Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

ДОСЛІДЖЕННЯ ТА РОЗРОБКА СТРАТЕГІЇ УПРАВЛІННЯ РИЗИКАМИ В ЖИТТЄВОМУ ЦИКЛІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

RESEARCH AND DEVELOPMENT OF A RISK MANAGEMENT STRATEGY IN THE SOFTWARE LIFE CYCLE

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерні науки»

Виконав: здобувач вищої освіти
групи КНмз-21
Рубльов Віктор Володимирович
_____
(підпис)
Керівник: к.т.н., доцент
Ліщина Валерій Олександрович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Валерій Олександрович
_____
(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерних наук*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *122 Комп'ютерні науки*
Освітня програма: *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Валерій ЛІЩИНА
«14» травня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Рубльов Віктор Володимирович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Дослідження та розробка стратегії управління ризиками в життєвому циклі програмного забезпечення»
Керівник к.т.н., доцент Ліщина Валерій Олександрович
затверджені наказом закладу вищої освіти від «*14*» *травня* *2025* р. № 255/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«05» грудня 2025 р.*
3. Вихідні дані до роботи: _____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, експерементальне дослідження результативності предмету дослідження.
_____
_
5. Перелік графічного матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблематики за темою роботи та постановка завдань дослідження	*Ліщина В. О.*		
Теоретичне дослідження та практична реалізація предмету дослідження	*Ліщина В. О.*		
Експерементальне дослідження результативності предмету дослідження	*Ліщина В. О.*		
Показник запозичень тексту	_____ %		
Інструментальна перевірка	*Кошелюк В. А.*		
Нормоконтроль	*Сачук В. О.*		
Гарант ОПП	*Ліщина В. О.*		

7. Дата видачі завдання *«14» травня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Провести огляд літературних джерел по темі кваліфікаційної роботи*	*до 30.06.2025 р*	
2	*Провести аналіз загальної проблеми і вибір напрямків дослідження*	*до 01.09.2025 р.*	
3	*Розробити функціональну схему роботи програмного продукту*	*до 01.10.2025 р*	
4	*Описати засоби розробки об'єкта проектування*	*до 15.10.2025 р.*	
5	*Практична реалізація об'єкта проектування*	*до 10.11.2025 р.*	
6	*Провести експерементальне дослідження результативності предмету дослідження*	*до 25.11.2025 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*до 05.12.2025 р.*	

Здобувач вищої освіти _____ Віктор РУБЛЬОВ

Керівник роботи _____ Валерій ЛІЩИНА

АНОТАЦІЯ

Рубльов В. В. Дослідження та розробка стратегії управління ризиками в життєвому циклі програмного забезпеченняє. Рукопис

Кваліфікаційна робота магістра ОП «Комп'ютерні науки». Луцький національний технічний університет. Луцьк, 2025.

Робота присвячена дослідженню та розробленню стратегії управління ризиками в життєвому циклі програмного забезпечення. На основі аналізу міжнародних стандартів (ISO/IEC 12207, ISO 15504, ISO 31000), сучасних методів ідентифікації, аналізу та оцінювання ризиків сформовано інтегровану стратегію, яка охоплює всі етапи життєвого циклу: від формування вимог до супроводу та виведення програмного продукту з експлуатації.

Запропоновано підхід до кількісного оцінювання ризикових втрат на основі цільової функції, що враховує ймовірність настання ризиків, вартість збитків і вагові коефіцієнти пріоритезації. Розроблено інформаційну систему підтримки стратегії управління ризиками, яка забезпечує ведення реєстру ризиків, їх пов'язування з процесами життєвого циклу, моніторинг показників та формування звітності для менеджерів проекту. На основі експериментальних розрахунків показано, що використання розробленої стратегії дає змогу суттєво знизити сумарну вартість ризикових втрат та підвищити керованість проєктів розробки програмного забезпечення.

Ключові слова: ризик, управління ризиками, життєвий цикл програмного забезпечення, стратегія, оптимізація, вартість ризикових втрат, інформаційна система.

ABSTRACT

Viktor Rublov. Research and development of a risk management strategy in the software life cycle. Manuscript.

Master's thesis in Computer Science. Lutsk National Technical University. Lutsk, 2025.

The thesis is devoted to the research and development of a risk management strategy for the software life cycle. Based on the analysis of international standards (ISO/IEC 12207, ISO 15504, ISO 31000), as well as modern methods for risk identification, analysis and assessment, an integrated strategy is proposed that covers all stages of the software life cycle: from requirements engineering to maintenance and decommissioning.

The thesis introduces a quantitative approach to modeling risk losses using an objective function that takes into account the probability of risk occurrence, the cost of losses and weighting coefficients for prioritization. An information system supporting the proposed risk management strategy has been designed and implemented. It provides a risk register, linkage of risks to life-cycle processes, monitoring of key indicators, and reporting for project managers. Experimental results demonstrate that the proposed strategy significantly reduces total risk-related losses and improves controllability of software development projects.

Keywords: risk, risk management, software life cycle, strategy, optimization, risk loss cost, information system.

ЗМІСТ

ВСТУП

Актуальність теми. Індустрія розробки програмного забезпечення характеризується зростанням складності систем, багатством функціональних вимог та високою залежністю бізнес-процесів від якості програмних продуктів. Це призводить до зростання кількості ризиків, пов’язаних з порушенням термінів, перевищенням бюджету, зниженням якості та безпеки даних. В умовах поширення Agile- та DevOps-підходів особливої ваги набуває безперервне управління ризиками протягом усього життєвого циклу програмного забезпечення, а не тільки на етапі планування проєкту.

Міжнародні стандарти ISO/IEC 12207 та ISO 15504 регламентують процеси життєвого циклу програмних засобів і вказують на необхідність систематичного управління ризиками як окремого процесу, інтегрованого в усі інші процеси створення та супроводу програмного забезпечення. Водночас практика показує, що в багатьох проєктах управління ризиками обмежується складанням реєстру ризиків та матриці «ймовірність-наслідки», без чіткого зв’язку з конкретними процесами життєвого циклу та без кількісного оцінювання впливу обраних заходів реагування.

Сучасні загрози кібербезпеці, вимоги до безперервності бізнесу та зростаюча регуляторна відповідальність ще більше підсилюють потребу у формуванні цілісної стратегії управління ризиками, яка:

– охоплює всі етапи життєвого циклу програмного забезпечення;

– спирається на стандартизовані процеси та метрики;

– підтримується інформаційною системою для моніторингу й аналізу ризиків у режимі, наближеному до реального часу [1].

Отже, дослідження та розроблення стратегії управління ризиками в життєвому циклі програмного забезпечення є актуальним завданням, спрямованим на підвищення якості, надійності та економічної ефективності ІТ-проєктів.

Мета кваліфікаційної роботи полягає у дослідженні процесів управління ризиками в життєвому циклі програмного забезпечення та розробленні інтегрованої стратегії управління ризиками, що забезпечує зниження вартості ризикових втрат і підвищення результативності проєктів розробки програмного забезпечення.

Об’єкт дослідження – процес управління ризиками в життєвому циклі програмного забезпечення, який охоплює ідентифікацію, аналіз, оцінювання, планування реагування, моніторинг та перегляд ризиків на всіх етапах створення та експлуатації програмних продуктів.

Предмет дослідження – моделі, методи, показники та програмні засоби формування і реалізації стратегії управління ризиками в життєвому циклі програмного забезпечення, включаючи алгоритми кількісного оцінювання ризикових втрат та інформаційну систему підтримки прийняття рішень.

Методи дослідження. У роботі використано методи системного аналізу, порівняльного аналізу міжнародних стандартів, методи якісного та кількісного аналізу ризиків (матриці ризиків, експертні методи, імітаційне моделювання, оптимізаційні підходи), методи математичного моделювання та експериментальних розрахунків, а також засоби проєктування інформаційних систем.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- виконати аналіз предметної області управління ризиками в життєвому циклі програмного забезпечення та узагальнити існуючі підходи до оцінювання ризиків;

- провести огляд та класифікацію існуючих методів і засобів керування ризиками, включаючи програмні продукти та стандартизовані процеси, і визначити їхні переваги та обмеження;

- проаналізувати міжнародні стандарти ISO/IEC 12207, ISO 15504, ISO 31000 щодо вимог до процесу управління ризиками та його інтеграції в життєвий цикл програмного забезпечення;

- сформувати концептуальну модель стратегії управління ризиками в життєвому циклі програмного забезпечення, визначити етапи, ролі, інформаційні потоки та ключові метрики;

- розробити метод кількісного оцінювання ризикових втрат на основі цільової функції, що враховує ймовірності настання ризиків, вартість збитків та вагові коефіцієнти пріоритезації, та обґрунтувати вибір параметрів цієї функції;

- спроєктувати та реалізувати інформаційну систему підтримки стратегії управління ризиками в життєвому циклі програмного забезпечення;

- розробити методику експериментального дослідження результативності запропонованої стратегії та провести розрахункові експерименти для обраного програмного проєкту;

- виконати аналіз отриманих результатів, оцінити економічну ефективність застосування стратегії та сформулювати практичні рекомендації щодо її впровадження в IT-проєктах.

Наукова новизна отриманих результатів полягає в такому:

– запропоновано інтегровану стратегію управління ризиками в життєвому циклі програмного забезпечення, яка поєднує вимоги стандартів ISO/IEC 12207, ISO 15504 та концепцію ризик-орієнтованого менеджменту згідно з ISO 31000;

– розроблено удосконалену цільову функцію для кількісного оцінювання ризикових втрат, що враховує вагові коефіцієнти пріоритезації видів діяльності та дозволяє оптимізувати розподіл ресурсів між заходами реагування;

– сформовано концептуальну модель інформаційної системи підтримки стратегії управління ризиками, яка забезпечує відображення ризиків у розрізі процесів життєвого циклу та підтримує моніторинг динаміки показників ризику.

Практичне значення отриманих результатів полягає в тому, що розроблена стратегія та інформаційна система можуть бути використані для підвищення ефективності управління ризиками в реальних проєктах розробки програмного забезпечення, зменшення вірогідності критичних відхилень за термінами,

бюджетом і якістю, а також для удосконалення процесів планування та контролю в IT-компаніях.

РОЗДІЛ 1

АНАЛІЗ ПРОБЛЕМАТИКИ УПРАВЛІННЯ РИЗИКАМИ В ЖИТТЄВОМУ ЦИКЛІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми

Оцінювання якості програмного забезпечення в життєвому циклі традиційно здійснюється з двох взаємопов’язаних позицій: з точки зору досягнення цільових характеристик (функціональність, надійність, продуктивність, зручність використання тощо) та з точки зору ризиків, які супроводжують створення і використання програмних систем. Під ризиком у контексті проєкту розробки програмного забезпечення розуміють негативну подію невизначеної природи, що може вплинути на цілі проєкту; величина ризику часто розглядається як математичне очікування можливих збитків.

Між характеристиками якості та ризиками існує тісний взаємозв’язок: підвищення якості програмного продукту, як правило, супроводжується зменшенням ризиків, тоді як накопичення невирішених ризиків призводить до деградації якості. Тому системи управління якістю та системи управління ризиками доцільно розглядати як взаємодоповнювальні компоненти єдиної системи управління життєвим циклом програмного забезпечення [2].

Розвиток ІТ-індустрії призвів до ускладнення проєктів розробки: збільшилась кількість зацікавлених сторін та інтеграцій із зовнішніми системами; значно зросли вимоги до інформаційної безпеки та захисту персональних даних; поширилися гнучкі підходи до розробки, у яких процес зміни вимог є нормою, а не винятком.

У таких умовах ризики виникають на всіх етапах життєвого циклу: на етапі аналізу вимог – ризики неповноти, суперечливості та неправильного розуміння потреб замовника; на етапі проєктування – ризики помилкових архітектурних рішень, які зумовлюють проблеми масштабованості та підтримуваності; на етапі реалізації – ризики дефектів коду, порушення стандартів кодування, зниження продуктивності; на етапі тестування – ризики недостатнього покриття тестами,

пропуску критичних дефектів; на етапі впровадження та експлуатації – ризики збоїв, порушення безпеки, відмови користувачів від системи [3].

Міжнародний стандарт ISO/IEC 12207 описує життєвий цикл програмного забезпечення як сукупність процесів, робіт і операцій, що охоплюють повний шлях системи – від формування вимог до зняття з експлуатації. Оновлені редакції стандарту включають процес управління ризиками як окремий процес, інтегрований в основні, допоміжні та організаційні процеси життєвого циклу.

