НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

ПУТІЙ ІЛЛЯ ДМИТРОВИЧ

УДК 004:005.8

ДИСЕРТАЦІЯ

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ УПРАВЛІННЯ СКЛАДНИМИ ІТ-ПРОЄКТАМИ В УМОВАХ НЕВИЗНАЧЕНОСТІ

122 – Комп’ютерні науки
12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів
мають посилання на відповідне джерело
_____ Путій Ілля Дмитрович

Науковий керівник
Тесленко Павло Олександрович,
кандидат технічних наук, доцент

Одеса – 2026

АНОТАЦІЯ

Путій І.Д. **Інформаційна технологія управління складними ІТ-проєктами в умовах невизначеності.** — Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 Комп'ютерні науки (12 Інформаційні технології). Національний університет «Одеська політехніка», Міністерство освіти і науки України, Одеса, 2026.

У дисертаційному дослідженні вирішена актуальна науково-прикладна задача з розробки та вдосконаленні існуючих моделей, методів та інформаційних засобів управління складними ІТ-проєктами в умовах невизначеності. Мета дослідження — у підвищенні ефективності управління складними ІТ-проєктами шляхом розробки та вдосконалення моделей, методів та інформаційних засобів управління складними ІТ-проєктами в умовах невизначеності через її зниження.

Розробка задач дослідження потребувало проведення аналізу складності ІТ-проєктів, розгляду сутності невизначеності в ІТ-проєктах, виявлення особливостей наукових досліджень та гіпотез в складних ІТ-проєктах. Проаналізовано існуючи моделі та методи управління складними ІТ-проєктами, підходи до управління гіпотезами в ІТ-проєктах. Було виявлено їх недоліки та обмеження та обґрунтована доцільність вдосконалення задля зменшення невизначеності та забезпечення успішного завершення таких проєктів.

В роботі виокремлено тип складності ІТ-проєктів, який пов'язаний з технологічною складністю, або складністю розробки. Складність, що пов'язана з інноваційністю та науковими розробками в роботі не розглядалася. Показано, що забезпечення успішного завершення таких ІТ-проєктів пов'язано зі зменшенням невизначеності через формування та перевірку гіпотез.

Задля виключення людського фактору, що має знизити помилки та проєктні ризики, в роботі запропоновано застосування моделей і методів інтелектуального аналізу даних та систем штучного інтелекту.

В дослідженні розроблено інформаційну модель складного ІТ-проєкту з науково-дослідницькою та дослідно-конструкторською розробкою, яка враховує доменну інтерпретацію ІТ-проєкту, через врахування предметної області, її сутності, процесів та взаємозв'язків між компонентами.

Запропонована концептуальна модель управління складним ІТ-проєктом з науково-дослідницькою та дослідно-конструкторською розробкою (НДДКР) в умовах невизначеності враховує заходи для роботи з невизначеністю в проєкті через її ідентифікацію, зниження рівня невизначеності та рефакторингом управління проєктом.

Модель управління гіпотезами в складних ІТ-проєктах з НДДКР в умовах невизначеності забезпечує прийняття проєктного рішення в залежності від експериментальних даних після опрацювання ідентифікованих гіпотез. Застосування систем ШІ та ІАД забезпечують автоматизацію процесу перевірки гіпотез та скорочують час ухвалення рішень та мінімізують помилки через людський фактор.

Представлені моделі лягли в основу розробки методу управління складними ІТ-проєктами, який уключає у себе: двоетапний метод формування гіпотез в складних ІТ-проєктах та рефакторинг-метод управління складними ІТ-проєктами. Останній було розроблено, спираючись на процедуру рефакторингу програмного коду, з метою його покращення. Рефакторинг-метод управління подано як процедуру покращення управління складним ІТ-проєктом через зниження невизначеності, що дозволило приймати обґрунтовані проєктні рішення. В роботі показано валідацію адекватності, достовірності та стійкості методу рефакторингу, яка довела його наукову обґрунтованість та практичну придатність. Практичним результатом дисертаційного дослідження стала розроблена інформаційна система управління складними ІТ-проєктами як цілісного інструменту підтримки

адаптивного управління в умовах невизначеності, технологічної новизни та багатофакторної складності. Інформаційна система формує інтелектуальний контур управління через аналітичну інтерпретацію, перевірку управлінських гіпотез та рефакторинг плану управління задля прийняття проектних рішень. На відміну від традиційних систем управління ІТ-проєктами, реалізована система працює з невизначеністю як з окремим об'єктом управлінського впливу.

В роботі запропоновано та розроблено структуру синтетичного датасету для перевірки працездатності системи в умовах керовано-реалістичного сценарію. Синтетичний характер датасету не знижує його наукової цінності, оскільки він відтворює не випадковий набір подій, а керовану історію розвитку складного ІТ-проєкту з подіями, аномаліями, прихованими невизначеностями, управлінськими реакціями та наслідками цих реакцій.

Реалізація системи управління складними ІТ-проєктами підтвердила практичну здійсненність всіх розроблених у дисертації моделей та методів. Система забезпечує поєднання збору даних, інтелектуального аналізу, сценарного моделювання, підтримки рішень та адаптації плану управління в єдиній ІС. Результати дисертаційного дослідження впроваджено в діяльність компаній ІТ-сфери України.

Ключові слова: проєкт, управління проєктами, складний ІТ-проєкт, управління складними ІТ-проєктами, ітеративна технологія, Agile, методологія управління ІТ-проєктами, невизначеність, Spikes, Proof of Concept, ІТ-проєкт, науково-дослідні та дослідно-констукторські розробки, інформаційна модель, концептуальна модель, доменна модель, доменний підхід, гіпотеза, управління гіпотезами, інтелектуальний аналіз даних, штучний інтелект, предиктивна аналітика, машинне навчання , глибоке навчання, рефакторинг управління, ризики, управління ризиками ІТ-проєктів, ризики складності, управління проєктами, інформаційна система, проект, технічна верифікація, ефективність проєктів, управління ризиками, інформаційна технологія, команда проєкту, управління командою.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Путій І. Д., Тесленко П. О. (2024). Аналіз сучасних підходів до управління складними ІТ-проєктами. *Управління розвитком складних систем,* 59, 81–89. DOI: 10.32347/2412-9933.2024.59.81-88. (Реєстр наукових фахових видань України, категорія «Б»).

2. Путій, І. Д., & Тесленко, П. О. (2025). Концептуальна модель складного ІТ-проєкту. Таврійський науковий вісник. Серія: Технічні науки, (2), 148 -154. DOI: https://doi.org/10.32782/tnv-tech.2025.2.16. (Реєстр наукових фахових видань України, категорія «Б»).

3. Путій І. Д., Тесленко П. О. (2025). «Канвас» методу формування гіпотез для складного ІТ-проєкту. Управління розвитком складних систем, 64, 120–127. DOI: 10.32347/2412-9933.2025.64. (Реєстр наукових фахових видань України, категорія «Б»).

4. I. Putii, P. Teslenko. (2025) Complex IT project management information system. Proceedings of Odessa Polytechnic University, ISSN 2223-3814 (online) Issue 2(72), 119 – 127. DOI: 10.15276/opu.2.72.2025.13. (Реєстр наукових фахових видань України, категорія «Б»).

Опубліковані праці апробаційного характеру:

5. Путій І.Д., Бондар О.О., Скопін Р.П. (2023). Особливості проєктів створення радіотехнічних продуктів. Збірка праць МНПК «Інтелектуальні інформаційні системи в управлінні проєктами та програмами», Коблево, 12–15 вересня 2023 р. — Харків: ХНУРЕ, 2023, С. 167-169.

6. Путій І., Косенко А., Бондар О. (2023). Управління проєктами створення радіотехнічних пристроїв як складними системами // Project, Program, Portfolio Management. P3M-2023: Тези доп. VIII Міжнар. науково-практ. конф., м. Одеса, 1–2 груд. 2023 р. Одеса, 2023. – С. 201-204

7. Путій І.Д., Бондар О.А., Мартиненко С.С. (2024). Особливості управління дослідженнями в складних ІТ проєктах. Кібербезпека, робототехніка, автоматика, комп'ютерна інженерія. КРАКІН'24. Матеріали 59-ї науково-технічної конференції студентів та молодих вчених ІШІР (13-15 травня 2024 р.). – Одеса: Національний університет «Одеська політехніка», С. 5 – 7.

8. Путій І.Д., Бондар О.А. (2024). Складність ІТ-проєктів з науково-дослідною та дослідно-конструкторською розробкою. Міжнародна науково-практична конференція «Інформаційні системи в управлінні проєктами та програмами», Коблево, ХНУРЕ, 197 –200.

9. Путій І.Д, Бобровський О. (2024). Інформаційна модель складного ІТ-проєкту з дослідницькою складовою. Project, Program, Portfolio Management. Р3М-2024: Тези доповідей IX Міжнародної науково-практичної конференції:[у 2т.]. Відп. за вип. П.О. Тесленко. Том 1. Одеса: ІШІР, 2024. С. 316-319. URL : https://www.doi.org/10.5281/zenodo.151651746.

10. Ілля Путій. (2025). Розробка інтелектуальної системи управління складними ІТ-проєктами. Штучний інтелект в інклюзивному розвитку. AIID-2025: Тези доповідей І Міжнародної науково-практичної конференції. Одеса. ІШІР, 123 – 127. URL: https://www.doi.org/10.5281/zenodo.19053150

11. Ілля Путій, Павло Тесленко. (2025). Рефакторинг-метод управління складними ІТ-проєктами. Project, Program, Portfolio Management. Р3М-2025: Тези доповідей X Міжнародної науково-практичної конференції. Одеса. ІШІР, 190 – 193. https://www.doi.org/10.5281/zenodo.18428325

ABSTRACT

Putii I.D. **Information technology for managing complex IT projects under conditions of uncertainty. –** Qualification scientific work in the form of a manuscript.

