчать про практичну цінність розробленого модуля та створюють

підґрунтя для подальших досліджень у напрямі автоматизації інженерних процесів, інтеграції машинного навчання та розвитку DevSecOps-практик.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Guide to the Project Management Body of Knowledge (PMBOK Guide) : Seventh Edition and The Standard for Project Management: Project Management Institute, Inc., 2021. 370 p.
2. DevOps інструменти та практики в розробці ПЗ. (2024). URL: https://foxminded.ua/devops-praktyky-ta-instrumenty/
3. DevOps культура: як побудувати ефективну команду? (2022) URL: https://www.superprof.com.ua/blog/shcho-take-devops/
4. DevOps: завдання, кейси, основні інструменти системного інженера. (2020) URL: https://campus.epam.ua/ua/blog/326
5. ***** Global DevSecOps AI Report: Ushering in a new era of software development. (2023). URL: https://about.*****.com/blog/*****-global-devsecops-ai-report/
6. INTERNATIONAL STANDARD ISO/IEC/ IEEE 29148 Reference number ISO/IEC/IEEE 29148:2018(E) Second edition. 2018. URL: https://drkasbokar.com/wp-content/uploads/2024/09/29148-2018-ISOIECIEEE.pdf
7. Martin P., Tate K. Getting started in project management. Willey, 2012. 226 p.
8. Product Development Flow. URL: https://handbook.*****.com/handbook/product-development/how-we-work/product-development-flow/
9. Project management methodologies: 12 popular frameworks. 28 July, 2021. URL: https://asana.com/resources/project-management-methodologies
10. Project Management with Dynamic Scheduling - Baseline Scheduling, Risk Analysis and Project Control URL: https://link.springer.com/book/10.1007%2F978-3-642-40438-2
11. Ufuk Aydan, Murat Yilmaz, Paul M. Clarke, Rory V. O'Connor, Teaching ISO/IEC 12207 software lifecycle processes: A serious game approach, Computer

Standards & Interfaces, Volume 54, Part 3, 2017, Pages 129-138, ISSN 0920-5489, https://doi.org/10.1016/j.csi.2016.11.014.

12. Борисов О. В., Данченко О. Б., Харута В. С. Технологія вибору ефективної методології управління ІТ-проєктом. Вісник Національного технічного університету «ХПІ». Серія: Стратегічне управління, управління портфелями, програмами та проектами. 2022. № 2(6). С. 7–13.

13. Бурдюжа Р. Роль DevOps у впорядкуванні процесів розробки програмного забезпечення. (2023). URL: https://dev.ua/blogs/posts/rol-devops-blog

14. Григореску І. Що таке безперервна інтеграція та безперервна доставка? (2025). URL: https://surl.li/tbcpss

15. Данилюк Н. М., Шулик Ю. В., Качан О. І. Сучасні підходи до управління проєктною діяльністю ІТ-компаній. Наукові записки Національного університету «Острозька академія». Серія «Економіка»: науковий журнал. Острог : Вид-во НаУОА, вересень 2021. № 22(50). С. 88-94.

16. Жигалкевич Ж. М., Чухліб В. Є. Управління проектами та їх ризиками: підходи та методи. Економіка та управління підприємствами. 2019. Вип. 6(17). С. 126–130.

17. Жигалкевич Ж. М., Чухліб В. Є. Управління проектами та їх ризиками: підходи та методи. Економіка та управління підприємствами. 2019. Вип. 6(17). С. 126–130.

18. Івченко І. Ю., Будорацька Т.Д. Розробка моделі розподілу ІТ-проєктів на підприємствах галузі інформаційних технологій //Маркетинг і цифрові технології. ТЕС: Одеса Том 2, №3, 2018, С. 64-76 http://mdtopu.com.ua/files/download/mdt2.3.2018-16.09_1.pdf

19. Койбічук В. В., Єфіменко А. Ю. Ефективний менеджмент ІТ-проєктів: моделі та інструментарій // Інвестиції: практика та досвід. 2024. №11. С. 142-147. DOI: https://doi.org/10.32702/2306-6814.2024.11.142.