На рисунку 1.1 показано місце процесу управління ризиками в узагальненій моделі життєвого циклу програмного забезпечення. Процес управління ризиками не виділяється як окремий етап, а пронизує всі стадії життєвого циклу, забезпечуючи зворотний зв’язок між постановкою вимог, проєктуванням, реалізацією, тестуванням, впровадженням і супроводом. Така інтерпретація відповідає підходу ISO/IEC 12207, де управління ризиками розглядається як інтегрований процес [4].

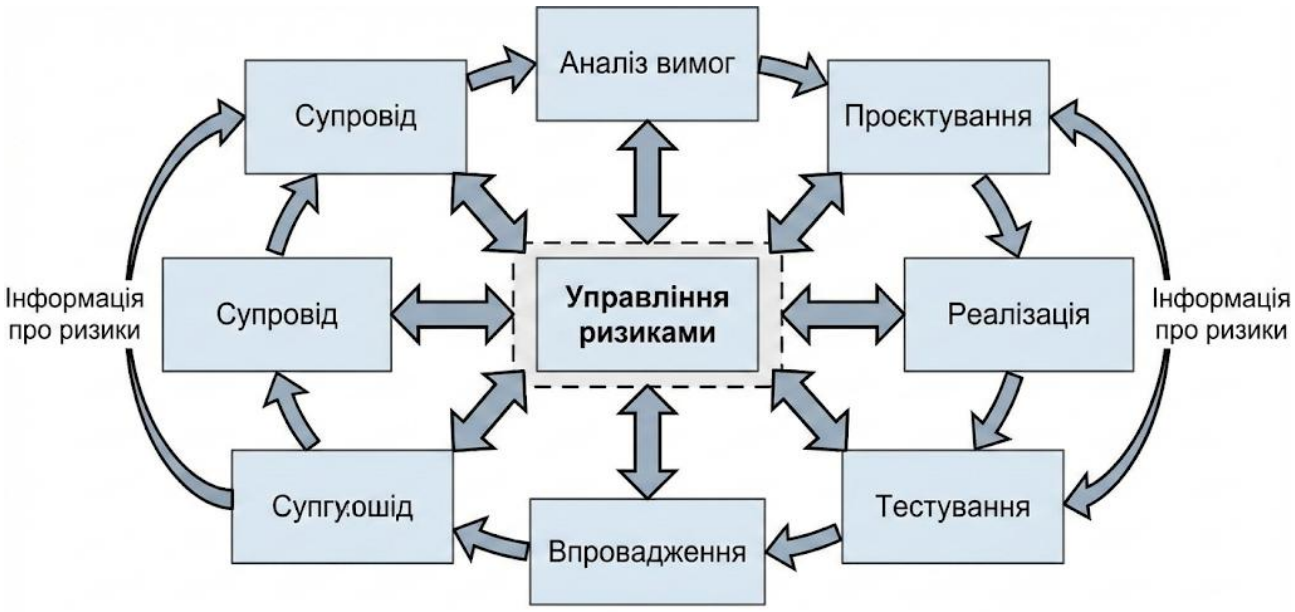

Рисунок 1.1 – Місце процесу управління ризиками в життєвому циклі програмного забезпечення за ISO/IEC 12207 [4]

Стандарт ISO 15504 (SPICE) доповнює цю картину, вводячи модель оцінювання зрілості процесів, у тому числі процесу управління ризиками. У

ньому визначено вимоги до ідентифікації, аналізу, планування реагування та моніторингу ризиків, а також наведено рекомендації щодо використання метрик ефективності процесу.

Однак на практиці в багатьох організаціях управління ризиками зводиться до одноразового заповнення реєстру ризиків на початку проєкту і нерегулярного його оновлення. Часто відсутні формалізований зв’язок між ризиками та процесами життєвого циклу; прозора методика кількісного оцінювання ризикових втрат; інструменти для інтеграції інформації про ризики з інструментами керування проєктами (системами керування задачами, CI/CD, сервісами моніторингу) [5].

На рисунку 1.2 наведено приклад ієрархічної класифікації ризиків у життєвому циклі програмного забезпечення. На верхньому рівні виділяються організаційні, технічні, процесні та зовнішні ризики. На наступних рівнях деталізуються, наприклад, технічні ризики як ризики архітектури, якості коду, продуктивності, безпеки, сумісності, а процесні – як ризики планування, комунікацій, зміни вимог, тестування.

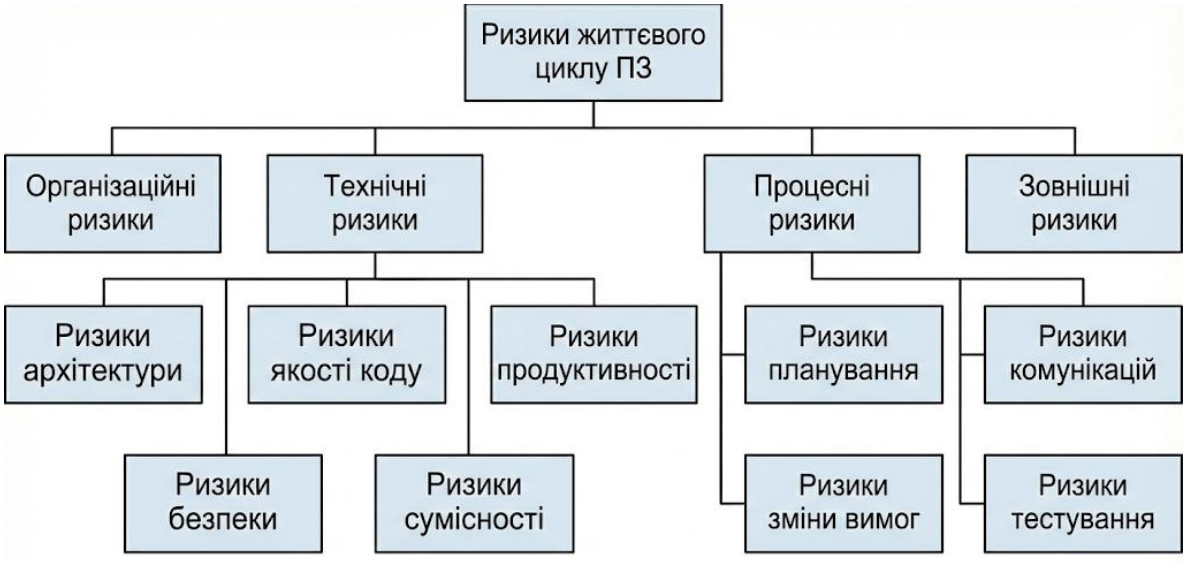

Рисунок 1.2 – Класифікація ризиків у життєвому циклі програмного забезпечення [6]

Ієрархічний вигляд класифікації дозволяє пов’язувати окремі види ризиків із відповідними процесами життєвого циклу та використовувати цю структуру як довідник під час ідентифікації ризиків у конкретних проєктах.

Це створює передумови для розроблення стратегії управління ризиками, орієнтованої саме на життєвий цикл програмного забезпечення, в якій: ризики класифікуються відповідно до етапів життєвого циклу та видів діяльності; для кожного етапу визначається набір типових ризиків та рекомендованих заходів реагування; використовуються узгоджені метрики та цільові показники, що дають змогу порівнювати результати між різними проєктами.

Таким чином, аналіз предметної області показує, що управління ризиками повинно розглядатися як безперервний, стандартизований та інструментально підтриманий процес, інтегрований у всі етапи життєвого циклу програмного забезпечення [7].

1.2 Огляд і аналіз методів та засобів розробки для вирішення проблеми дослідження

Управління ризиками в ІТ-проєктах реалізується за допомогою поєднання організаційних процедур, методів аналізу ризиків та програмних інструментів. Для розроблення стратегії управління ризиками доцільно проаналізувати існуючі методи ідентифікації, аналізу та оцінювання ризиків, а також програмні продукти, що підтримують ці процеси.

До найпоширеніших методів ідентифікації ризиків належать:

– метод мозкового штурму, який дозволяє швидко сформувати широкий перелік потенційних ризиків за участю команди проєкту та експертів;

– метод Delphi, що передбачає анонімне опитування групи експертів у кілька ітерацій з метою досягнення узгодженої думки щодо переліку та пріоритетів ризиків;

– метод аналізу основних причин, спрямований на виявлення причинно-наслідкових зв’язків між подіями та концентрування зусиль на усуненні кореневих причин ризиків;

– SWOT-аналіз, який дозволяє розглянути ризики через призму сильних і слабких сторін, можливостей і загроз;

– метод номінальних груп та методи експертних оцінок, які застосовуються за умов обмеженої статистичної інформації й спираються на досвід фахівців.

Якісний аналіз ризиків зазвичай виконується на основі матриць «ймовірність-наслідки», де експертно оцінюються ймовірність настання ризику та ступінь його впливу на цілі проєкту (незначний, середній, критичний). На цій основі визначаються пріоритети для подальшого планування реагування [8].

Кількісний аналіз ризиків використовує різні математичні підходи, зокрема:

– метод Монте-Карло для моделювання розподілу можливих результатів за наявності невизначеності у вхідних параметрах;

– байєсовські мережі довіри для моделювання імовірнісних залежностей між ризиками та подіями;

– метод критичного шляху та аналіз резервів для оцінювання впливу ризиків на терміни виконання проєкту.

Огляд програмних засобів показує, що існує широкий спектр рішень для управління ризиками:

– Essential ERM – спеціалізоване програмне забезпечення для управління ризиками підприємства, яке підтримує структури ризик-апетиту, теплові карти ризиків, формування звітів для ради директорів;

– TeamMate+ Audit – комплексне рішення для підтримки процесів аудиту, що дозволяє виявляти та оцінювати ризики, планувати аудиторські перевірки та відстежувати усунення виявлених проблем;

– KYC Portal – система для автоматизації процесів перевірки клієнтів та оцінювання ризику з використанням динамічних параметрів і правил;

– AdaptiveGRC – платформа для управління ризиками та дотриманням нормативних вимог, яка інтегрується з ISO 31000 та підтримує ведення централізованого реєстру ризиків;

– TOPIA – система, орієнтована на управління технологічними ризиками в IT-інфраструктурі, що аналізує взаємодію застосунків та операційних систем [9].

Більшість розглянутих продуктів забезпечує ведення реєстру ризиків та категоризацію, налаштування матриць ризиків, формування звітів та візуалізацію (теплові карти, діаграми).

Таблиця 1.1 узагальнює найпоширеніші моделі життєвого циклу програмного забезпечення з позиції управління ризиками. Для кожної моделі наведено характерні особливості роботи з ризиками, типові сильні й слабкі сторони, а також доцільні сфери застосування.

Таблиця 1.1 – Порівняння моделей життєвого циклу ПЗ з точки зору управління ризиками

Модель ЖЦ ПЗ	Особливості управління ризиками	Переваги	Обмеження
Каскадна	Аналіз ризиків переважно на початку проєкту	Простота планування, чіткі етапи	Пізнє виявлення багатьох ризиків, низька гнучкість
Спіральна	Ризик-орієнтовані ітерації, кожен виток включає аналіз	Раннє виявлення, можливість перегляду рішень	Складність управління, вища вартість планування
Інкрементна	Ризики оцінюються для кожного інкременту	Гнучкість, поступова доставка цінності	Можлива фрагментарність аналізу ризиків
Agile (Scrum тощо)	Ризики інтегруються у беклог і планування спринтів	Постійний зворотний зв’язок, швидка реакція	Ризик недооцінки стратегічних і зовнішніх ризиків
DevOps	Акцент на експлуатаційні та інфраструктурні ризики	Безперервний моніторинг, швидке виправлення	Потреба у зрілій інфраструктурі та культурі

За даними таблиці видно, що жодна модель не усуває ризики повністю. Радше вони по-різному розподіляють акценти: традиційні моделі концентруються на ризиках вимог і архітектури, гнучкі – на ризиках швидких змін, DevOps – на експлуатаційних і безпекових ризиках. Це обґрунтовує

необхідність стратегії управління ризиками, яка здатна адаптуватися до обраної моделі ЖЦ.

Водночас виявлено низку обмежень: відсутність «тонкої» інтеграції з моделями життєвого циклу програмного забезпечення (ISO/IEC 12207), тобто ризики не завжди прив’язуються до конкретних процесів і робіт; недостатня увага до особливостей ітеративної та інкрементальної розробки (Agile, DevOps); обмежені можливості кількісної оптимізації розподілу ресурсів між заходами реагування на ризики [10].