Dissertation for the degree of Doctor of Philosophy in the specialty 122 Computer Science (12 Information Technology). National University "Odesa Polytechnic", Ministry of Education and Science of Ukraine, Odesa, 2026.

The dissertation study solves a relevant scientific and applied problem of developing and improving existing models, methods and information tools for managing complex IT projects under conditions of uncertainty. The purpose of the study is to increase the efficiency of managing complex IT projects by developing and improving models, methods and information tools for managing complex IT projects under conditions of uncertainty through its reduction.

The development of the research tasks required an analysis of the complexity of IT projects, consideration of the essence of uncertainty in IT projects, identification of the features of scientific research and hypotheses in complex IT projects. Existing models and methods for managing complex IT projects, approaches to managing hypotheses in IT projects were analyzed. Their shortcomings and limitations were identified and the feasibility of improvement was substantiated in order to reduce uncertainty and ensure the successful completion of such projects.

The work identifies the type of complexity of IT projects, which is associated with technological complexity or development complexity. The complexity associated with innovation and scientific developments was not considered in the work. It is shown that ensuring the successful completion of such IT projects is associated with reducing uncertainty through the formation and testing of hypotheses.

In order to exclude the human factor, which should reduce errors and project risks, the work proposes the use of models and methods of intelligent data analysis and artificial intelligence systems.

The study developed an information model of a complex IT project with research and development, which takes into account the domain interpretation of the IT project, taking into account the subject area, its essence, processes and relationships between components.

The proposed conceptual model for managing a complex IT project with research and development (R&D) under uncertainty takes into account measures for working with uncertainty in the project through its identification, uncertainty reduction, and project management refactoring.

The hypothesis management model in complex IT projects with R&D under uncertainty ensures that a project decision is made depending on experimental data after processing the identified hypotheses. The use of AI and IAD systems automate the process of hypothesis testing and reduce decision-making time and minimize errors due to the human factor.

The presented models formed the basis for the development of a method for managing complex IT projects, which includes: a two-stage method for forming hypotheses in complex IT projects and a refactoring method for managing complex IT projects. The latter was developed based on the procedure for refactoring software code, with the aim of improving it. The refactoring management method is presented as a procedure for improving the management of a complex IT project by reducing uncertainty, which allowed making informed design decisions. The paper shows the validation of the adequacy, reliability and stability of the refactoring method, which proved its scientific validity and practical applicability. The practical result of the dissertation research was the development of an information system for managing complex IT projects as an integral tool for supporting adaptive management in conditions of uncertainty, technological novelty and multifactorial complexity. The information system forms an intellectual management circuit through analytical interpretation, verification of management hypotheses and refactoring of the

management plan for making project decisions. Unlike traditional IT project management systems, the implemented system works with uncertainty as a separate object of management influence. The paper proposes and develops the structure of a synthetic dataset for testing the system's performance in a controlled-realistic scenario. The synthetic nature of the dataset does not reduce its scientific value, since it does not reproduce a random set of events, but a controlled history of the development of a complex IT project with events, anomalies, hidden uncertainties, management reactions and the consequences of these reactions.

The implementation of the complex IT project management system confirmed the practical feasibility of all the models and methods developed in the dissertation. The system provides a combination of data collection, intelligent analysis, scenario modeling, decision support and adaptation of the management plan in a single IS. The results of the dissertation research have been implemented in the activities of IT companies in Ukraine.

Keywords: project, project management, complex IT project, complex IT project management, iterative technology, Agile, IT project management methodology, uncertainty, Spikes, Proof of Concept, IT project, research and development, information model, conceptual model, domain model, domain approach, hypothesis, hypothesis management, data mining, artificial intelligence, predictive analytics, machine learning, deep learning, management refactoring, risks, IT project risk management, complexity risks, project management, information system, project, technical verification, project effectiveness, risk management, information technology, project team, team management.

LIST OF PUBLICATIONS OF THE APPLICANT ON THE TOPIC OF THE DISSERTATION

Scientific works in which the main scientific results of the dissertation are published:

1. Putii I. D., Teslenko P. O. (2024). Analysis of modern approaches to the management of complex IT projects. Management of complex systems development, 59, 81–89. DOI: 10.32347/2412-9933.2024.59.81-88. (Register of scientific professional publications of Ukraine, category "B").

2. Putii, I. D., & Teslenko, P. O. (2025). Conceptual model of a complex IT project. Tavria Scientific Bulletin. Series: Technical Sciences, (2), 148 -154. DOI: https://doi.org/10.32782/tnv-tech.2025.2.16. (Register of Scientific Professional Publications of Ukraine, category "B").

3. Putii I. D., Teslenko P. O. (2025). "Canvas" of the method of forming hypotheses for a complex IT project. Management of the development of complex systems, 64, 120–127. DOI: 10.32347/2412-9933.2025.64. (Register of Scientific Professional Publications of Ukraine, category "B").

4. I. Putii, P. Teslenko. (2025) Complex IT project management information system. Proceedings of Odessa Polytechnic University, ISSN 2223-3814 (online) Issue 2(72), 119 – 127. DOI: 10.15276/opu.2.72.2025.13. (Register of Scientific Professional Publications of Ukraine, category "B").

Published works of an approbation nature:

5. Putii I.D., Bondar O.O., Skopin R.P. (2023). Peculiarities of projects for the creation of radio engineering products. Collection of works of the MNPK "Intelligent information systems in project and program management", Koblevo, September 12–15, 2023 - Kharkiv: KhNURE, 2023, pp. 167-169.

6. Putii I., Kosenko A., Bondar O. (2023). Management of projects for the creation of radio engineering devices as complex systems // Project, Program,

Portfolio Management. P3M-2023: Abstracts of the VIII International Scientific and Practical Conference, Odessa, December 1–2. 2023 Odesa, 2023. – P. 201-204

7. Putii I.D., Bondar O.A., Martynenko S.S. (2024). Peculiarities of research management in complex IT projects. Cybersecurity, robotics, automation, computer engineering. KRAKIN'24. Materials of the 59th scientific and technical conference of students and young scientists of ISHI (May 13-15, 2024). – Odesa: National University "Odesa Polytechnic", P. 5 – 7.

8. Putii I.D., Bondar O.A. (2024). Complexity of IT projects with research and development. International scientific and practical conference "Information systems in project and program management", Koblevo, KhNURE, 197 –200.

9. Putii I.D., Bobrovsky O. (2024). Information model of a complex IT project with a research component. Project, Program, Portfolio Management. P3M-2024: Abstracts of the 9th International Scientific and Practical Conference: [in 2 vols.]. Correspondence to issue P.O. Teslenko. Volume 1. Odesa: ISHI, 2024. P. 316-319. URL: https://www.doi.org/10.5281/zenodo.151651746.

10. Ilya Putii. (2025). Development of an intelligent management system for complex IT projects. Artificial intelligence in inclusive development. AIID-2025: Abstracts of the 1st International Scientific and Practical Conference. Odesa. ISHR, 123 – 127. URL: https://www.doi.org/10.5281/zenodo.19053150

11. Ilya Putii, Pavlo Teslenko. (2025). Refactoring method of managing complex IT projects. Project, Program, Portfolio Management. P3M-2025: Abstracts of the X International Scientific and Practical Conference. Odesa. ISHR, 190 – 193. https://www.doi.org/10.5281/zenodo.18428325

ЗМІСТ

ВСТУП

Актуальність теми. Управління ІТ-проєктами в умовах невизначеності супроводжується зростанням складності сучасних програмних систем та динамічністю середовища їх розробки. Сучасні ІТ-проєкти характеризуються високим рівнем невизначеності, що проявляється у змінності вимог, недостатній визначеності бізнес-цілей, швидкій еволюції технологій та залежності від зовнішніх факторів. Додатковими проблемами є складність архітектурних рішень, інтеграція з іншими системами, обмеженість ресурсів, а також наявність ризиків, пов’язаних із людським фактором та недостатнім рівнем компетенцій команди.

Існуючі методи управління ІТ-проєктами, зокрема класичні та гнучкі підходи, не завжди забезпечують ефективність прийняття проєктних рішень в умовах невизначеності. Вони часто орієнтовані на стабільні умови або передбачають поступове уточнення вимог без достатнього рівня формалізації процесу роботи з невизначеністю. У багатьох випадках рішення приймаються інтуїтивно або на основі обмеженої інформації, що призводить до помилок, перевитрат ресурсів і зриву термінів реалізації проєкту.

Для зниження рівня невизначеності необхідно впроваджувати системні підходи, які дозволяють ідентифікувати, формалізувати та перевіряти припущення, що виникають у процесі розробки. Важливим є створення інформаційних технологій, які забезпечують підтримку прийняття рішень на основі аналізу даних, використання моделей оцінювання складності ризиків та прогнозування результатів. Це дозволяє переходити від інтуїтивного до обґрунтованого управління.

Серед ефективних методів зниження невизначеності в управлінні складними ІТ-проєктами можна виділити підхід, заснований на формуванні та перевірці гіпотез. Він передбачає формулювання припущень щодо можливих рішень і їх експериментальну перевірку за допомогою прототипів, дослідних розробок та тестування. Важливу роль відіграє проведення наукових

досліджень у межах ІТ-проєкту, що дозволяє отримувати нові знання та обґрунтовувати вибір технологій і архітектурних рішень. Крім того, необхідним є безперервне навчання команди, особливо у випадках, коли вона не має достатніх компетенцій у певних аспектах розробки програмного продукту, що сприяє підвищенню якості прийнятих рішень.

Стан невизначеності в ІТ-проєктах не є випадковим явищем, а виступає їх природною характеристикою. Це пов’язано з інноваційністю, складністю та високою динамікою розвитку галузі. Подібно до бізнесу, де високий рівень ризику часто супроводжується можливістю отримання значного прибутку, у складних ІТ-проєктах саме робота в умовах невизначеності створює передумови для досягнення конкурентних переваг.