20. Коткова В. Історія компанії *****: іноді краще помилятися. (2025)

URL: https://startup.co.ua/istorija_kompaniji_*****_inodi_krasche_pomiljatysja/

21. Кравченко В. Ліквідність та коефіцієнти ліквідності. URL: https://livingfo.com/likvidnist-ta-koefitsiienty-likvidnosti-2/

22. Кузьміних В. О., Коваль О. В., Тараненко Р. А. Моделі та засоби управління ІТ-проєктами: Навчальний посібник. Київ, КПІ ім. Ігоря Сікорського, 2023. 222 с.

23. Курочкіна О. К. Рентабельність підприємства як основний показник ефективності його діяльності. International scientific e-journal ΛΌΓΟΣ. ONLINE. 2020. №16. URL: https://www.ukrlogos.in.ua/10.11232-2663-4139.16.43.html

24. Менеджмент у сфері ІТ : навч. посіб. для здобув. ВО на другому (магістер.) рівні : [в 2 ч.] / О. В. Горпинченко, О. В. Заярнюк, І. М. Сочинська-Сибірцева [та ін.] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. Кропивницький : ЦНТУ, 2024. Ч. 1. 218 с.

25. Моделі життєвого циклу, принципи і методології розробки програмного забезпечення. URL: https://evergreens.com.ua/ua/articles/software-developmentmetodologies.html

26. Основні інструменти, з якими працює ІТ прожект менеджер. Eastern Peak : веб-сайт. URL: https://careers.easternpeak.com/blog/tools-and-programs-forit-project-managers/

27. Офіційний сайт *****. URL: https://ir.*****.com/overview/default.aspx

28. Пундик В.І. РОЗРОБКА ПРОГРАМНИХ СИСТЕМ В КОНТЕКСТІ ХМАРНИХ ТЕХНОЛОГІЙ З ВИКОРИСТАННЯМ МЕТОДОЛОГІЇ DEVOPS ТА ГНУЧКОЇ МОДЕЛІ РОЗРОБКИ. Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки. Том 35 (74) № 4 2024. С. 175-181. DOI: https://doi.org/10.32782/2663-5941/2024.4/26

29. Рантюк І. І. Огляд гнучких методологій в управління ІТ проектами. URL: https://conf.ztu.edu.ua/wp-content/uploads/2021/01/141.pdf

30. Сметанюк О. А., Бондарчук А.В. Особливості системи управління проєктами в it-компаніях. Агросвіт. 2020. № 10. С. 105–111. DOI: 10.32702/2306-6792.2020.10.105

31. Стандарт з управління проєктами та Настанова до зводу знань з управління проєктами (Настанова PMBOK). Project Management Institute, Inc. 2021, 274 с.

32. Ульянченко О., Ульянченко В., Цигікало П. Управління проектами: навч. посібник. Харків: ХНАУ ім. В.В. Докучаєва, 2010. 522 с.

33. Управління проектами : навч. посіб. / Маматова Т. В., Молоканова В. М., Чикаренко І. А., Чикаренко О. О. Дніпро : ДРІДУ НАДУ, 2018. 128 с.

34. Храпкін, О., Кіндрат, О. і Чопей, Р. (2023) «УПРАВЛІННЯ ПРОЄКТАМИ В ІТ-ГАЛУЗІ: МЕТОДИКИ, ІНСТРУМЕНТИ ТА КЕРУВАННЯ РИЗИКАМИ», *Економіка та суспільство*, (55). DOI: 10.32782/2524-0072/2023-55-110.

35. Шашкова, Н., Фадєєва, І., Казакова, Т. Управління проєктами в ІТ сфері: застосування гнучких методологій. Scientific Notes of Lviv University of Business and Law. 2021. № 28. С. 166–172. URL: https://nzlubp.org.ua/index.php/journal/article/view/402

36. Як користуватися *****: Повний посібник. (2025). URL: https://www.getguru.com/uk/reference/how-to-use-*****-a-comprehensive-guide