В таблиці 1.2 наведено порівняльний огляд ключових міжнародних стандартів, що регламентують управління ризиками в проєктах розробки програмного забезпечення.

Таблиця 1.2 – Міжнародні стандарти управління ризиками в ПЗ

Стандарт	Основна сфера застосування	Роль щодо ризиків
ISO/IEC 12207	Процеси ЖЦ програмного забезпечення	Визначає місце процесу управління ризиками
ISO/IEC 15504	Оцінювання зрілості процесів (SPICE)	Регламентує цілі, результати і метрики процесу ризиків
ISO 31000	Загальні принципи управління ризиками	Надає рамковий підхід до ризик-менеджменту
ISO/IEC 27005	Управління ризиками інформаційної безпеки	Орієнтується на ризики ІБ та захисту даних

Включення цих стандартів у порівняльний аналіз дозволяє чітко відмежувати загальні принципи ризик-менеджменту від специфічних вимог, що стосуються процесів розробки й експлуатації програмного забезпечення [11].

На рисунку 1.3 схематично показано відмінності між якісним та кількісним підходами до аналізу ризиків. Для якісного аналізу характерне використання вербальних шкал і експертних оцінок, тоді як кількісний аналіз спирається на числові значення ймовірностей, розподіли та математичні моделі.

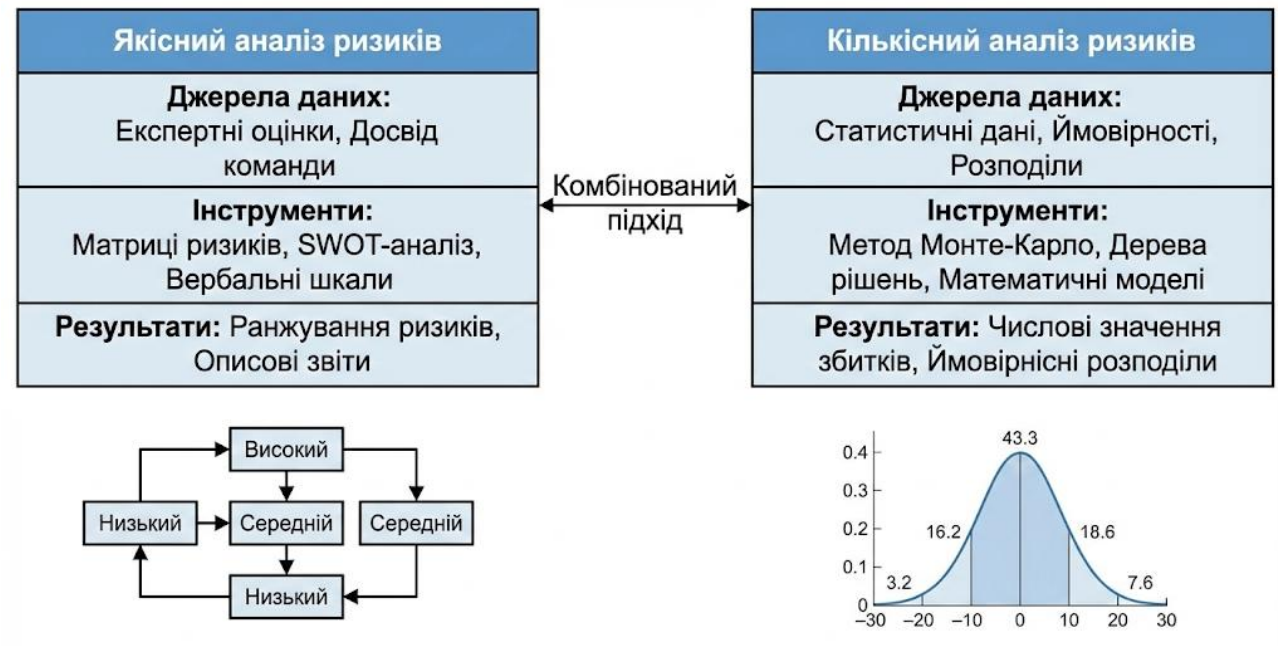

Рисунок 1.3 – Порівняння якісного та кількісного аналізу ризиків

Таким чином, аналіз існуючих методів і засобів показує, що, незважаючи на наявність розвинених інструментів, залишається актуальною задача розроблення стратегії управління ризиками, яка безпосередньо інтегрується в процеси життєвого циклу програмного забезпечення, підтримує як якісний, так і кількісний аналіз ризиків, забезпечує можливість оптимізації розподілу ресурсів на основі формалізованої цільової функції [12].

1.3 Постановка завдання на кваліфікаційну роботу магістра

На основі проведеного аналізу предметної області, методів і засобів управління ризиками сформульовано загальну науково-практичну задачу кваліфікаційної роботи: розробити та дослідити стратегію управління ризиками в життєвому циклі програмного забезпечення, яка забезпечує зниження вартості ризикових втрат та підвищення ефективності виконання ІТ-проєктів.

- виконати огляд літературних джерел та нормативних документів, присвячених управлінню ризиками в життєвому циклі програмного забезпечення;

- проаналізувати існуючі методи ідентифікації, аналізу та оцінювання ризиків, а також програмні продукти для підтримки управління ризиками;

- дослідити вимоги міжнародних стандартів ISO/IEC 12207, ISO 15504 та ISO 31000 до організації процесу управління ризиками в IT-проєктах;

- розробити концептуальну модель стратегії управління ризиками в життєвому циклі програмного забезпечення, визначивши основні етапи, інформаційні потоки, ролі та показники ефективності;

- сформулювати цільову функцію та математичну модель для кількісного оцінювання ризикових втрат і оптимізації розподілу ресурсів між заходами реагування на ризики;

- спроєктувати архітектуру та інформаційну структуру інформаційної системи підтримки стратегії управління ризиками в життєвому циклі програмного забезпечення;

- реалізувати прототип інформаційної системи, що забезпечує ведення реєстру ризиків, їх зв’язок із процесами життєвого циклу, формування звітів і візуалізацію показників;

- розробити методику експериментального дослідження ефективності запропонованої стратегії та провести розрахункові експерименти з оцінювання зниження вартості ризикових втрат і зміни показників ризику до та після впровадження стратегії;

- виконати обробку та аналіз результатів експерименту, сформулювати висновки та рекомендації щодо практичного застосування результатів роботи.

Сформульовані завдання повністю корелюють із метою роботи, відповідають структурі кваліфікаційної роботи магістра і забезпечують логічний перехід від теоретичного аналізу до побудови моделі, реалізації програмного прототипу та експериментальної перевірки результативності розробленої стратегії.

Розділ 2

Теоретичне дослідження та практична реалізація стратегії управління ризиками в життєвому циклі програмного забезпечення

2.1 Моделі життєвого циклу програмного забезпечення та вбудований процес управління ризиками

Життєвий цикл програмного забезпечення визначають як послідовність етапів, робіт і операцій, які регламентують управління розробкою на всіх стадіях, починаючи від формування технічного завдання і завершуючи відключенням програмного продукту від експлуатації. Стандарт ISO/IEC 12207 став першою спробою структурної стандартизації життєвого циклу програмного забезпечення на основі регламентації вимог до процесів, завдань і робіт, що входять до стандартної структури життєвого циклу [13].

У цьому стандарті виділено основні, допоміжні та організаційні процеси. До основних належать процеси постачальника, розробника, експлуатувальника та супроводу. Допоміжні процеси включають конфігураційне управління, верифікацію, валідацію, аудит, спільну оцінку, документацію, а також процеси забезпечення якості. Організаційні процеси охоплюють управління проектом, інфраструктурні процеси, навчання персоналу та, у нових редакціях стандарту, управління ризиками. Включення процесу управління ризиком до складу процесів життєвого циклу стало важливим кроком, який підкреслює невід’ємність ризик-менеджменту від інших процесів розробки програмного забезпечення.

Метою процесу управління ризиками в ISO/IEC 12207 є постійне виявлення та зниження ризиків протягом усього життєвого циклу. У результаті його виконання повинні бути визначені область дії управління ризиками, каталог типовий ризиків, стратегії реагування та механізми моніторингу. Стандарт вимагає не лише фіксації наперед відомих ризиків, а й систематичної ідентифікації нових ризиків, які виникають у ході виконання проекту, із подальшим аналізом, пріоритезацією та виділенням ресурсів для їх моніторингу.

Методи моніторингу мають дозволяти вчасно виявляти зміни в стані ризику та коригувати заходи реагування [14].

Стандарт ISO 15504 (SPICE) доповнює вимоги ISO/IEC 12207, запроваджуючи процесну модель оцінювання зрілості, у якій процес управління ризиками описаний як окремий процес зі своїми цілями, результатами і показниками. У відповідному розділі передбачено визначення джерел ризиків у вихідних вимогах до характеристик якості та до проєкту в цілому, аналіз і збалансування пріоритетів зниження ризиків, вибір раціональних стратегій управління, розроблення метрик для відстеження змін стану ризиків, а також оцінювання ефективності стратегії на контрольних точках життєвого циклу.

Розгляд моделей життєвого циклу програмного забезпечення з позицій ризик-орієнтованого підходу дозволяє по-різному структурувати процеси управління ризиками. Каскадна модель передбачає жорстку послідовність етапів і найчастіше призводить до того, що більшість ризиків виявляються пізно, на стадіях тестування та впровадження. У таких умовах управління ризиками має концентруватися на ранній, максимально повній ідентифікації ризиків вимог та архітектури, оскільки їх виправлення на пізніх стадіях є особливо дорогим [15].

У спіральній моделі життєвого циклу ризик-менеджмент є ключовим елементом кожного витка спіралі. Кожна ітерація включає уточнення цілей, аналіз альтернатив, оцінювання ризиків та планування робіт з урахуванням результатів такого оцінювання. Це фактично реалізує ідею безперервного циклу: ідентифікація ризиків, аналіз, вибір заходів реагування, реалізація, перегляд. Саме спіральна модель стала концептуальним прототипом для сучасних гнучких методологій, де ризики розглядаються в межах коротких ітерацій і постійного зворотного зв’язку із замовником.

Інкрементні та Agile-моделі життєвого циклу зміщують акцент на управління ризиками вимог, планування релізів, технічний борг і ризики якості. У рамках таких підходів ризики часто фігурують у беклозі нарівні з функціональними вимогами, для них оцінюється трудомісткість, пріоритет, потенційний вплив на бізнес-цілі. Управління ризиками інтегрується з

процесами планування ітерацій, забезпечення якості коду та автоматизованого тестування [16].

DevOps-орієнтовані моделі додають новий вимір – ризики експлуатації, безперервної доставки, інфраструктури та безпеки. Тут особливе значення набуває автоматизований моніторинг показників доступності, швидкості розгортання, частоти інцидентів, часу відновлення після відмови. Таким чином, управління ризиками поширюється на повний життєвий цикл, включаючи експлуатацію та супровід, а інформація про експлуатаційні ризики повертається до етапів проєктування і розробки.

Структурована стратегія управління ризиками, що пропонується в цій роботі, будується з урахуванням вимог ISO/IEC 12207 та ISO 15504 і містить три взаємопов’язані рівні. Перший рівень – організаційно-політичний, де визначаються політика управління ризиками, допустимий рівень ризику, ролі та відповідальність, загальні принципи прийняття рішень. Другий рівень – процесний, на якому ризики пов’язуються з конкретними процесами життєвого циклу і контрольними точками: постановка вимог, проєктування, реалізація, тестування, впровадження, супровід. На цьому рівні визначаються типові ризики та базові сценарії реагування для кожного етапу. Третій рівень – інструментальний, що включає методи ідентифікації та аналізу ризиків, кількісні моделі, метрики, а також інформаційну систему підтримки, через яку реалізується реєстрація, моніторинг і оптимізація [17].

На перетині процесного та інструментального рівнів формується реєстр ризиків, який є центральним елементом стратегії. Для кожного ризику фіксуються джерело, пов’язаний процес життєвого циклу, параметри ймовірності та наслідків, обрана стратегія реагування, а також динамічні показники, що характеризують зміну стану ризику в часі. На основі цього реєстру реалізується кількісне оцінювання та пріоритезація ризиків.