Таким чином, існує протиріччя між запитом ІТ-ринку на впровадження систем управління складними ІТ-проєктами, що забезпечать їх гарантоване успішне завершення, з однієї сторони, та недосконалістю існуючих моделей та методів прийняття проєктних рішень в умовах невизначеності, з іншої сторони. Тому актуальними є дослідження, що направлені на усунення цього протиріччя та присвячені вирішенню важливої науково-практичної задачі розроблення ефективних моделей та методів, інформаційних технологій управління складними ІТ-проєктами в умовах невизначеності.

Зв’язок роботи з науковими програмами, планами, темами.

Робота виконана у відповідності до тематики наукових планів кафедри штучного інтелекту та аналізу даних Національного університету «Одеська політехніка» в рамках науково-дослідної роботи № НДР 236-177 (№ держреєстрації 0123U102593) «Інтелектуальні технології управління ІТ-проєктами в умовах невизначеності», 2023 – 2027 р.р., де автор був виконавцем окремих розділів.

Мета і завдання дослідження.

Метою дисертаційної роботи є підвищення ефективності управління складними ІТ-проєктами через розробку та вдосконалення моделей, методів

та інформаційних засобів зменшення невизначеності проєктного середовища задля забезпечення можливості прийняття обґрунтованих проєктних рішень.

Досягнення поставленої мети потребує вирішення наступних **завдань** дисертаційного дослідження:

− виконати аналіз наукових досліджень та практичних результатів щодо особливостей складних ІТ-проєктів, проблеми управління ними та окреслити межі наукового пошуку; дослідити моделі, методи та інструменти управління складними ІТ-проєктами в умовах невизначеності;

− розробити інформаційну модель складного ІТ-проєкту з науково-дослідницькою та дослідно-конструкторською складовою враховуючи доменну структуру предметної галузі;

− розробити концептуальну модель управління складним ІТ-проєктом в умовах невизначеності з науково-дослідницькою та дослідно-конструкторською складовою з адоптацією процедури рефакторингу до управління проєктами, що зробить управління проєктом прогнозованим, гнучким і науково обґрунтованим;

− розробити модель управління гіпотезами складними ІТ-проєктами з науково-дослідницькою та дослідно-конструкторською складовою, інтегруючи системи штучного інтелекту та інтелектуального аналізу даних у процес управління розробкою, що дозволить знизити високу невизначеність проєктного середовища;

− застосувати процедуру канванса до розробки методу формування гіпотез для складного ІТ-проєкту в умовах невизначеності;

− вдосконалити рефакторинг-метод до управління складними ІТ-проєктами в умовах невизначеності;

− розробити метод управління складними ІТ-проєктами в умовах невизначеності;

− розробити інформаційну систему управління складними ІТ-проєктами в умовах невизначеності.

Об'єктом дослідження є процеси управління складними ІТ-проєктами в умовах невизначеності.

Предметом дослідження є моделі, методи та інформаційні засоби управління складними ІТ-проєктами в умовах невизначеності.

Методи дослідження.

Методологічну основу дослідження склали принципи гіпотетико-дедуктивного методу, системного аналізу та формалізації, що дозволило описати структуру проєкту у вигляді інформаційних і концептуальних моделей. Для обробки даних використано методи Data Mining, машинного навчання та NLP, а також байєсівські мережі і теорію прийняття рішень для обґрунтування управлінських дій. Реалізація в ІС базувалася на UML-моделюванні, принципах модульності та сценарному аналізі, що забезпечило інтеграцію аналітики, перевірки гіпотез і рефакторингу в єдиний адаптивний контур управління.

Наукова новизна одержаних результатів.

Вперше:

– розроблено інформаційну модель складного ІТ-проєкту з науково-дослідницькою та дослідно-конструкторською складовою, засновану на доменному підході до аналізу складових ІТ-проєктів, що дозволяє виокремити значущі сутності проєкту в умовах складності та невизначеності;

– розроблено метод управління складними ІТ-проєктами, який заснований на інтеграції наукових принципів гіпотетико-дедуктивного підходу, інструментів аналізу даних і концепції адаптивного рефакторингу управлінських процесів, що дозволяє зменшити невизначеність під час реалізації ІТ-проєктів та створити самонавчальну систему управління, здатну постійно вдосконалюватися на основі емпіричних результатів і зворотного зв'язку.

Удосконалено:

– рефакторинг-метод управління складними ІТ-проєктами, що переносить концепцію рефакторингу з програмної інженерії до сфери

управління проєктами, який передбачає поетапне вдосконалення структури управління ІТ-проєктами без порушення інваріантів проєкту та базується на поєднанні ML, графових імовірнісних моделей та LLM-алгоритмів.

Одержали подальший розвиток:

– двоетпний метод формування гіпотез у формі канвансу («Hypothesis Canvas 2.0 для ІТ-проєктів»), який адаптовано до специфіки технічних і архітектурних рішень у складних ІТ-проєктах та на відміну від класичних бізнесових моделей полягає у спрямованості на технічну життєздатність і критичність гіпотез, що забезпечує допомогу у прийнятті рішень і створює основу для подальшої автоматизації управлінських дій.

Практичне значення одержаних результатів

Практичне значення результатів дисертаційного дослідження полягає у підвищенні ефективності управління сучасними ІТ-проєктами за рахунок використання розробленої інформаційної технології управління складними ІТ-проєктами в умовах невизначеності. Інформаційна технологія містить у собі розроблені в дисертації моделі, методи та інформаційну систему, які у комплексі призначені для зниження невизначеності проєктного середовища, що дає можливість команді приймати обґрунтовані проєктні рішення.

Розроблена в дисертаційній роботі інформаційна система управління складними ІТ-проєктами в умовах невизначеності була задіяна у бізнесових процесах ТОВ «СФ Інвест» (Додаток А), а також знайшли відображення у навчальному процесі та науково-дослідницький діяльності Національного університету «Одеська політехніка» в НДР № 236-177 «Інтелектуальні технології управління ІТ-проєктами в умовах невизначеності» (номер державної реєстрації 0123U102593) та НДР № 237-177 «Програмні системи машинного навчання та їх застосування в галузі інтелектуального аналізу даних», (номер держреєстрації 01230102594) (Додаток Б).

Результати дисертаційної роботи можуть бути використані в діяльності проєктно-орієнтованих ІТ-компаній, профілем яких є розробка та

впровадження складних ІТ-проєктів, які потребують наукових та/або експериментальних досліджень.

Особистий внесок здобувача. Усі наукові результати, що виносяться на захист, одержані здобувачем самостійно. У публікаціях, виконаних у співавторстві, особисто здобувачу належать:

В [1] –аналіз сучасних підходів до управління складними ІТ-проєктами, визначено джерела та фактори складності, обґрунтовано застосування Spikes як підхід до подолання складності;

В [2] – розробка концептуальної моделіь управління складним ІТ-проєктом з науково-дослідною складовою з процедурою рефакторингу управління яка дозволяє корегувати структуру проєкту без зміни його мети;

В [3] – запропонована структура Canvas для систематизації процесу генерації технічних гіпотез, з метою структурування невизначеності, у формі методу формування гіпотез для складного ІТ-проєкту;

В [4] – обґрунтування, розробка та тестування працездатності інформаційної системи управління складними ІТ-проєктами в умовах невизначеності;

У [5], [6] – аналіз особливостей проєктів створення радіотехнічних продуктів як складних проєктів;

У [7] – аналіз особливостей управління дослідженнями в складних ІТ проєктах;

У [8] – обґрунтування складності ІТ-проєктів з науково-дослідною та дослідно-конструкторською розробкою;

В [9] – розробка інформаційної моделі складного ІТ-проєкту з дослідницькою складовою;

В [10] – розробка інтелектуальної системи управління складними ІТ-проєктами;

В [11] – рефакторинг-метод управління складними ІТ-проєктами.

Апробація результатів дисертації.

Результати досліджень дисертаційної роботи доповідалися та обговорювалися на таких національних та міжнародних науково-практичних конференціях:

Міжнародна науково-практична конференція «Інтелектуальні інформаційні системи в управлінні проєктами та програмами» (ІСУПП-2023, ІСУПП-2024), (Коблево – Харків, Україна: ХНУРЕ, 2023, 2024); Міжнародна науково-практична конференція «Project, Program, Portfolio Management» (P3M-2023, P3M-2024, P3M-2025), (м. Одеса, Україна: Національний університет «Одеська політехніка», 2023, 2024, 2025); Всеукраїнська науково-практична конференція «Кібербезпека, робототехніка, автоматика, комп'ютерна інженерія» (КРАКІН'24), (м. Одеса, Україна: Національний університет «Одеська політехніка», 2024); Міжнародна науково-практична конференція «Штучний інтелект в інклюзивному розвитку» (AIID-2025), (м. Одеса, Україна: Національний університет «Одеська політехніка», 2025).

Публікації.

За результатами дослідження опубліковано 11 наукових праць: 4 наукових статті у фахових виданнях України категорії «Б»; 7 тез доповідей на наукових конференціях (Додаток В).

Структура дисертації.

Дисертація включає вступ, 4 розділи, висновки та 5 додатків. Загальний обсяг дисертації – 198 сторінки, з них основного тексту – 152 сторінок. Дисертація містить 15 рисунків, 19 таблиць в основному тексті та посилання на 143 використаних джерел.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Сучасні моделі, методи та стандарти управління складними IT-проєктами

1.1.1 Особливості та класифікація складних IT-проєктів

В роботі досліджуються IT-проєкти, які зазвичай плануються та реалізуються за гнучкими техніками, але навіть «стандартні» гнучкі техніки розробки не змозі повною мірою нівелювати невизначеність, що має місце виключно через особливості таких складних проєктів. Тому у першому розділі розглянуто складні IT-проєкти, їх особливості та проблеми, їх місце у категорії IT-проєктів, та існуючи техніки і інструменти управління такими проєктами. За складністю проєкти класифікують на прості, організаційно-складні, технічно-складні, ресурсно-складні, комплексно-складні [1].

Загальне визначення складних проєктів звертає увагу на присутність в них нетривіальних технічних, організаційних або ресурсних завдань, рішення яких передбачає спеціальні підходи та збільшені витрати на їх вирішення [1]. Зазвичай до складних проєктів відносіть такі, що потребують проведення досліджень перед тим як будуть прийняті проєктні рішення для проведення подальших робіт. У технічних галузях такі проєкти мають назву науково-дослідницьких та дослідно-конструкторських, так звані проєкти НДДК [2]. Складність IT-проєктів, як і будь-якої складної системи визначається значною кількістю складових, а також складністю їх поведінки та зв’язків між ними [3]. До першої категорії можна віднести великий обсяг функціоналу або багатьох модулів розробки через велику кількість задач та управління ними. Сюди ж можна віднести велику кількість зацікавлених сторін з різними потребами та очікуваннями. Крім того, може бути складним управління конфліктами між стейкхолдерами та забезпечення згоди між різними сторонами.

Складність поведінки та складність зв’язків між складовими IT-проєктів може проявлятися через технічну складність реалізації проєктів, яка вимагатиме використання складних технологій, значної інтеграції з іншими

системами або розробки складної архітектури. Низький рівень стандартизації, або корпоративної культури ІТ-компанії, через застосування нових технологій чи вирішення нових проблем/задач, можуть стати джерелом складності через відсутність досвіду та належних практик, щодо їх подолання.

Зовнішні фактори можуть додавати складності проєктам через: залежність від доступних ресурсів; від співпраці з іншими організаціями; залежність від політичних чинників. Це проявляється через невпевненість, невизначеність або не контрольованість над цими факторами. Вимоги, що постійно змінюються або розширюються з часом, можуть бути складними через необхідність адаптації до змін та управління ризиками. Всі зазначені фактори можуть впливати на складність ІТ-проєктів, або безпосередньо формувати її. Для подолання зазначеної складності та забезпечення ефективного управління ІТ-проєктами розробники застосовують гнучкі технології управління, Agile. Але навіть в них існують ситуації коли команда проєкту немала досвіду у розробці продукту, або команда не володіє технологією для реалізації такого продукту, коли заздалегідь невідомий результат застосування технології.

Виходом з такої ситуації є процедура науково-дослідницької та дослідно конструкторської розробки (НДДР). Але ІТ-проєкти і в цьому питанні мають власні особливості. У більшості своїй ІТ-проєкти створюють програмний продукт який є віртуальним. Він не містить фізичних складових, таких як друковані плати, елементи конструкції, елементи живлення та інше, на кшталт радіотехнічним продуктам [4]. Це означає, що у повному обсязі наукові дослідження з метою реалізації їх результатів у конструкторські розробки для ІТ-проєктів виконувати непотрібно. Але, якусь частину досліджень та експериментів в ІТ галузі виконувати все ж таки необхідно, але вони, як і самі ІТ-проєкти будуть знаходитеся у віртуальній площині. Необхідність проведення досліджень, пошуку, перевірки гіпотез для практичних рішень в ІТ-проєктах не є даниною моді. Окремо виділимо групу факторів складності ІТ-проєктів, що пов’язані зі складністю поведінки системи управління ІТ-

проєктами через відсутність досвіду технологій реалізації практичних рішень та належних практик, що потребує попереднього пошуку або досліджень перед розробкою власне продукту ІТ-проєкту.

В практиці управління такими складними ІТ-проєктами застосовують інструмент, який англійською називають Spikes, або шипи [5]. Це компонент циклу гнучкої розробки для вирішення технічних та функціональних проблем. За допомогою Spikes команда ліквідовує будь-яку невизначеність в User story.

Використання Spikes у гнучкій розробці програмного забезпечення, що має затверджену абревіатуру ASD (Agile software development) може дозволити проєктно-орієнтованої організації планувати весь цикл розробки, забезпечуючи дотримання вимог проєкту. В [6] подано результат застосування Spikes у проєкті IoT підчас інтеграції датчиків та інших пристроїв до системи управління енергоспоживанням. Spikes використовувалися для визначення оптимальних технічних рішень та мінімізації ризиків, шляхом тестування різних підходів до збору та обробки даних. Прикладом складності технічного завдання може бути проєкт інтеграції нового API від стороннього постачальника, коли команда не мала досвіду роботи з цією технологією. У проєкті було проведений Spikes для отримання необхідних навичок, що зменшило технічні ризики [7].

Іншим прикладом технічної складності будемо вважати типову задачу, за якою необхідно оцінити можливість інтеграції платформи, що розробляється до існуючої інфраструктури замовника, з метою уникнути потенційних проблем, а значить і зниження рівня ризикованості [6].

Ще однією причиною досліджень в ІТ-проєктах є необхідність застосування інноваційних підходів. Це супроводжується використанням нових технологій або методологій, які потребують попереднього тестування їх ефективності для конкретного проєкту. В теорії управління проєктами до складних ІТ-проєктів різні науковці додають проєкти за різними класифікаційними ознаками. Так в [8] автори автори вважають складними «проєкти, які потребують складних адаптивних систем для управління,

враховують швидку зміну вимог та є технологічно насиченими». В [9] дана характеристика «звичайного» складного проєкту. «.. Це такий проєкт, що включає велику кількість взаємозалежних елементів, має високий рівень невизначеності, багатозадачність та потребує адаптивного управління».

В [10] автор проаналізував «наукові згадування» про складні проєкти різної галузевої прив'язки, та сформулював таке інтегроване бачення: «Складні проєкти характеризуються інтеграцією багатьох технологій, великою кількістю зацікавлених сторін, високим рівнем ризиків і непередбачуваністю в процесі реалізації»

Основні складові, що часто зустрічаються у визначеннях складного проєкту наступні: 1) Багаторівнева структура – велика кількість підсистем та взаємозв'язків; 2) Висока невизначеність – швидка зміна вимог, потреба в гнучкому управлінні; 3) Велика кількість зацікавлених сторін – складна комунікація та координація; 4) Високі ризики – технічні, організаційні, фінансові ризики; 5) Інноваційність та технологічна складність – використання новітніх технологій; 6) Взаємозалежність компонентів – зміни в одній частині впливають на інші; 7) Глобальна або міжфункціональна, розподілена команда – залучення спеціалістів з різних областей; 8) Необхідність адаптивного управління – Agile, Lean, Hybrid методології; 9) Системний підхід – керування складною взаємодією між компонентами.

За аналізом доступних літературних джерел можна стверджувати, що Spikes є своєрідним маркером складних ІТ-проєктів [11]. Сформулюємо визначення Spikes.

Spikes — це різновид спринта (або часового інтервалу ІТ-проєкту) за час якого команда не створює програмний продукт (або будь-яку його частину). Результатом виконання Spikes може бути: перевірка гіпотези; вибір варіанту реалізації проєкту / продукту; оцінка ефективності рішення; вибір методу реалізації та інше. Можна сказати, що за час проведення Spikes команда проєкту набуває знання, досвід, кваліфікацію, практичні навички при вирішенні складних для неї завдань. За час Spikes команда проєкту вибудовує

виключно себе. Якщо під час розробки складного, або як зазначено у [12], проєкту з високим ступенем інновацій, паралельно вибудовується два проєкти: один це створення програмного продукту, то другий — це створення креативної команди [13], то після звичайного спринта, співвідношення збільшення програмного продукту і «командного продукту», можна вважати 50%/50% (мається на увазі, що збільшуються обидва продукти, наведена кількість у 50% — умовна), то під час Spikes співвідношення буде: 0%/100%, тобто зростає тільки команда, а продукт проєкту у своєму вимірному обсязі залишається незмінним.

Оскільки інноваційність є однією з причин застосування Spikes, необхідно проаналізувати категорію «інноваційний проєкт» та її співвідношення до категорії «складний проєкт» [14].

Інноваційні ІТ проєкти — це такі проєкти, що впроваджують нові технології, методології або рішення, що значно змінює існуючі процеси або створює нові можливості. Визначення інноваційного ІТ-проєкту базується на кількох ключових характеристиках: використовують передові технології, які ще не набули широкого розповсюдження; мають на меті вирішити проблеми або задовольнити потреби, які не могли бути ефективно вирішені раніше, за відсутністю сучасних підходів або методів; сприяють значним покращенням в продуктивності, ефективності або якості послуг та продуктів.

Таким чином, інноваційні ІТ-проєкти визначаються не тільки використанням новітніх технологій, але й значним впливом на ефективність, продуктивність та якість процесів і продуктів. Але не всі інноваційні ІТ-проєкти можна вважати складними. Інноваційність та складність є різними поняттями, хоча вони можуть взаємодіяти.

Вважають, що поняття «інноваційність» є ширшим за «складність». Інноваційність означає впровадження нових ідей, технологій або методів, що змінюють існуючі процеси або створюють нові можливості. Складність, з іншого боку, стосується рівня труднощів у реалізації проєкту, включаючи технічні, організаційні та управлінські аспекти [15]. Інноваційний проєкт може

бути простим, якщо нововведення легко впроваджуються та не вимагають значних змін у існуючих системах або процесах.

Так само і не всі складні ІТ-проєкти є інноваційними. Складність може бути викликана багатьма факторами, такими як масштаб проєкту, кількістю зацікавлених сторін, інтеграцією з існуючими системами або складністю управління ресурсами. Проєкт може бути складним через технічні проблеми або високі вимоги до безпеки, але при цьому не містити жодних інноваційних елементів. Таким чином, інноваційність та складність є незалежними характеристиками ІТ-проєктів. Інноваційність може включати елементи новизни та змін, тоді як складність визначається рівнем труднощів у реалізації проєкту. для наочної інтерпретації, простір складності ІТ-проєктів подано на рисунку 1.1.