Кількісна основа стратегії закладається у математичну модель та цільову функцію, що відображає сумарну вартість ризикових втрат. У загальному вигляді для кожного ризику вводяться такі параметри: імовірність настання ризику,

оцінка збитку в разі його реалізації, ваговий коефіцієнт пріоритетності, який відображає важливість відповідного виду діяльності або процесу життєвого циклу. Очікувані втрати від окремого ризику визначають як добуток ймовірності на вартість наслідків з урахуванням відповідного коефіцієнта ваги. Сумарна вартість ризикових втрат визначається як сума таких добутків по всіх суттєвих ризиках. Саме ця величина виступає цільовою функцією, яку потрібно мінімізувати в результаті застосування стратегій управління ризиками [18].

Практична інтерпретація цільової функції полягає в тому, що кожен захід реагування (наприклад, посилення тестування, додаткове уточнення вимог, навчання персоналу, резервування інфраструктури) впливає або на ймовірність настання ризику, або на розмір потенційних збитків, або на обидва показники одночасно. Таким чином, змінюються значення доданків у цільовій функції, а завдання управління ризиками перетворюється на оптимізаційну задачу пошуку такого набору заходів, який за обмежених ресурсів забезпечує мінімальне значення очікуваних втрат.

На рисунку 2.1 наведено загальну структуру стратегії управління ризиками в життєвому циклі програмного забезпечення.

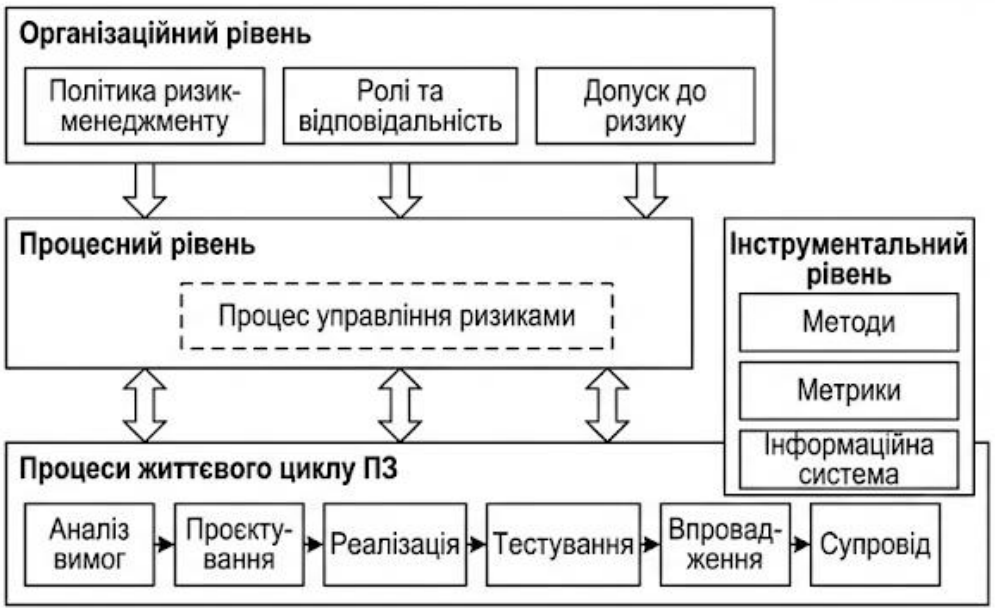

Рисунок 2.1 – Трирівнева структура стратегії управління ризиками в ЖЦ ПЗ

Стратегія подана у вигляді трьох рівнів: організаційного, процесного та інструментального. На організаційному рівні розміщуються політика ризик-менеджменту, ролі, допуск до ризику. Процесний рівень прив’язує управління ризиками до конкретних процесів життєвого циклу. Інструментальний рівень включає методи, метрики та інформаційну систему підтримки.

На рисунку 2.2 схематично подано послідовність етапів розрахунку.

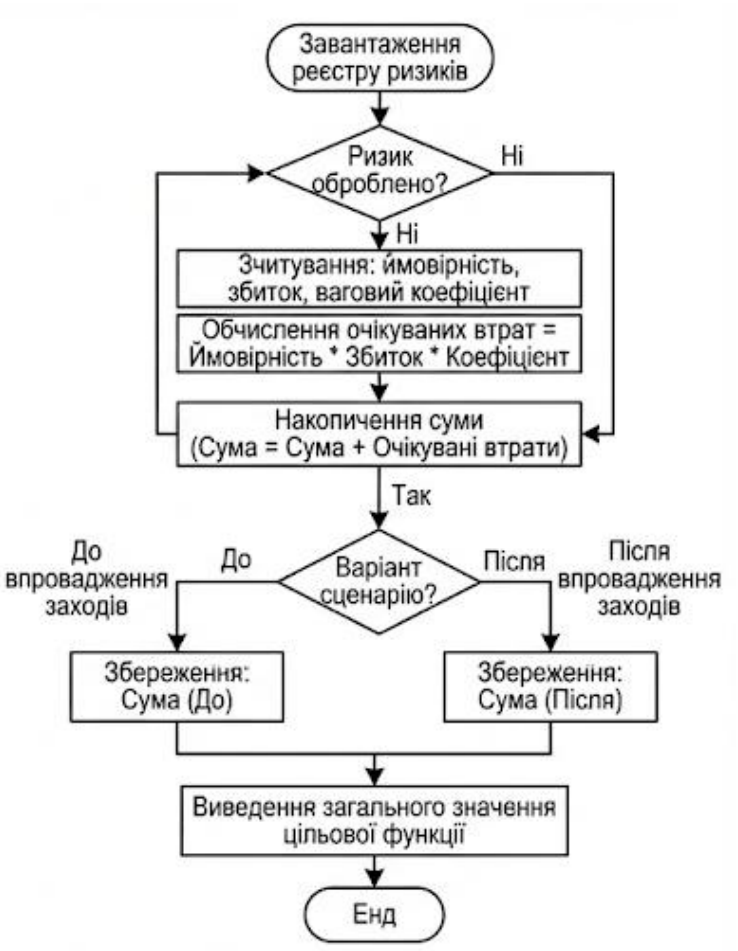

Рисунок 2.2 – Блок-схема алгоритму обчислення цільової функції вартості ризикових втрат

У блок-схемі послідовно показано такі кроки: завантаження реєстру ризиків; для кожного ризику зчитування значень імовірності, збитку та вагового коефіцієнта; обчислення очікуваних втрат як добутку цих параметрів; накопичення суми для всіх ризиків; виведення загального значення цільової

функції. Окремими гілками можуть бути показані варіанти сценаріїв до та після впровадження заходів реагування.

Окремий аспект стратегії – система метрик, що дозволяє вимірювати динаміку ризиків та ефективність обраних заходів. Відповідно до рекомендацій ISO 15504 для кожного виду ризику повинні бути визначені метрики, які відображають зміни в стані ризиків у залежності від виконуваних робіт зі зниження ризиків, а також показники, що дають змогу оцінювати успішність стратегії на контрольних точках життєвого циклу. У запропонованій стратегії до таких метрик належать очікувана вартість ризикових втрат, кількість високопріоритетних ризиків, що залишилися після впровадження заходів, частка ризиків, для яких виконано план реагування, та відносне зменшення ймовірностей ключових ризиків до і після застосування стратегії.

Таблиця 2.1 узагальнює набір основних метрик, які використовуються для оцінювання ефективності стратегії.

Таблиця 2.1 – Основні метрики оцінювання ефективності управління ризиками

Позначення метрики	Назва метрики	Короткий опис
E(L)	Очікувана вартість ризикових втрат	Сума добутків імовірності на збиток і ваговий коефіцієнт
N_high	Кількість високопріоритетних ризиків	Число ризиків у червоній зоні матриці
R_done	Частка ризиків з виконаним планом реагування	Відношення кількості оброблених ризиків до загальної
ΔE(L)	Відносне зменшення очікуваних втрат	Порівняння E(L) до і після реалізації стратегії

Завдяки формалізації цільової функції та системи метрик стратегія управління ризиками перетворюється з набору загальних рекомендацій на інструмент, придатний до кількісного аналізу, експериментальної перевірки та програмної реалізації.

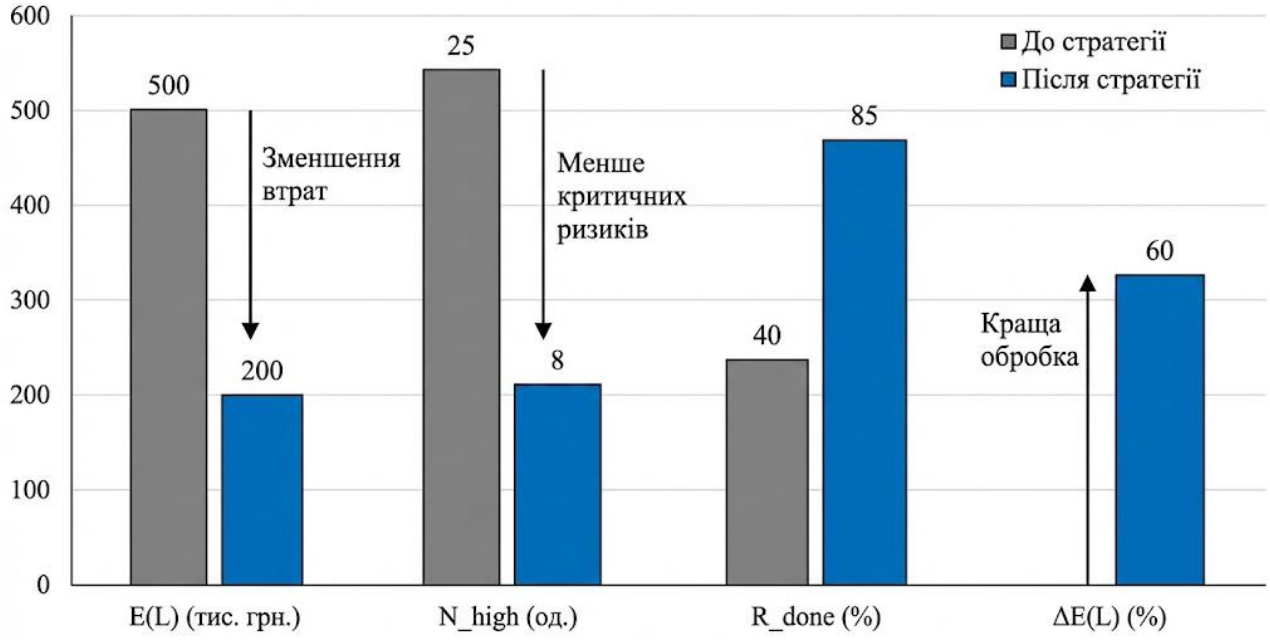

Рисунок 2.3 – Порівняння значень основних метрик до й після впровадження стратегії

На рисунку 2.3 показано приклад порівняння значень метрик до та після застосування стратегії. Чітко видно зменшення очікуваної вартості ризикових втрат і кількості високопріоритетних ризиків, а також зростання частки ризиків, для яких виконано план реагування.

2.2 Архітектура та реалізація інформаційної системи підтримки стратегії управління ризиками

Для практичної реалізації запропонованої стратегії управління ризиками в життєвому циклі програмного забезпечення розроблено інформаційну систему підтримки, яка забезпечує ведення реєстру ризиків, їх зв’язок із процесами життєвого циклу, виконання розрахунків за цільовою функцією, моніторинг метрик і формування звітів. Така система розглядається як складова інфраструктури управління проєктами розробки програмного забезпечення.

Архітектура інформаційної системи побудована за трирівневою схемою. На презентаційному рівні знаходиться веб-інтерфейс користувача, який надає доступ до реєстру ризиків, засобів фільтрації та сортування, форм введення та

редагування, панелей моніторингу та звітів. На прикладному рівні реалізовано бізнес-логіку обробки даних про ризики, включаючи обчислення показників, застосування цільової функції, формування теплових карт, генерацію таблиць і графіків для звітності. На рівні даних функціонує реляційна база даних, у якій зберігаються відомості про проєкти, процеси життєвого циклу, ризики, заходи реагування, результати оцінювання та обчислені метрики.

Прикладом реалізації прикладного рівня може бути серверний додаток на основі фреймворку Django з використанням ORM для роботи з базою даних PostgreSQL. У цьому випадку напрацьовується чітке розділення на модулі доступу до даних, модулі бізнес-логіки та модулі інтеграції з користувацьким інтерфейсом через REST-сервіси. Веб-інтерфейс може бути реалізований або засобами шаблонів серверного фреймворку, або на основі окремого клієнтського застосунку (наприклад, React), що взаємодіє з сервером через програмний інтерфейс прикладного рівня.

Логічна модель системи включає кілька основних сутностей, які відображають структуру життєвого циклу та ризиків. Кожен проєкт розробки програмного забезпечення описується окремим записом, для якого вказуються базові атрибути, такі як назва, замовник, тривалість, бюджет, обрана модель життєвого циклу. Для кожного проєкту фіксуються пов’язані процеси життєвого циклу, які узгоджуються зі стандартом ISO/IEC 12207 і охоплюють етапи постановки вимог, проєктування, реалізації, тестування, впровадження та супроводу.

Сутність ризику містить атрибути ідентифікатора, короткої та розширеної назви, опису джерела, пов’язаного процесу, категорії, а також числових параметрів, необхідних для кількісного аналізу: оцінки ймовірності, оцінки збитку, вагового коефіцієнта пріоритетності, поточного статусу. Для кожного ризику зберігаються зв’язки з одним або кількома заходами реагування, що описуються окремою сутністю: тип заходу, короткий опис, оцінка витрат на реалізацію, очікуваний ефект у вигляді зміни ймовірності чи збитку. На основі

цих даних система реалізує перерахунок очікуваних ризикових втрат та оновлення значення цільової функції.

Для відображення динаміки ризиків у часі вводиться сутність оцінювання ризику, яка включає дату, значення ймовірності до і після застосування заходів, значення очікуваних втрат та інші метрики. Її записи дозволяють будувати часові ряди та оцінювати ефективність стратегії, наприклад, шляхом порівняння суми очікуваних втрат до і після запровадження плану управління ризиками. У вихідній роботі на основі такого підходу було показано суттєве зниження ймовірностей реалізації ключових ризиків після впровадження заходів реагування, а також зменшення мінімального значення цільової функції приблизно на 81 відсоток.

Таблиця 2.2 відображає структуру бази даних системи. Для найважливіших сутностей наведено основні поля й короткий опис.

Таблиця 2.2 – Структура основних таблиць бази даних

Таблиця	Основні поля	Призначення
projects	id, name, customer, budget, model_lc	Зберігання відомостей про проєкти
processes	id, project_id, name, phase	Опис процесів ЖЦ, пов’язаних із проєктом
risks	id, project_id, process_id, title, prob, loss, weight, status	Реєстр ризиків і їх числових параметрів
actions	id, risk_id, type, description, cost, effect_prob, effect_loss	Опис заходів реагування
metrics	id, risk_id, date, prob_before, prob_after, loss_expected	Історія оцінювання ризиків
users	id, name, role	Облікові записи користувачів системи

На концептуальному рівні структура даних може бути подана у вигляді ER-діаграми, де вершинами виступають сутності «Проєкт», «Процес життєвого циклу», «Ризик», «Оцінювання ризику», «Захід реагування», «Користувач», а дуги описують зв’язки між ними. Для прикладу, один проєкт пов’язаний з багатьма ризиками, кожен ризик має багато оцінювань у різні моменти часу, а користувачі системи (менеджер проєкту, аналітик, представник керівництва) мають ролі й права доступу до різних функцій інформаційної системи.

На рисунку 2.4 наведено трирівневу архітектуру інформаційної системи підтримки управління ризиками. Виділено рівень представлення, де розташований веб-інтерфейс користувача; прикладний рівень, який реалізує бізнес-логіку, обчислення метрик і роботу з цільовою функцією; рівень даних, що містить реляційну базу.

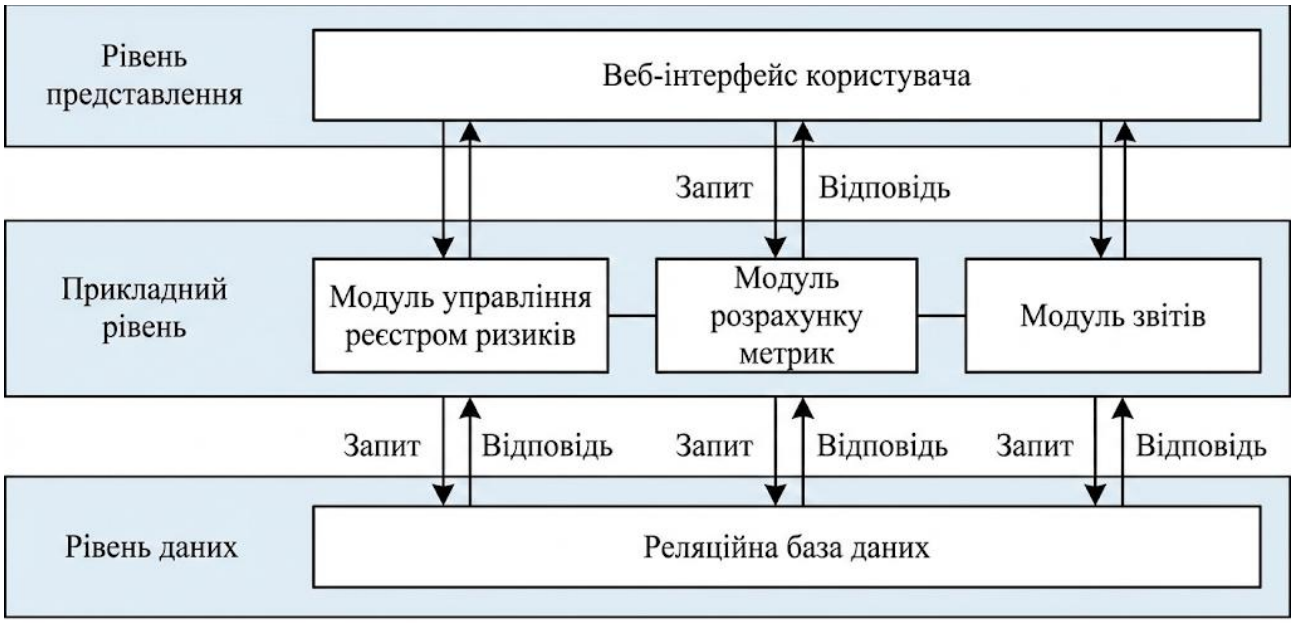

Рисунок 2.4 – Трирівнева архітектура інформаційної системи підтримки управління ризиками

Функціональність системи підтримки стратегії управління ризиками можна описати через варіанти використання. Типовий сценарій починається з того, що менеджер проєкту створює новий проєкт, вказує його основні характеристики, обирає модель життєвого циклу та задає перелік процесів. Далі аналітик з ризиків на основі проведеного аналізу ідентифікує ризики, реєструє їх у системі, визначає параметри ймовірності та збитку, обирає початкові стратегії реагування. Після цього система автоматично розраховує поточне значення цільової функції та виводить його у вигляді узагальненого показника очікуваних ризикових втрат для проєкту.

Наступні дії користувача можуть включати моделювання альтернативних варіантів розподілу ресурсів між заходами реагування. Для цього у відповідному

модулі системи задаються параметри доступного бюджету чи трудомісткості, вибираються варіанти реалізації заходів, після чого система розраховує оновлені значення ймовірностей, очікуваних втрат і цільової функції. Порівняння сценаріїв дозволяє обрати конфігурацію, яка забезпечує мінімальне значення очікуваних втрат за заданих обмежень. Результати можуть відображатися у вигляді таблиць і графіків, наприклад, графіка зниження вартості ризикових втрат залежно від обраного набору заходів.

Користувацький інтерфейс інформаційної системи орієнтовано на інтуїтивну роботу з реєстром ризиків. На головному екрані доцільно розмістити панель показників, де відображаються кількість ризиків загалом, кількість високопріоритетних ризиків, поточне значення цільової функції, а також відносна зміна цього показника порівняно з базовим станом. Нижче може бути подана таблиця реєстру ризиків з можливістю фільтрації за проєктом, процесом життєвого циклу, категорією та статусом. Окремі вікна призначені для введення та редагування ризиків, налаштування заходів реагування, перегляду історії змін показників.

Важливою складовою інтерфейсу є візуалізація ризиків у форматі матриці «ймовірність–наслідки» та теплових карт. На відповідних екранах (рис. 2.5) ризики відображаються як точки чи прямокутники, розташовані у клітинках матриці згідно з їх поточними параметрами. Це дозволяє швидко ідентифікувати області концентрації ризиків і відстежувати, як у процесі реалізації стратегії управління ризиками точки переміщуються з зон високого ризику до зон середнього або низького ризику.

Архітектурно система передбачає підтримку ролей і прав доступу. Менеджер проєкту має можливість створювати та редагувати проєкти, затверджувати плани реагування на ризики, переглядати зведені звіти. Аналітик з ризиків відповідає за наповнення реєстру ризиків, внесення оцінок ймовірностей і збитків, запуск сценаріїв моделювання. Представники керівництва отримують доступ до узагальненої інформації, включаючи звіти

щодо зниження очікуваних втрат, виконання плану реагування та рівня залишкового ризику.

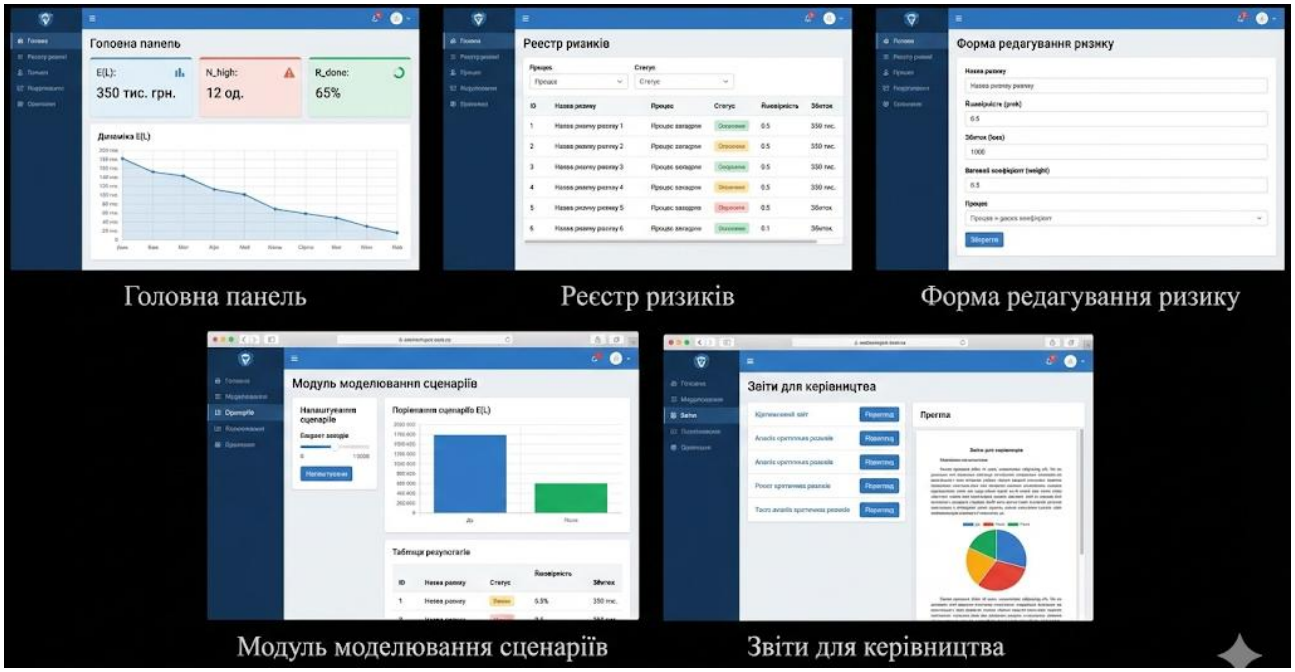

Рисунок 2.5 – Основні екрани інформаційної системи

На завершальному етапі реалізації інформаційної системи проводиться її інтеграція в процеси управління проєктами організації. За результатами експлуатації система дозволяє автоматизувати ті етапи управління ризиками, які у традиційних підходах виконувалися вручну: ведення реєстру, актуалізація параметрів ризиків, розрахунок сумарної вартості ризикових втрат, формування звітності та документування прийнятих рішень. Це підвищує прозорість процесу, зменшує ймовірність помилок і створює основу для подальшого розвитку стратегії управління ризиками в життєвому циклі програмного забезпечення.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ СТРАТЕГІЇ УПРАВЛІННЯ РИЗИКАМИ В ЖИТТЄВОМУ ЦИКЛІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Постановка експерименту та вихідні дані

Метою експериментального дослідження є перевірка результативності запропонованої стратегії управління ризиками та інформаційної системи підтримки прийняття рішень у реалістичному сценарії життєвого циклу програмного забезпечення. Передбачається, що впровадження формалізованої стратегії дає змогу зменшити очікувані ризикові втрати та раціональніше розподілити ресурси між видами превентивної діяльності.