Рисунок 1.1 – Простір складності ІТ-проєктів. Джерело: розроблено автором

Простір сформований у вигляді квадратної матриці підприємство горизонталі якої відкладається рівень складності та підприємство вретикалі — рівень інноваційної. В середині подані типи ІТ-проєктів, які відповідають відповідній градації. За результатами досліджень було сформульовано визначення «складного ІТ-проєкту» яке буде використовуватися в дисертаційному дослідженні.

Визначення 1.1. ***Складний ІТ-проєкт*** — це проєкт у сфері інформаційних технологій, який характеризується високим рівнем

невизначеності, великою кількістю взаємозалежних компонентів, складною архітектурою, багаторівневою інтеграцією систем, необхідністю використання новітніх або нестандартних технологій, значною кількістю зацікавлених сторін та потенційно мінливими вимогами. Його реалізація потребує спеціальних підходів до управління, таких методологій як Agile, Lean, Spikes або процедуру наукових-досліджень, а також значних ресурсних вкладень для мінімізації ризиків через зменшення невизначеності. Складність такого проєкту може бути зумовлена як технічними аспектами (інноваційні алгоритми, нестандартні технології, інтеграція з іншими складними системами), так і організаційними факторами (розподілені команди, управління конфліктами між стейкхолдерами, зовнішні впливи та регуляторні обмеження). Усі зазначені компоненти складності формують простір невизначеності в якому неможливо, або складно прийняте вірне проєктне рішення.

Ці фактори визначають складність ІТ-проєкту та вимагають специфічних підходів до його управління. Структура складності подана на рисунку 1.2.

Таким чином, в роботі, будемо розглядати саме складні ІТ-проєкти, що має сенс у складності реалізації. Як було зазначено вище, інноваційні ІТ-проєкти також будуть розглядатися в дисертації, але як одна з складових складних, в яких складність реалізації обумовлена науковою складовою.

1.1.2 Аналіз особливостей управління складними ІТ –проектами

Гнучкі техніки, моделі та методи розробки ІТ проєктів (Agile) є важливими для успішного управління складними ІТ-проєктами, особливо коли потрібно використовувати Spikes для дослідження та зниження невизначеностей. Найбільш ефективними Agile методологіями для таких проєктів є Scrum, Kanban та Extreme Programming (XP) [16].

Рисунок 1.2 – Структура складності IT-проєктів. Джерело: розроблено автором

Scrum як розвиток ітеративної технології забезпечує структуру для розбиття проєкту на короткі, керовані ітерації або спринти, де кожна ітерація завершується конкретним результатом. Використання Spikes у Scrum дозволяє командам виділяти спеціальні ітерації для дослідження та вивчення нових технологій, оцінки ризиків або вирішення складних технічних проблем, що пов’язані з невизначеністю без загрози загальному прогресу проєкту [17]. Канбан, з його візуальним управлінням завданнями та гнучкістю, дозволяє командам реагувати на зміну або невизначеність вимог та пріоритетів у реальному часі без порушення загальної структури.

XP акцентує увагу на технічній досконалості та постійних інтеграціях, що допомагає в управлінні технічними ризиками та забезпечує високу якість коду. В XP Spikes можуть використовуватися для тестування нових підходів, інструментів або методів перед їх впровадженням у основний процес розробки. Застосування DevOps у поєднанні з Agile також є ефективним підходом для складних IT-проєктів. DevOps забезпечує безперервну

інтеграцію та розгортання, що дозволяє швидко реагувати на зміни та зменшує час від виявлення проблеми до її вирішення.

Інші Agile методології, такі як Lean та Crystal, також можуть бути ефективними в залежності від специфіки проєкту та команди. Lean підхід, з акцентом на зменшення відходів і оптимізацію процесів підвищує ефективність і продуктивність команди.

Crystal може адаптуватися до розмірів команди та проєкту, забезпечує індивідуальний підхід до кожного проєкту, що особливо корисно для складних і динамічних IT-проєктів. Зазначимо, що незалежно від вибору методології, ключовим аспектом успішного управління складними IT-проєктами є корпоративна культура та постійне вдосконалення команди. Використання Agile методологій та регулярним зворотним зв'язком забезпечує гнучкість, адаптивність та високу якість продукту проєкту. Тим не менш, для повної інтеграції Spikes до методології Scrum, Kanban та XP необхідно внести певні зміни. У Scrum варто додати спеціальні спринти, призначені виключно для Spikes, з чіткими критеріями оцінки їх результатів [18]. Це дозволить команді зосередитися на дослідженнях без тиску основних задач.

В Kanban впроваджують окремі колонки для Spikes на дошці завдань.

В Extreme Programming Spikes інтегрують у практики безперервної інтеграції та тестування, що забезпечує можливість швидкого експериментування з новими технологіями або підходами із проведення Spikes та аналізу їх результатів [19] а також регулярні ретроспективи, присвячені обговоренню результатів Spikes, їх впливу на загальний процес розробки. Порівняння технологій та їх відношення до Spikes подано в таблиці 1.1.

Загалом, у всіх методологіях важливо забезпечити чітке документування та обмін набутого досвіду, отриманими під час Spikes, щоб вся команда проєкту могла використовувати ці результати для покращення майбутніх ітерацій.

Таблиця 1.1 – Гнучкі технології розробки щодо застосування Spikes

Методологія	Основний принцип / підхід	Сильні сторони	Робота з невизначеністю	Можливість використання Spikes	Необхідні зміни для Spikes	Обмеження / слабкі сторони
Scrum	Ітеративна розробка через спринти	Чітка структура регулярний фідбек, контроль прогресу	Через короткі ітерації та ретроспективи	Частково вбудовано	Введення окремих Spike-спринтів; визначення критеріїв результату	Жорсткість спринтів може обмежувати дослідження
Kanban	Безперервний потік задач	Гнучкість, візуалізація процесу, швидка реакція на зміни	Динамічна пріоритизація задач	Вбудовано у потік	Додавання окремих колонок для Spikes; політики пріоритизації та time-boxing	Відсутність чітких ітерацій ускладнює контроль досліджень
XP	Технічна досконалість і якість коду	Висока якість, швидкий пошук проблем	Через тестування, CI/CD, часті релізи	Добре інтегрується	Включення Spikes у процеси тестування	

Джерело: розроблено автором

1.1.3 Сутність невизначеності в складних ІТ–проектах

Наступним етапом, сформуємо понятті невизначеності в межах дослідження з управління складними ІТ-проєктами. У загальновживаному значенні невизначеність — це стан, що пов’язаний з частково невідомою або неповною інформацією про події чи ситуації, що використовується у прогнозуванні, фізичних вимірюваннях, статистиці, тощо. Вона виникає через неповну видимість, стохастичність, складність або динамічність середовища.

В управлінні проєктами невизначеність порівнюють з ризиком, при тому наголошують на те, що традиційні засоби управління (планування, контроль) недостатні для її обробки, її ключові елементи — рефлексивне навчання та sensemaking, що допомагають обрати альтернативні дії у ситуації невизначеності [20]. Тоді невизначеність трактується як більш загальне явище,

яке не обмежується лише ризиками, але включає недостатність знань чи прогнозів для прихованого майбутнього.

Тобто ризик — це можлива подія з ймовірністю та наслідками, а невизначеність — це ситуація, в якій невідомі як результати, так і ймовірності цих результатів [21]. Це технічне визначення використовується для практичних задач управління проєктами: ризики можна кількісно оцінити, а невизначеність — ні. В проєктному менеджменті, в роботі [22] невизначеність визначається так: суб'єктивний брак знань про об'єкт, який ускладнює розуміння ситуації, що потребує рішення. Це означає, що невизначеність — це саме ситуація браку знань, а не конкретний ризик, її визначення побудоване навколо того, що інформації недостатньо для прийняття однозначного рішення.

В контексті складних IT проєктів майже відсуне окреме строго формалізоване визначення саме "невизначеності IT-проєкту". В роботах за напрямком Software Project Management під невизначеністю зазвичай розуміють: стан, у якому невідомі параметри майбутнього розвитку проєкту через складність вимог, новизну технологій, неоднозначність вимог, взаємозалежності, які не можуть бути виражені у ймовірностях на початку проєкту, як це подано наприклад в [23].

Розкриття сутності невизначеності, на яке будемо спиратися у дисертаційному досліджені, необхідно через те, що заходи управління складними проєктами будуть направлені на зменшення цієї невизначеності задля підвищення керованості проєктом. Тому невизначеність будемо вважати фундаментальною властивістю фізичної реальності, і вона не є наслідком неточності вимірювання, а виникає через саму структуру квантового світу, як межа знання, обумовлена структурою теорії [24]. В економіці Френк Найт також розмежував ризики та невизначеність. Ризик це відома ймовірність, а невизначеність — це стан, коли майбутнє не може бути кількісно описане, що є епістемічним визначеням. В теорії інформації Клод Шеннон [25] трактує невизначеність як ентропію інформаційного повідомлення:

$$H = -\sum p(x) \log p(x)$$

Чим більше варіантів — тим більша невизначеність. Вона вимірюється кількісно і може бути охарактеризована як математизована невизначеність як міра варіативності станів.

В теорії складних систем Іллі Пригожина, невизначеність є необхідною умовою самоорганізації [26]. Флуктуації та нестійкість створюють можливість переходу системи до нового порядку. Тобто, без невизначеності немає розвитку. Це вже еволюційна, або системна невизначеність. Результати проведеного аналізу зведено до таблиці 1.2.