Як експериментальний об’єкт розглянуто типовий проєкт розроблення корпоративної інформаційної системи зі строком реалізації близько дванадцяти місяців. Команда складається з аналітика, архітектора, трьох–чотирьох розробників, тестувальника та менеджера проєкту. Для такого класу проєктів на основі аналізу літератури й експертних інтерв’ю було відібрано п’ять ключових ризиків, які найчастіше впливають на дотримання термінів, бюджету та вимог замовника: невірна оцінка часових меж проєкту; недоліки тестування; недостатня взаємодія із замовником; недостатня кількість кваліфікованих співробітників; зміна вимог до підзадач у процесі реалізації.

Для кожного з ризиків було проведено опитування експертів. Групу експертів сформовано з фахівців кафедри інженерії програмного забезпечення та представників IT-компаній-партнерів. Кожний експерт оцінював ймовірність реалізації ризику та орієнтовні фінансові втрати у разі його настання. Оцінювання ймовірностей виконувалося за шкалою від нуля до одиниці, а розмір втрат задавався в умовних одиницях, що відповідають трудозатратам або перевитратам бюджету.

На основі агрегування експертних оцінок сформовано вихідні дані, наведені в таблиці 3.1.

Таблиця 3.1 – Вихідні оцінки ймовірностей ризиків та очікуваних втрат у базовому сценарії

№	Назва ризику	Позначення	Ймовірність настання P_i	Орієнтовні втрати C_i, ум. од.	Очікувані втрати $R_i = P_i \cdot C_i$, ум. од.
1	Невірна оцінка часових меж проєкту	R_1	0,3	20	6,0
2	Недоліки тестування	R_2	0,5	20	10,0
3	Недостатня взаємодія із замовником	R_3	0,4	35	14,0
4	Недостатня кількість кваліфікованих співробітників	R_4	0,3	33,3	10,0
5	Зміна вимог до підзадач проєкту у процесі реалізації	R_5	0,6	20	12,0

Очікувані втрати від і-го ризику розраховуються за співвідношенням 3.1.

$$L_i = P_i \cdot C_i,\ i = 1..5. \tag{3.1}$$

Сумарна очікувана вартість ризикових втрат у базовому сценарії без спеціальної стратегії управління визначається як 3.2.

$$E_0(L) = \Sigma\, L_i = \Sigma\, P_i \cdot C_i \approx 52 \text{ ум. од.} \tag{3.2}$$

Для переходу від оцінки втрат до безпосереднього планування робіт вводяться керувальні змінні x_i, які задають обсяг ресурсів, що виділяються на основні напрями превентивної діяльності: уточнення та валідація вимог (x_1), організація тестування й автоматизація тестів (x_2), комунікація з замовником та демонстрації проміжних результатів (x_3), внутрішній контроль якості коду, рев'ю та рефакторинг (x_4), резерв проєктного часу та бюджету для компенсації непередбачених робіт (x_5). Змінні вимірюються в людино-годинах або еквівалентних умовних одиницях ресурсу.

На основі значень L_i та аналізу впливу кожного виду діяльності на відповідні групи ризиків для моделі сформовано питомі коефіцієнти ризикових

втрат R_i, які інтерпретуються як очікувані втрати на одиницю ресурсу відповідного типу. Для базового сценарію вони дорівнюють 6, 10, 14, 10 та 12 умовних одиниць відповідно до напрямів діяльності x_1–x_5.

Цільова функція задачі оптимального розподілу ресурсів у базовому сценарії має вигляд 3.3.

$$F_0(x) = 6\,x_1 + 10\,x_2 + 14\,x_3 + 10\,x_4 + 12\,x_5 \rightarrow \min. \qquad (3.3)$$

При цьому додатково накладається система обмежень, що відображає організаційні та ресурсні характеристики проєкту 3.4:

$$\begin{gathered} x_1 \geq 1,\ x_2 \geq 15,\ x_3 \geq 1,\ x_4 \geq 1,\ x_5 \geq 1; \\ 500 \leq x_1 + x_2 + x_3 + x_4 + x_5 \leq 700; \\ x_1 + x_2 + x_3 + x_4 \leq x_5; \\ x_3 \geq x_1. \end{gathered} \qquad (3.4)$$

Перше обмеження фіксує мінімально необхідний рівень робіт за кожним напрямом, друге задає допустимий діапазон загального обсягу ресурсів, третє відображає той факт, що сумарний обсяг превентивних заходів не може перевищувати доступний резерв часу та бюджету, а четверте підкреслює важливість активної комунікації з замовником порівняно з одноразовим аналізом вимог.

Задача 3.3-3.4 розв’язувалася засобами модуля оптимізаційних розрахунків інформаційної системи підтримки управління ризиками. Для неї було обрано підхід лінійного програмування, що забезпечує пошук глобального оптимуму за заданих лінійних обмежень. У результаті розв’язання отримано базовий розподіл ресурсів, наведений в таблиці 3.2.

Таблиця 3.2 – Оптимальний розподіл ресурсів у базовому сценарії

Ресурс	Позначення x_i	Витрати ресурсу, год	Коментар
Аналіз та деталізація вимог	x_1	15	Мінімально необхідний обсяг для формалізації вимог
Організація тестування	x_2	15	Базовий рівень модульного тестування
Комунікація з замовником	x_3	15	Періодичні зустрічі та демо без систематизації
Внутрішній контроль якості коду та рефакторинг	x_4	155	Основна частина зусиль витрачається на доопрацювання
Резерв часу та бюджету	x_5	300	Буфер на виправлення помилок та позапланові роботи

Підстановка знайденого розподілу у формулу 3.3 дає значення цільової функції $F_0(x) = 5600$ умовних одиниць. Це означає, що за вихідних параметрів очікувані сукупні ризикові втрати, пов’язані з неефективним використанням ресурсів, є досить значними, а значна частина зусиль фактично витрачається на ліквідацію наслідків ризиків, а не на їх попередження.

3.2 Результати експериментальних розрахунків та їх аналіз

На другому етапі експерименту було змодельовано впровадження запропонованої стратегії управління ризиками. Ця стратегія включає уточнення вимог на ранніх етапах, впровадження систематичного тестування, регламентацію комунікацій із замовником, підвищення кваліфікації співробітників, а також формалізацію резервів часу та бюджету. Після шести місяців використання стратегії повторно проведено експертне опитування щодо ймовірностей реалізації розглянутих ризиків.

Результати повторного опитування показали суттєве зниження ймовірностей настання більшості ризиків.

У таблиці 3.3 наведено порівняння значень ймовірностей та відповідних очікуваних втрат до та після впровадження стратегії.

Таблиця 3.3 – Зміна ймовірностей та очікуваних втрат за ризиками

№	Назва ризику	P_i до впровадження	P_i після впровадження	Очікувані втрати до, ум. од.	Очікувані втрати після, ум. од.	Відносне зниження, %
1	Невірна оцінка часових меж проєкту	0,30	0,15	6,00	3,00	50,0
2	Недоліки тестування	0,50	0,10	10,00	2,00	80,0
3	Недостатня взаємодія із замовником	0,40	0,15	14,00	5,25	62,5
4	Недостатня кількість кваліфікованих співробітників	0,30	0,05	10,00	1,67	83,3
5	Зміна вимог до підзадач проєкту у процесі реалізації	0,60	0,20	12,00	4,00	66,7

Сумарна очікувана вартість ризикових втрат після впровадження стратегії становить близько 15,92 умовної одиниці, тоді як у вихідному сценарії вона дорівнювала приблизно 52 умовним одиницям. Таким чином, інтегральний показник очікуваних втрат зменшився приблизно на 69 відсотків, що свідчить про ефективність запропонованих профілактичних заходів.

На основі оновлених очікуваних втрат та врахування трудомісткості окремих видів робіт були скориговані питомі коефіцієнти R_i у цільовій функції. Вони відображають не тільки прямі втрати від реалізації ризиків, а й витрати на виконання заходів реагування. Для другого етапу експерименту ці коефіцієнти набули значень 3,0; 3,25; 3,0; 2,0; 2,25 умовних одиниць на одиницю ресурсу відповідно до напрямів x_1-x_5.

Модифікована цільова функція має вигляд 3.5.

$$F_1(x) = 3\,x_1 + 3{,}25\,x_2 + 3\,x_3 + 2\,x_4 + 2{,}25\,x_5 \rightarrow \min. \qquad (3.5)$$

Система обмежень 3.4 залишилася незмінною, оскільки часові та організаційні рамки проєкту не змінювалися. Оптимізаційний модуль інформаційної системи підтримки управління ризиками повторно розв’язав задачу 3.5 з тими самими обмеженнями, використовуючи симплекс-метод.

Розрахунки показали, що у разі використання оновлених коефіцієнтів доцільно змінити структуру розподілу ресурсів так, як наведено в таблиці 3.4.

Таблиця 3.4 – Розподіл ресурсів після оптимізації з урахуванням стратегії

Ресурс	Позначення x_i	Витрати ресурсу, год	Інтерпретація змін
Аналіз та деталізація вимог	x_1	15	Підтримується стабільний обсяг робіт з аналізу
Організація тестування	x_2	15	Зберігається мінімально допустимий рівень
Комунікація з замовником	x_3	60	Кількість годин істотно збільшено для регулярних демо й уточнення вимог
Внутрішній контроль якості коду та рефакторинг	x_4	30	Частину зусиль перенесено на більш ранні етапи
Резерв часу та бюджету	x_5	380	Резерв використовується переважно для превентивних робіт та коротких ітерацій
Разом		500	Загальний обсяг ресурсів зберігається незмінним

Підстановка значень з таблиці 3.4 у формулу (3.5) дає значення цільової функції $F_1(x) \approx 1188{,}75$ умовної одиниці. Порівняно з значенням $F_0(x) = 5600$ умовних одиниць загальна вартість ризикових втрат, що враховує як рівень ризиків, так і структуру використання ресурсів, зменшилася приблизно на 79 відсотків. При цьому загальний обсяг затрачуваних ресурсів не зріс, а лише був перерозподілений між видами діяльності.

Зменшення ризикових втрат супроводжується зміною структури профілю ризиків. Якщо до впровадження стратегії всі п’ять розглянутих ризиків належали до високопріоритетних, оскільки очікувані втрати перевищували п’ять умовних одиниць, то після впровадження лише один ризик зберіг високий статус, а решта перейшли до групи середніх або низьких. Фактично це означає, що основні загрози, пов’язані з дефіцитом кваліфікованих кадрів та комунікацією із замовником, були переведені з критичної зони у прийнятний діапазон завдяки систематичному використанню аналітичних та організаційних заходів.

Для візуалізації результатів у інформаційній системі підтримки управління ризиками сформовано декілька графічних представлень. На рисунку 3.1 подано стовпчиковий графік, на якому для кожного з п’яти ризиків показано пару значень ймовірності до та після впровадження стратегії.