Таблиця 1.2 – Сутність невизначеності за різними джерелами

Автор	**Тип невизначеності**	**Природа**	**Можливість вимірювання**	**Ключова ідея**
Гейзенберг	Квантова	Онтологічна	Так (формула)	Вбудована у реальність
SEP	Фізико-філософська	Структурна	Так	Межа опису
Найт	Економічна	Епістемічна	Ні	Невідомі ймовірності
Шеннон	Інформаційна	Статистична	Так (ентропія)	Міра варіативності
Філософський словник	Концептуальна	Системна	Частково	Брак однозначності
Пригожин	Складні системи	Еволюційна	Ні (якісно)	Джерело розвитку

Джерело: розроблено автором

Далі пов’яжемо складний ІТ-проєкт з невизначеністю. Складний ІТ-проєкт є відкритою системою, що має множинність можливих станів, які ведуть до непередбачуваних наслідків. Разом з тим, складний ІТ-проєкт має структурні межі пізнання та не може бути повністю детермінований.

Отже, невизначеність складного ІТ-проєкту :

– не зводиться до ризику;

– не є помилкою планування;

– є властивістю відкритої адаптивної системи;

– виникає через множинність альтернатив;

– створює умови для еволюції рішення.

Будемо інтерпретувати її як ширину множини можливих управлінських рішень та розрив між необхідним і наявним знанням. Тобто невизначеність — це не просто “відсутність інформації”, а стан системи, в якому через дефіцит релевантних знань неможливо однозначно визначити причинно-наслідкові зв’язки і, відповідно, неможливо обґрунтувати єдине рішення.

Визначення 1.2. ***Невизначеність складного ІТ-проєкту*** — це системний стан проєкту, зумовлений дефіцитом або неповнотою релевантних знань щодо вимог, технологій, середовища та взаємодії учасників, який призводить до неможливості обґрунтовано вибрати однозначне управлінське рішення та породжує множинність альтернатив із невідомими наслідками.

Концептуально це визначення можна розгорнути наступним чином:

1. Джерело — брак інформації (epistemic gap).
2. Прояв — нечіткість причинно-наслідкових зв’язків.
3. Наслідок — множинність управлінських альтернатив.
4. Ризик — різні варіанти матимуть різну, але невідому ефективність.
5. Мета управління — зменшення дефіциту інформації і звуження простору альтернатив до раціонально обґрунтованого вибору.

Тим не менш, невизначеність ≠ хаос. Вона означає, що:

– існує багато можливих рішень,

– але немає достатніх підстав вибрати одне як раціонально домінуюче.

Тоді управління невизначеністю — це не “боротьба з ризиками”, а процес: зменшення туману → уточнення причин → звуження альтернатив → стабілізація рішення.

Визначення 1.3. ***Управління невизначеністю складного ІТ-проєкту*** — це формалізований, ітеративний та автоматизований управлінський процес, спрямований на виявлення слабких сигналів нестабільності, їх аналітичну інтерпретацію, перетворення у конкретні проблемні зони та реалізацію цілеспрямованих втручань (дослідницьких, організаційних або технічних), які зменшують варіативність результатів проєкту, звужують множину

допустимих альтернатив рішень і підвищують передбачуваність його розвитку.

Зниження невизначеності буде досягатися за рахунок:

– централізованого автоматизованого збору проєктних даних;

– виявлення ранніх відхилень та нестабільних ділянок;

– формування та аналітичної оцінки управлінських гіпотез;

– експериментальної перевірки цих гіпотез через PoC, Spike, навчання або архітектурні зміни;

– вимірювання впливу втручань на динаміку ключових показників;

– адаптації плану проєкту відповідно до отриманих результатів.

Таким чином, управління невизначеністю — це не спроба її повного усунення, а структурований механізм переведення нестабільності з неконтрольованого стану у керований режим розвитку проєкту.

Таке управління передбачає автоматизований збір слабких сигналів із проєктного середовища [27], їх аналітичну обробку, інтерпретацію та формалізацію у вигляді проблемних зон, а також генерацію, оцінку і перевірку управлінських гіпотез через дослідницькі та експериментальні дії (PoC, Spike, навчання, архітектурні зміни). В умовах Scrum та Agile IT-проєктів, де команди є невеликими і функціонально завантаженими, управління невизначеністю має спиратися на автоматизовані інтелектуальні механізми, які забезпечують раннє виявлення слабких сигналів та підтримку прийняття рішень без додаткового навантаження на команду.

Одним із напрямків зменшення невизначеності є виявлення та аналіз «ранніх ознак» або слабких, ще неформалізованих сигналів. Цей підхід використовують у різних галузях [28], в тому числі і в управлінні IT-проєктами [29].

Концепція ранніх ознак описана в теорії sensemaking К. Вейка [30], за допомогою якої можливо інтерпретувати неоднозначні сигнали з навколишнього середовища, щоб запобігти кризам або адаптуватися до змін. Ранні ознаки — це слабкі, часто непомітні сигнали, які передують значущим

подіям. Їх своєчасне розпізнавання дає можливість суб'єктам переосмислити ситуацію та скорегувати дії до того, як загроза стане очевидною. У процесі sensemaking увага спрямовується не стільки на сам факт наявності сигналів, скільки на колективну інтерпретацію їх значення. Це передбачає взаємодію між членами організації, обмін знаннями та створення спільних уявлень про те, що відбувається. Таким чином, концепція ранніх ознак у теорії Вейка пояснює, як процес осмислення допомагає перетворювати невизначеність у зрозумілі моделі дії, забезпечуючи ефективне реагування на зміни у складних соціальних системах.

1.2. Особливості ІТ-проєктів НДДК розробки

В підрозділі 1.1 була розглянута теперішня ситуація в ІТ-проєктах з якою стикаються практики. Мова йшла про складності в розробці, вирішення яких потребує проведення експериментальних досліджень. Тобто складності тої групи ІТ-проєктів які потребували експериментальних досліджень містилися в процесах управління та розробки.

Між тим, існує ще одна група ІТ-проєктів в яких «складність» міститься вже на самому початку таких проєктів, в їх меті. Це так звані проєкти з науково-дослідною та дослідно-конструкторською розробкою (НДДК-проєкти). Для коректності подальшого викладу матеріалу слід визначитися з терміном «конструкторської розробки» в ІТ-проєктах. Відомо, що у більшої частини ІТ-проєктів, особливо в тих, які відносять до групи програмних проєктів, конструктивна розробка, та конструктивна частина — відсутні. Вона може бути лише в ІТ-проєктах, продуктом яких є апаратно-програмний комплекс [31].

НДДК — це сталий та відомий термін, який був дуже поширений у XX сторіччі серед науковців та розробників нової техніки (коли ІТ ще не було), розробці яких передували наукові дослідження і саме за результатами тих досліджень створювалося нова техніка. Тобто термін НДДК розробки

розуміють створення нового продукту (в будь-який технічній галузі) якої передують наукові дослідження та експерименти [1].

Слід зазначити, що точного, закріпленого в державних стандартах (ДСТУ) або міжнародних глосаріях (ISO/PMBOK) терміна «Складний ІТ-проєкт з НДДКР» як єдиної конструкції не існує. Проте у науковому середовищі, зокрема в працях фахівців з управління проєктами в Україні, це поняття активно використовується та розкривається через поєднання трьох складових: складність, ІТ-сфера та науково-дослідна компонента.

Термін «НДДКР» часто інтегрують в опис проєктів. В [32] подано таке визначення R&D project. Це шлях до відкриття невідомого та вирішення проблем, що не мають очевидних рішень, шляхом методичних експериментів, які проводяться компетентними фахівцями для досягнення наукового або технологічного прогресу.

В академічній літературі складність проєкту часто визначається як «складність систем».

Так в [33] складний ІТ-проєкт це екосистема, сформована з взаємопов'язаних складних підсистем, поведінку яких неможливо точно передбачити або контролювати на основі знань про кожну підсистему окремо.

Основними характеристики таких проєктів є наступні критерії: невизначеність результату, на відміну від типової розробки (software development), у проєктах з НДДКР неможливо гарантувати 100% успіх на старті; наукова новизна, проєкт спрямований на створення нових методів, а не лише на імплементацію існуючих рішень; висока складність управління потребує інтелектуальних систем підтримки прийняття рішень через динамічну зміну вимог.

Тому в дисертаційній роботі, для розкриття сенсу процесів дослідження вважаємо за доцільне використовувати термін НДДК у комбінації «ІТ-проєкти з НДДК», що означатиме, що для створення ІТ продукту, в ІТ-проєкті слід виконати наукові дослідження та провести експерименти для перевірки доцільності та ефективності застосування їх результатів.

ІТ-проєкти з НДДК можуть бути охарактеризовані як комплекс заходів, які спрямовані на створення нових інформаційних технологій або вдосконалення існуючих, шляхом проведення наукових досліджень та експериментів з розробки інноваційних рішень. Окремо зазначимо, що наукові дослідження і дослідницькі експерименти є частиною таких ІТ-проєктів, є частиною їх WBS (Work Breakdown Structure), або змісту.

Основною особливістю таких проєктів є висока ступінь невизначеності та ризиків, що пов'язана з необхідністю проведення досліджень, експериментів та тестувань нових технологічних підходів.

ІТ проєкти з НДДК відрізняються від звичайних НДДК проєктів тим, що включають в себе специфічні аспекти, пов'язані з розробкою програмного забезпечення, інформаційних систем та технологічних платформ, що потребує залучення висококваліфікованих ІТ фахівців та використання сучасних методів управління проєктами. ІТ проєкт з НДДК можна визначити як систематичний процес дослідження, розробки та впровадження нових інформаційних технологій або вдосконалення існуючих, який включає в себе науково-дослідні та дослідно-конструкторські роботи з метою створення інноваційного продукту або рішення. Від звичайного ІТ проєкту він відрізняється насамперед наявністю науково-дослідного компоненту, що може передбачати проведення фундаментальних та прикладних досліджень, а також значний обсяг експериментальної роботи.