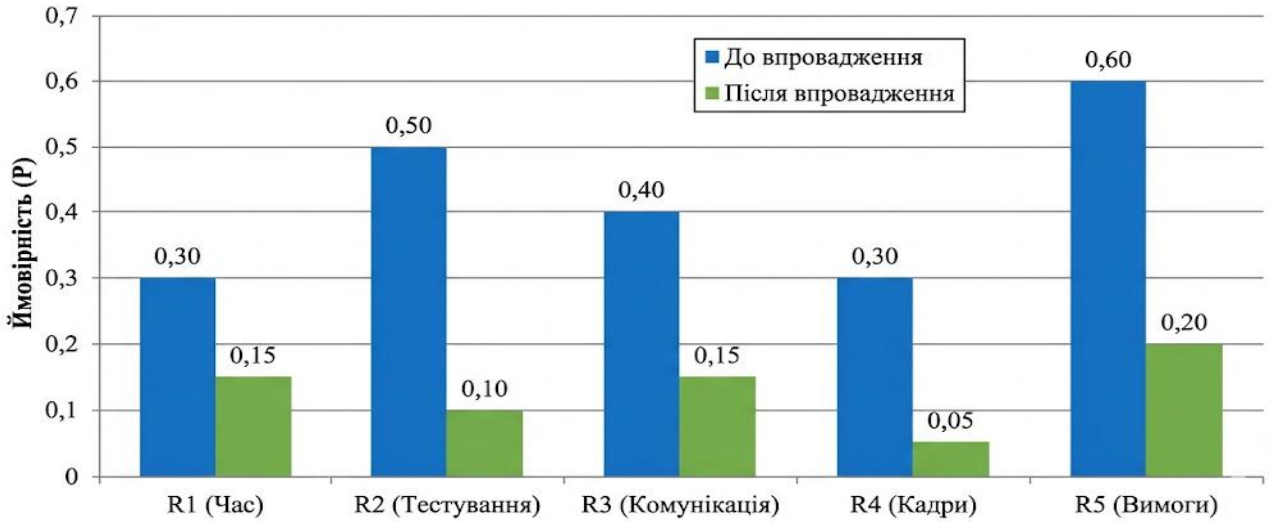

Рисунок 3.1 – Зміна ймовірностей реалізації ключових ризиків до та після впровадження стратегії

Графік наочно ілюструє істотне зменшення ймовірності реалізації ризиків тестування, кадрового дефіциту та зміни вимог, тоді як ризик невірної оцінки часових меж проєкту хоча і зменшується вдвічі, але залишається помітним.

На рисунку 3.2 наведено порівняння розподілу ресурсів між основними видами діяльності у базовому та оптимізованому сценаріях. На осі абсцис відкладено типи ресурсів x_1-x_5, а на осі ординат – обсяг витрат часу. Для кожного ресурсу побудовано по дві колонки, що відповідають базовому та оптимізованому варіантам. Графік демонструє зсув акцентів з трудомісткого виправлення помилок на ранні профілактичні дії, передусім на регулярну комунікацію із замовником та більш кероване використання резерву.

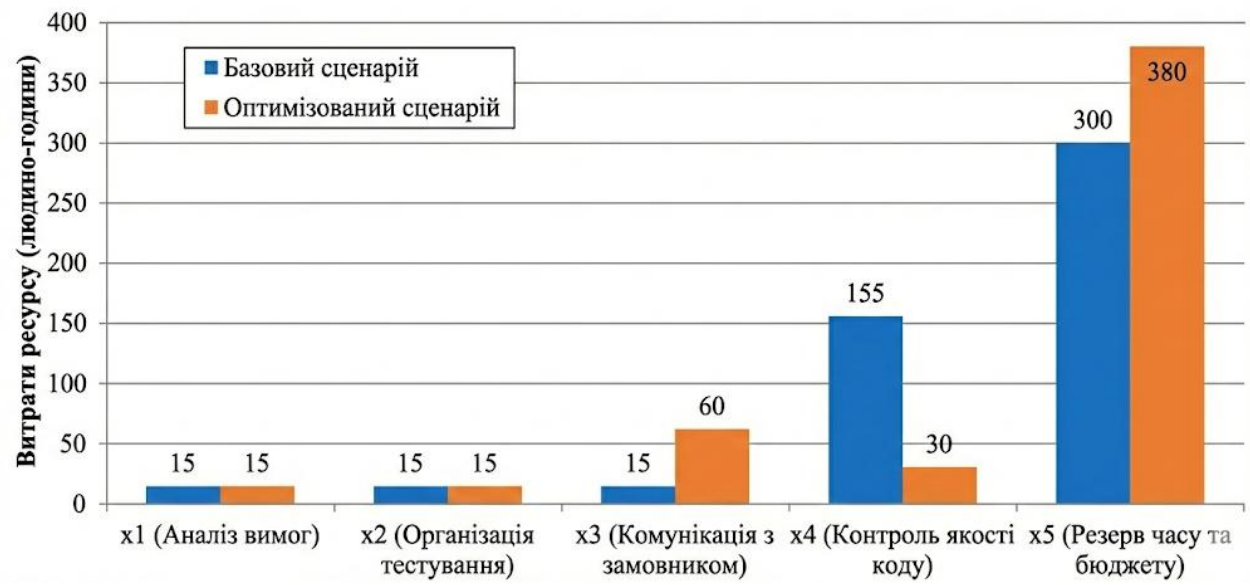

Рисунок 3.2 – Порівняння розподілу ресурсів до та після оптимізації стратегії управління ризиками

Для узагальнення результатів експерименту окремо побудовано графік інтегральних метрик, на якому порівнюються значення $E_0(L)$, $E_1(L)$ та $F_0(x)$, $F_1(x)$. Для кожної з двох груп показників відображено пару стовпчиків «до» і «після». На графіку чітко видно, що як очікувані втрати від реалізації ризиків, так і узагальнена вартість ризикових втрат з урахуванням структури ресурсів істотно зменшуються після впровадження стратегії та використання інформаційної системи.

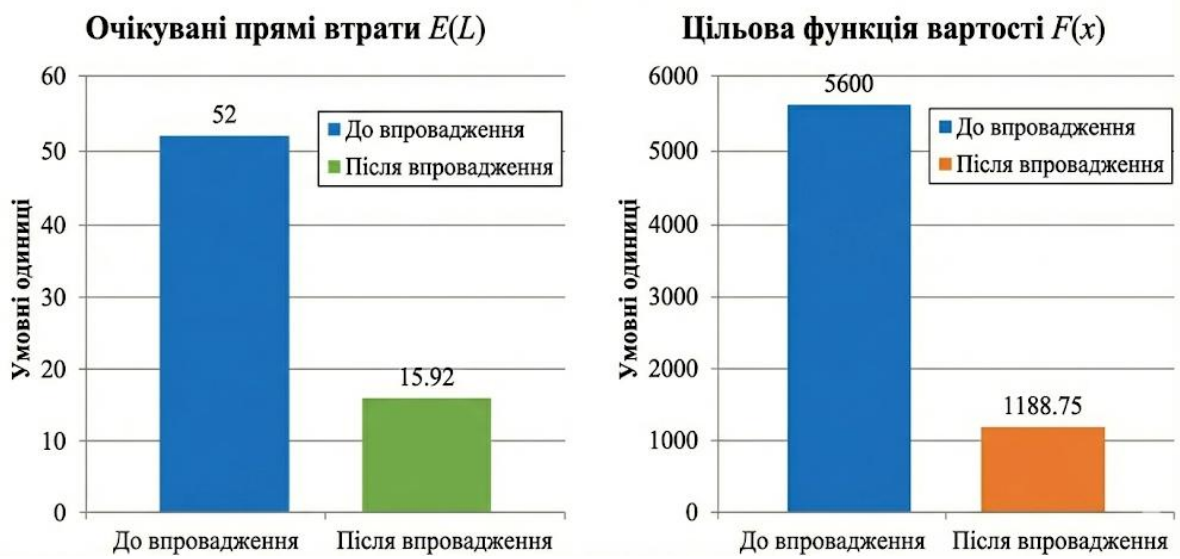

Рисунок 3.3 – Зміна інтегральних метрик ризику та вартості ризикових втрат

Проведене експериментальне дослідження показало, що розроблена стратегія управління ризиками в життєвому циклі програмного забезпечення є не лише теоретично обґрунтованою, а й практично ефективною. Комбінація формалізованої моделі оптимізації, системи показників якості управління ризиками та програмної реалізації у вигляді інформаційної системи дає змогу істотно знизити як ймовірність реалізації критичних ризиків, так і очікувані втрати, не збільшуючи сумарний обсяг задіяних ресурсів. Це підтверджує доцільність інтеграції ризик-менеджменту у всі етапи життєвого циклу програмного забезпечення та використання запропонованих підходів у практиці сучасних ІТ-проєктів.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано науково-практичну задачу дослідження та розроблення стратегії управління ризиками в життєвому циклі програмного забезпечення, орієнтованої на зниження вартості ризикових втрат і підвищення результативності ІТ-проєктів.

На основі аналізу літературних джерел, міжнародних стандартів ISO/IEC 12207, ISO 15504 та ISO 31000, а також сучасних підходів до ризик-менеджменту сформовано цілісне уявлення про місце процесу управління ризиками в життєвому циклі програмного забезпечення. Показано, що традиційні підходи часто обмежуються статичним реєстром ризиків і не забезпечують інтеграції з процесами аналізу вимог, проєктування, реалізації, тестування, впровадження та супроводу.

У роботі уточнено та формалізовано поняття ризику в контексті життєвого циклу програмного забезпечення, досліджено взаємозв’язок між характеристиками якості програмних систем та рівнем проєктних ризиків. Запропоновано класифікацію ризиків за організаційними, технічними, процесними та зовнішніми ознаками з прив’язкою до етапів життєвого циклу, що стало основою для побудови реєстру ризиків і вибору адекватних заходів реагування.

Сформовано та обґрунтовано трирівневу стратегію управління ризиками, яка охоплює організаційний рівень (політика, ролі, допустимий рівень ризику), процесний рівень (інтеграція управління ризиками в процеси життєвого циклу за ISO/IEC 12207) та інструментальний рівень (методи, метрики та інформаційна система підтримки). Такий підхід дозволяє поєднати вимоги стандартів із практичними потребами проєктних команд.

Розроблено математичну модель і цільову функцію, яка описує сумарну очікувану вартість ризикових втрат з урахуванням ймовірностей реалізації ризиків, розміру потенційних збитків і вагових коефіцієнтів пріоритетності. Показано, що задача управління ризиками може бути зведена до задачі

оптимального розподілу ресурсів між превентивними та компенсуючими заходами за умови обмеженого бюджету та часу.

Спроєктовано архітектуру та логічну структуру інформаційної системи підтримки стратегії управління ризиками, яка реалізує ведення реєстру ризиків, зв’язок ризиків із процесами життєвого циклу, обчислення показників, розв’язання задачі оптимізації та формування звітності для різних категорій користувачів. Описано структуру бази даних, основні модулі, сценарії взаємодії користувачів та елементи інтерфейсу.

Проведено експериментальне дослідження на прикладі типового проєкту розроблення корпоративної інформаційної системи. Для п’яти ключових ризиків (оцінювання строків, тестування, взаємодія із замовником, кадровий потенціал, зміна вимог) отримано вихідні експертні оцінки ймовірностей і збитків, побудовано базовий сценарій без спеціальної стратегії управління ризиками та сценарій після її впровадження. Результати розрахунків показали істотне зменшення ймовірностей реалізації ризиків, сумарної очікуваної вартості ризикових втрат та значення цільової функції. При незмінному загальному обсязі ресурсів перерозподіл зусиль у відповідності до запропонованої моделі дозволив скоротити інтегральний показник ризикових втрат майже в декілька разів і перевести більшість ризиків із критичної зони у зону прийнятних значень.

Отримані результати підтверджують гіпотезу про те, що інтеграція управління ризиками в усі етапи життєвого циклу програмного забезпечення, поєднання якісних і кількісних методів аналізу та використання спеціалізованої інформаційної системи дають змогу суттєво підвищити керованість IT-проєктів. Запропонована стратегія може бути адаптована для різних моделей життєвого циклу, включно з каскадними, спіральними, інкрементними та Agile/DevOps-підходами, і використана як методична основа для внутрішніх політик ризик-менеджменту IT-компаній.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Systems and software engineering. Life cycle processes. Risk management. URL: https://www.iso.org/standard/74371.html (дата звернення: 14.10.2025).

2. Systems and software engineering. System life cycle processes. ISO/IEC/IEEE 15288:2023. URL: https://www.iso.org/standard/82074.html (дата звернення: 14.10.2025).

3. Systems and software engineering. Software life cycle processes. Part 2. Relation and mapping between ISO/IEC/IEEE 12207:2017 and ISO/IEC 12207:2008. ISO/IEC/IEEE 12207-2:2020. URL: https://www.iso.org/standard/79486.html (дата звернення: 14.10.2025).

4. ДСТУ ISO/IEC/IEEE 15288:2025. Системна та програмна інженерія. Процеси життєвого циклу систем (ISO/IEC/IEEE 15288:2023. IDT). Київ: ДП «УкрНДНЦ». 2025. 76 с.

5. Masso J. E., Pino F. J., Pardo C., García F., Piattini M. Risk management in the software life cycle: A systematic literature review // Computer Standards Interfaces. 2020. Vol. 71. Art. 103431. DOI: 10.1016/j.csi.2020.103431.

6. Khan M. N. A., Mirza A. M., Saleem I. Software Risk Analysis with the use of Classification Techniques: A Review/Engineering, Technology Applied Science Research. 2020. Vol. 10, №. 3. C. 5678-5682. DOI: 10.48084/etasr.3440.

7. Andrade R., Sadaoui S. RM3: A Risk Management Framework for IT Project Success//Proceedings of the 2021 3rd International Conference on Computer Science and Software Engineering (CSSE 2021). Amsterdam: Atlantis Press. 2021. C. 36-40. DOI: 10.2991/978-94-6239-473-3_6.

8. Babenko V. O., Alisevych Y., Koniukh O., Nesterenko S. IT Project Risk Management Model//Distributed Computing and Artificial Intelligence, Special Sessions, 22nd International Conference. Cham: Springer. 2022. (Lecture Notes in Networks and Systems, Vol. 461). C. 219-229. DOI: 10.1007/978-3-031-04137-4_21.

9. Hopkin P., Thompson C. Fundamentals of Risk Management: Understanding, Evaluating and Implementing Effective Enterprise Risk Management. 6th ed. London: Kogan Page. 2021. 512 p.

10. O'Regan G. Risk Management//O'Regan G. Guide to Software Project Management. Cham: Springer. 2025. С. 223-252. DOI: 10.1007/978-3-031-41927-2_10.

11. Грабіна К. В., Шендрик В. В. Метод управління ризиками IT-проєктів з врахуванням загроз та можливостей//Управління розвитком складних систем. 2023. № 55. С. 18-28. DOI: 10.32347/2412-9933.2023.55.18-28.

12. Назарова К., Парасій-Вергуненко І., Остапець А. Класифікація ризиків компаній IT-індустрії//Scientia Fructuosa. 2023. № 4. С. 59-68. DOI: 10.36074/scientia-fructuosa-2023.04.09.

13. Решетняк О. І., Данько Н. І., Юрченко О. К. Управління ризиками IT-проєктів: сутність та особливості застосування підходів PMBoK і Agile//Business Inform. 2023. № 10. С. 36-374. DOI: 10.32983/2222-4459-2023-10-366-374.

14. Грицюк П. Ю., Іванишин А. В., Грицюк Ю. І. Забезпечення якості програмного продукту за стандартом IEEE 730-2014 в межах життєвого циклу реалізації проєкту // Науковий вісник НЛТУ України. 2023. Т. 33, № 2. С. 94-101. DOI: 10.36930/40330215.

15. Лях І. М., Кіш Ю. В. Особливості управління ризиками під час життєвого циклу тестування програмного забезпечення//Поліграфія і видавнича справа. 2023. № 2 (86). С. 71-78. DOI: 10.32403/0554-4866-2023-2-86-71-78.

16. Ліщина Н. М., Бойко Л. С., Гульчук Ю. М. Оцінка ризиків та їх вплив на життєвий цикл розробки програмного забезпечення//Herald of Khmelnytskyi National University. Technical Sciences. 2024. Вип. 343, № 6(1). С. 141-145. DOI: 10.31891/2307-5732-2024-343-6-21.

17. Kolisnichenko O., Kolomytsev M., Nosok S. Software security risk management in DevOps methodology//Theoretical and Applied Cybersecurity. 2021. Vol. 3, Iss. 1. С. 75-77.

18. Kovalchuk O. I., Samilo A. V., Zhuk I. M., Kalinych V. S. DevOps methodology for risk management of IT projects//Актуальні проблеми пожежної безпеки та запобігання надзвичайним ситуаціям: матеріали міжнар. наук.-практ. конф. Харків: НУЦЗУ. 2024. С. 123-128.

ДОДАТКИ

ДОДАТОК А

Реєстр ризиків експериментального проєкту

ID	Процес ЖЦ ПЗ	Категорія	Опис ризику	Основні причини	Можливі наслідки	Ймовірність P_i	Вплив C_i, ум. од.	Рівень ризику	Стратегія реагування	Відповідальний	Статус
R1	Планування, ініціація	Процесний	Невірна оцінка часових меж проєкту	Нестача історичних даних, оптимістичні припущення	Зрив дедлайнів, перевитрата бюджету	0,30	20	Високий	Збір статистики попередніх проєктів, резервування часу	Менеджер проєкту	Активний
R2	Тестування	Технічний	Недоліки тестування, низьке покриття тестами	Обмежений час, відсутність автоматизації	Вихід у продуктив з критичними дефектами	0,50	20	Високий	Впровадження тест-дизайну, автоматизованих тестів	Провідний тестувальник	Активний
R3	Аналіз вимог	Процесний	Недостатня взаємодія із замовником	Рідкі зустрічі, нечіткі канали комунікації	Неправильна реалізація вимог, переробка функціоналу	0,40	35	Високий	Регулярні демо, протоколи зустрічей, уточнення специфікацій	Бізнес-аналітик	Активний
R4	Усі етапи	Організаційний	Недостатня кількість кваліфікованих співробітників	Плинність кадрів, паралельні проєкти	Зниження продуктивності, падіння якості коду	0,30	33,3	Високий	Наставництво, навчання, резервування ключових фахівців	Технічний директор	Активний
R5	Реалізація, супровід	Процесний	Зміна вимог до підзадач у процесі реалізації	Нестабільні бізнес-пріоритети	Позапланова переробка, зростання технічного боргу	0,60	20	Високий	Введення процедури зміни вимог, backlog-менеджмент	Менеджер проєкту	Активний
R6	Інтеграція	Технічний	Несумісність із зовнішніми системами	Неповні специфікації API, відсутність тестового стенду	Затримка інтеграції, додаткові витрати	0,25	18	Середній	Раннє інтеграційне тестування, контрактне тестування	Системний архітектор	Планується
R7	Інфраструктура, DevOps	Технічний	Збої CI/CD конвеєра та середовищ розгортання	Нестабільна конфігурація, відсутність резервування	Затримки поставки релізів, простої команди	0,20	15	Середній	Стандартизація пайплайнів, резервні агенти, моніторинг	DevOps-інженер	Планується
R8	Інформаційна безпека	Зовнішній/технічний	Уразливості безпеки в модулі автентифікації	Відсутність аудитів безпеки, використання застарілих бібліотек	Компрометація даних, штрафи	0,15	40	Середній–високий	Періодичні сканування, оновлення бібліотек, code review	Провідний розробник	Моніторинг
R9	Супровід	Процесний	Недостатня якість експлуатаційної документації	Документація пишеться «у кінці», брак часу	Ускладнене впровадження, залежність від окремих осіб	0,35	10	Середній	Документація впродовж розробки, використання шаблонів	Технічний письменник	Активний
R10	Управління змінами	Організаційний	Неконтрольоване накопичення технічного боргу	Відкладання рефакторингу, відсутність обмежень	Ускладнене впровадження нових фіч, збільшення дефектів	0,40					

ДОДАТОК Б

Ролі користувачів та їх повноваження в системі

Роль користувача	Опис ролі	Основні повноваження
Менеджер проєкту	Відповідає за планування та виконання проєкту	Створення проєктів, затвердження ризиків і заходів, перегляд зведених звітів
Аналітик з ризиків	Проводить ідентифікацію і аналіз ризиків	Ведення реєстру ризиків, розрахунок параметрів, запуск оптимізації
Технічний лід	Керує технічною частиною проєкту	Оцінювання технічних ризиків, призначення виконавців, контроль заходів
DevOps-інженер	Забезпечує безперервну інтеграцію та поставку	Ведення ризиків інфраструктури, перегляд метрик розгортання
Представник керівництва	Приймає стратегічні рішення	Перегляд зведених звітів, інтегральних метрик та історії змін

ДОДАТОК В

Генерація діаграм в Python

```
import matplotlib.pyplot as plt
import numpy as np

# Налаштування стилю
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams['font.family'] = 'sans-serif'

# ------------------------------------------------------------
# Рисунок 3.1 – Зміна ймовірностей
# ------------------------------------------------------------
risks = ['R1', 'R2', 'R3', 'R4', 'R5']
probs_before = [0.30, 0.50, 0.40, 0.30, 0.60]
probs_after = [0.15, 0.10, 0.15, 0.05, 0.20]

x = np.arange(len(risks))
width = 0.35

fig1, ax1 = plt.subplots(figsize=(10, 6))
rects1 = ax1.bar(x - width/2, probs_before, width, label='До
впровадження', color='#d65f5f')
rects2 = ax1.bar(x + width/2, probs_after, width, label='Після
впровадження', color='#5fba7d')

ax1.set_ylabel('Ймовірність (P)')
ax1.set_title('Рисунок 3.1 – Зміна ймовірностей реалізації
ключових ризиків\nдо та після впровадження стратегії')
ax1.set_xticks(x)
ax1.set_xticklabels(risks)
ax1.legend()
ax1.set_ylim(0, 0.7)

# Додавання значень над стовпчиками
ax1.bar_label(rects1, padding=3)
ax1.bar_label(rects2, padding=3)

fig1.tight_layout()

# ------------------------------------------------------------
# Рисунок 3.2 – Розподіл ресурсів
# ------------------------------------------------------------
resources = ['x1', 'x2', 'x3', 'x4', 'x5']
# x1=Аналіз, x2=Тестування, x3=Комунікація, x4=Контроль якості,
x5=Резерв
```

```
    res_base = [15, 15, 15, 155, 300]
    res_opt = [15, 15, 60, 30, 380]

    fig2, ax2 = plt.subplots(figsize=(10, 6))
    rects_res1 = ax2.bar(x - width/2, res_base, width,
label='Базовий сценарій', color='#4c72b0')
    rects_res2 = ax2.bar(x + width/2, res_opt, width,
label='Оптимізований сценарій', color='#dd8452')

    ax2.set_ylabel('Витрати ресурсу (людино-години)')
    ax2.set_title('Рисунок 3.2 – Порівняння розподілу ресурсів\nдо
та після оптимізації стратегії')
    ax2.set_xticks(x)
    ax2.set_xticklabels(resources)
    ax2.legend()

    ax2.bar_label(rects_res1, padding=3)
    ax2.bar_label(rects_res2, padding=3)

    fig2.tight_layout()

    # ------------------------------------------------------------
    # Рисунок 3.3 – Інтегральні метрики
    # ------------------------------------------------------------
    # Оскільки масштаби дуже різні (50 vs 5000), робимо два сабплоти
    labels_metrics = ['E(L)\nОчікувані втрати', 'F(x)\nЦільова
функція']
    e_l_vals = [52, 15.92]
    f_x_vals = [5600, 1188.75]

    fig3, (ax3a, ax3b) = plt.subplots(1, 2, figsize=(12, 6))
    fig3.suptitle('Рисунок 3.3 – Зміна інтегральних метрик ризику
та вартості ризикових втрат')

    # Графік для E(L)
    x_metric = np.arange(1)
    width_m = 0.5
    ax3a.bar(x_metric - width_m/2, [e_l_vals[0]], width_m,
label='До впровадження', color='#d65f5f')
    ax3a.bar(x_metric + width_m/2, [e_l_vals[1]], width_m,
label='Після впровадження', color='#5fba7d')
    ax3a.set_title('Очікувані прямі втрати E(L)')
    ax3a.set_ylabel('Умовні одиниці')
    ax3a.set_xticks([])
    ax3a.legend(loc='upper right')
    # Підписи значень
```

```
    for i, v in enumerate([e_l_vals[0]]):
        ax3a.text(i - width_m/2, v + 1, str(v), ha='center',
fontweight='bold')
    for i, v in enumerate([e_l_vals[1]]):
        ax3a.text(i + width_m/2, v + 1, str(v), ha='center',
fontweight='bold')

    # Графік для F(x)
    ax3b.bar(x_metric - width_m/2, [f_x_vals[0]], width_m,
label='До впровадження', color='#4c72b0')
    ax3b.bar(x_metric + width_m/2, [f_x_vals[1]], width_m,
label='Після впровадження', color='#dd8452')
    ax3b.set_title('Цільова функція вартості F(x)')
    ax3b.set_ylabel('Умовні одиниці')
    ax3b.set_xticks([])
    ax3b.legend(loc='upper right')
    # Підписи значень
    for i, v in enumerate([f_x_vals[0]]):
        ax3b.text(i - width_m/2, v + 100, str(v), ha='center',
fontweight='bold')
    for i, v in enumerate([f_x_vals[1]]):
        ax3b.text(i + width_m/2, v + 100, str(v), ha='center',
fontweight='bold')

    fig3.tight_layout()

    # Показати всі графіки
    plt.show()
```