В процесі НДДК розробок можуть з’являтися нові дані та гіпотези, що може призвести до зміни початкових цілей чи завдань проєкту. З цього виникає два завдання: перше — відслідковування та дотримання початкової мети проєкту, друге — формування та управління гіпотезами проєкту.

Особливість структури ІТ проєктів з НДДК полягає в їх складності та багатокомпонентності. Зазвичай такі проєкти включають етапи проведення наукових досліджень, розробки прототипів, тестування, вдосконалення та впровадження результатів досліджень у практичну діяльність. WBS таких проєктів відображає детальну декомпозицію робіт, необхідних для досягнення

поставлених цілей, та враховує специфіку науково-дослідних робіт, розробки програмного забезпечення та інтеграції отриманих результатів.

Управління командою в ІТ проєктах з НДДК також має свої особливості. Важливою є наявність у команді не лише програмістів та інженерів, але й науковців, дослідників та аналітиків, що володіють спеціалізованими знаннями та досвідом у відповідних галузях. Це вимагає від керівника проєкту високого рівня компетентності в управлінні міждисциплінарними командами та здатності забезпечити ефективну комунікацію між учасниками проєкту.

В [34] дослідження та розробку визначено як угоду між суб'єктами господарювання, що мають спільні інноваційні цілі та працюють в режимі гібридної організації, та розподіляють матеріальні та нематеріальні ресурси для спільної розробки дослідницького проєкту. Автори [34] відзначають складність визначення компетенції академічних груп та їх навичок. Також складність присутня при описі механізмів координації та контролю. Важливим є зобов'язання академічного партнера не співпрацювати з конкурентами під час розробки спільного дослідницького проєкту, що направлено на зменшення ризику моральної небезпеки.

ІТ проєкти з НДДК мають додаткові ризики. Вони пов'язані з високою невизначеністю результатів досліджень, можливими технічними складнощами при розробці нових технологій, необхідністю дотримання вимог щодо безпеки та конфіденційності даних, а також можливими змінами у зовнішньому середовищі, які можуть вплинути на хід проєкту.

В [35] зазначається, що науково-дослідні проєкти включають різні невизначеності, пов'язані з часом, технологіями, фінансами та знаннями, що потребує уважне дослідження управління ризиками цих проєктів, щоб визначити всі потенційні дії, які можуть мати негативний вплив на процеси або результати проєкту. Для зменшення ризиків проєктів з НДДК розробкою автори [35] пропонують застосовувати підхід інтервального інтуїтивно нечіткого АНР та аксіому нечіткої інформації. Це дозволить визначити ступінь важливості визначених факторів ризику. В роботі виявлено, що важливим

ризиком в проєктах з НДДК розробкою є »Аномальні зміни вартості», а найменш важливим є «Недоліки в статтях договору».

Автори [36] відзначають, що оскільки проєкти НДДКР мають високу невизначеність щодо графіка та якості продукції, управління ризиками таких проєктів виходить на перший план. Автори пропонують системний підхід до відслідковування невдач НДДКР та управління ризиками в процесі виконання проєктів НДДКР. В статті запропоновано використати метод FMEA (Failure Modes and Effects Analysis) в контексті аналізу режимів і ефектів відмов. Під терміном «відмова» мається на увазі «невдача» або «помилка», яка може статися в процесі розробки продукту або виконання проєкту. Це поняття походить від англійського терміну «failure». «Режим відмов» (failure modes) означає потенційні способи або причини, через які система, процес або продукт можуть не виконати свої функції належним чином. Цей метод допомагає передбачити можливі ризики заздалегідь і розробити стратегії для їх запобігання або мінімізації їхнього впливу.

Автори модифікували «режим відмов» відповідно до специфічних особливостей науково-дослідної діяльності за допомогою моделі етапних воріт, яка може ідентифікувати режими відмов на кожному етапі процесу дослідження та розробки. Також запропоновано процес визначення пріоритетності ризиків невдачі НДДКР для підтримки процесу прийняття рішень в управлінні НДДКР проєктами шляхом застосування лабораторії випробувань та оцінки прийняття рішень (DEMATEL). Метод DEMATEL (Decision Making Trial and Evaluation Laboratory) є потужним інструментом для управління ризиками проєктів. Він дозволяє досліджувати та аналізувати складні взаємозв'язки між ризиками. DEMATEL не лише ідентифікує ризики, але й визначає, які з них є активаторами інших проблем і як ці ризики впливають на проєкт загалом [37].

В роботі [38] розглянуто методи управління ризиками для науково-дослідних проєктів у Південній Австралії. За результатами дослідження автори запропонували використовувати метод RFMEA до матриці

ранжирування ризиків, а саме оцінку виявлення ризиків. Це дозволило авторам перерозподілити зусилля які не планувалося завіювати на управління ризиками на інші види діяльності в проєкті. Метод RFMEA (Risk-based Failure Modes and Effects Analysis) є розширенням традиційного підходу FMEA, спрямованим на більш глибоке управління ризиками в проєктах. Він об'єднує класичний аналіз режимів і ефектів відмов FMEA [36] з додатковим акцентом на оцінку ризиків, що дозволяє більш точно визначати пріоритетність ризиків і їхній вплив на проєкт.

У [39] автори досліджували ймовірність провалу проєкту при формуванні портфеля проєктів науково-дослідних розробок. Запропоновано рішення задачі стохастичного програмування через еквівалентну детерміновану конічну модель програмування другого порядку.

В [40] розглянута інтеграція концепцій Design Thinking, Lean Startup і Agile у науково-дослідні проєкти. Автори відзначають покращення засвоєння знань, покращення рішень і визначення короткострокових можливостей. Разом с тим було виявлено проблему «галасливої інтеграції», потребу у складних навичках керівника проєкту, і наявність складних процесів прийняття рішень.

В [41] для отримання даних щодо інструментів управління експериментами в галузі машинного навчання (ML) автори пропонують такі контрольовані експерименти: «Як ці інструменти впливають на продуктивність користувачів?», «Як користувачі їх сприймають?», «Як різні реалізації (парадигми) функцій інструментів впливають на користувачів?»

Експерименти, що зосереджені на зручності та навчанні, також важливі, включаючи учасників з різним досвідом і рівнем знань (наприклад, досвідчені проти недосвідчених, практикуючі проти дослідників, інженери програмного забезпечення проти науковців з даних). Експерименти можуть бути спрямовані на конкретні питання експериментів з ML. Було проведено експеримент, у якому студенти-розробники з досвідом у ML, але без досвіду

використання інструментів управління експериментами, виконували завдання, використовуючи інструмент з графічним інтерфейсом.

Таким чином, за результатами аналізу наукових літературних джерел можна стверджувати, що науково-дослідна розробка існує в ІТ-проєктах, такі проєкти та управління ними мають свої особливості, які потребують адоптації та розробки дієвих засобів управління такими проєктами, задля забезпечення їх успішного завершення. Життєвий цикл таких проєктів буде уключати додатково: дослідження, формування та перевірку гіпотез. у загальному вигляді життєвий цикл ІТ-проєктів з НДДК подано на рисунку 1.3.

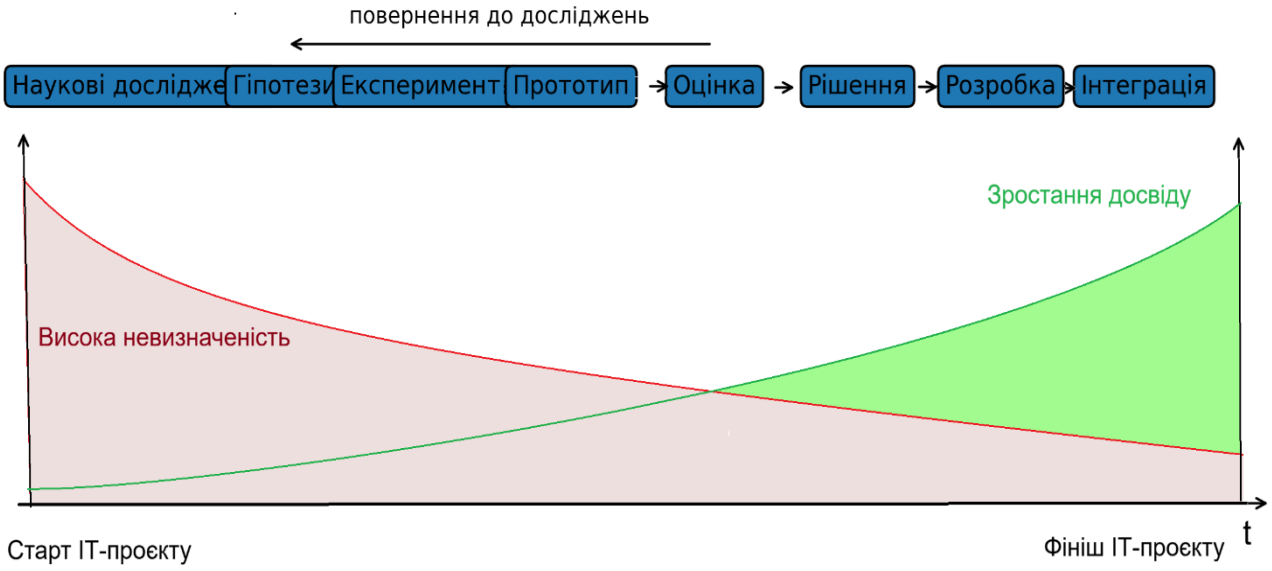


Рисунок 1.3 – Життєвий цикл складного ІТ-проєкту з НДДКР. Джерело: розроблено автором

В англомовній літературі ІТ-проєкти з елементами наукової розробки та досліджень позначаються як R&D проєкти. На думку автора, якщо порівнювати проєкти R&D та НДДК ІТ-проєкти за ступенем, або глибиною наукових досліджень в межах проєкту, то НДДК ІТ-проєкти мають біль широке значення, а R&D проєкти входитимуть в них, як частка.

1.3. Аналіз засобів управління гіпотезами, як складової складних ІТ-проєктів

Як було зазначено у попередньому підрозділі, у проєктах з НДДК розробкою підчас передпроєктних досліджень та впродовж їх життєвого циклу може з’явитися не тільки нове бачення продукту, але й нові гіпотези, щодо його існування призначення та конфігурації. Ця сутність притаманна більшою мірою проєктам з НДДК, а також іншим складним проєктам. Тому необхідно проаналізувати інструменти управління гіпотезами, як одні із засобів, що мають забезпечити їх успішне завершення.

Управління гіпотезами в ІТ-проєктах є важливою складовою процесу розробки програмного забезпечення, особливо в контексті інноваційних та науково-дослідних проєктів. Основна мета управління гіпотезами — це перевірка різних припущень, які можуть впливати на продукт, його функціональність, ефективність або прийняття користувачами.

Гіпотези в проєктах відіграють ключову роль, оскільки вони є основою для дослідження і перевірки припущень щодо функціонування системи або реалізації певної ідеї [42]. Гіпотеза — це попереднє припущення або твердження, яке пропонує можливе пояснення певного явища чи проблеми, і яке потребує емпіричної перевірки [43, 44]. В рамках проєктної діяльності гіпотези слугують для визначення основних напрямків дослідження, допомагають сфокусувати зусилля на перевірці ключових припущень, а також знижують ризики невизначеності та неправильної інтерпретації даних [45, 46].

А в [46] навпаки, автори застосовують проєктну методологію задля навчання студентів сутності та правильності формування гіпотез при розробці та управління проєктами. Існують різні типи гіпотез, що можуть бути використані у проєктах. Наприклад, нульова гіпотеза є твердженням про відсутність змін чи впливу певних факторів на результат, тоді як альтернативна гіпотеза припускає наявність такого впливу [43, 44]. Також гіпотези можуть бути кількісними, коли припускають зміни певних числових

показників, або якісними, коли йдеться про зміни властивостей чи характеристик об'єкта дослідження.

Правильне формулювання гіпотези задля досягнення успішного завершення науково-дослідного проєкту вимагає чіткості, конкретності та можливості її перевірки. Гіпотеза має бути сформульована таким чином, щоб її можна було емпірично дослідити через дані, які будуть доступні команді проєкту. Фахівці рекомендують уникати надто загальних тверджень, які важко або неможливо перевірити [47].

Гіпотеза повинна мати вигляд простого та чіткого припущення про те, як певний фактор впливає на результат або яке явище можна спостерігати за певних умов. Автор [48] пропонує підхід до оцінки інноваційних проєктів через тестування гіпотез.

В [49] автори пропонують будувати гіпотези на основі факторів успішності IT-проєктів та оцінюють їх через ієрархічну модель дослідження.

Одним із методів організації гіпотез є створення дерева гіпотез. Дерево гіпотез — це структурована схема, яка допомагає візуалізувати та систематизувати припущення на основі їх взаємозв'язків [50]. Верхівкою такого дерева зазвичай виступає основна, загальна гіпотеза.

Автори [50] розглядають новітні та популярні методи контролю помилок у тестуванні гіпотез, структурованих у вигляді дерев, на прикладі багаторазового тестування. Було досліджено частоту помилок на рівні сімейства FWER (family wise error rates) та частоту помилкових відкриттів FDR (false discovery rate) на противагу новітньому методу контроля частоти помилкового відбору FSR (false selection rate) для навігації по деревоподібних структурах у світлі цих помилок.

В [51] досліджується ефективне управління кластерами в задачах відстеження кількох гіпотез, що пов’язано зі зменшенням комбінацій проблем за допомогою дерева гіпотез. Автори [51] застосовують техніку відстеження кількох гіпотез MHT (multiple hypothesis tracking) для вирішення комбінаторної проблеми, що пов’язана з формуванням кількох гіпотез

асоціації даних. Вони зменшують весь набір елементів вхідних даних на окремі кластери, тим самим розділяють велику проблему відстеження на кілька менших проблем, які вирішуються незалежно.

До основних інструментів управління гіпотезами IT-проєктів відносять: підхід Lean Startup, підхід на основі Канвасу Гіпотез, модель Build-Measure-Learn, метод A/B тестування та Гіпотетико-дедуктивний метод. Коротко представимо результати аналізу кожного з них [42].

Підхід Lean Startup був розроблений Еріком Райсом у 2011 році, та оприлюднений в його книзі [52]. Підхід використовує гіпотези як основу для перевірки бізнес-ідей і продуктів на ранніх етапах розробки. Основний акцент автор робить на швидкому тестуванні гіпотез за допомогою мінімально життєздатного продукту (MVP) і зборі зворотного зв'язку від користувачів.

Процес управління гіпотезами за допомогою Lean Startup містить такі етапи:

– формулювання гіпотези;

– розробка MVP для перевірки гіпотези;

– збір даних і зворотного зв'язку від користувачів;

– аналіз результатів і прийняття рішення: чи продовжувати розробку, чи змінювати напрямок пошуку.

Підхід, або технологія Lean Startup отримала доволі широке застосування в управлінні IT-проєктами. Так в [53] представлена модель управління проєктами, яка об'єднує методології Agile, Lean Startup і Design Thinking. Такий підхід спрямований на створення програмного забезпечення, що орієнтоване на користувача та підтримує інновації через перевірку гіпотез і взаємодію з користувачами на ранніх етапах розробки. Таку модель управління проєктом називають Converge.

В [54] розглядається використання Lean Startup для покращення інноваційних проєктів. Автори досліджували як техніки Lean Startup забезпечують гнучке управління гіпотезами та ризиками проєктів в умовах невизначеності.

Модель Build-Measure-Learn використовується у проєктах, що застосовують Agile та Lean підходи. Вона може розглядатися, як складова підходу Lean Startup. Але може використовуватися і окремо, для управління гіпотезами в проєктах, де необхідно проводити наукові дослідження та експерименти. Модель базується на циклічному процесі управління гіпотезами [55]:

– Build: розробка прототипу або MVP на основі гіпотези;

– Measure: вимірювання результатів взаємодії користувачів з продуктом;

– Learn: аналіз даних і отримання знань про те, чи варто продовжувати гіпотезу або змінювати підхід.

У якості гіпотези команда розробників може припустити, що впровадження нової моделі машинного навчання дозволить збільшити точність рекомендацій, що в свою чергу підвищить конверсію клієнтів на 10%. В рамках моделі Build-Measure-Learn, команда починає з фази Build, тобто створення MVP, який реалізує основні функції нової моделі.

В наступному етапі Learn, команда робить висновки на основі зібраних даних. Якщо гіпотеза підтверджується, команда приймає рішення впроваджувати новий алгоритм у ширшому масштабі. Якщо ж нова система не забезпечує очікуваних результатів, гіпотеза відхиляється або коригується.

Таким чином, за допомогою моделі Build-Measure-Learn команда ітеративно вдосконалює продукт проєкту, спираючись на фактичні дані та результатах експериментів. Вона сприяє зниженню ризиків та дозволяє швидко адаптувати продукт до змінних умов ринку або нових вимог користувачів.

Гіпотетико-дедуктивний метод — це класичний науковий підхід, який також застосовується і в управлінні ІТ-проєктами для перевірки гіпотез в різних дослідницьких контекстах. Він передбачає формування системи припущень (гіпотез), на основі яких здійснюється дедуктивний процес для перевірки їхньої правдивості через спостереження або експерименти. У

контексті управління гіпотезами, цей метод дозволяє структурувати процес перевірки та вдосконалення припущень, щоб досягти найбільш точного й обґрунтованого результату. Процес починається з формулювання гіпотези, що ґрунтується на спостереженнях або знаннях про продукт чи ринок. Далі розробляються експерименти або тести, які мають підтвердити або спростувати гіпотезу. Етапи гіпотетико-дедуктивного методу:

– формулювання гіпотези: припущення, яке пояснює спостережуване явище;

– планування експерименту: створення умов, в яких можна перевірити гіпотезу;

– збір даних: проведення експерименту та отримання результатів;

– аналіз результатів: перевірка відповідності результатів очікуванням;

– підтвердження або спростування гіпотези.

Гіпотетико-дедуктивний метод краще проаналізувати в контексті розробки складного програмного забезпечення, коли проєкт вимагає проведення наукових досліджень для перевірки певних гіпотез щодо функціональності або ефективності системи, наприклад, розробки програмного забезпечення для автономного керування автомобілем. Це завдання передбачає інтеграцію кількох технологій: машинне навчання, обробку зображень та аналіз даних із відеокамер та датчиків.

Формулювання гіпотез розпочинається з попередніх досліджень або припущень. Наприклад, розробники можуть висунути гіпотезу, що застосування певної архітектури нейронної мережі, яка дозволить точніше і швидше розпізнавати об'єкти на дорозі в різних погодних умовах. На основі цієї гіпотези будується дедуктивне припущення, яке передбачає конкретний результат: система повинна розпізнавати об'єкти з точністю 95% під час дощу або снігу. Наступним кроком є розробка експериментальної системи, що дозволяє перевірити це припущення в реальних умовах. Для цього проводяться тести в симуляціях або в реальному середовищі, де система отримує дані з камер і датчиків в умовах дощу або снігу, і визначається

точність розпізнавання об'єктів. Зазначимо, що потреба в експериментах в реальних умовах, тому що сніг йде лише у певну пору року і взагалі необхідно зібрати дані для всіх типів освітлення во всі пори року.

Таким чином, гіпотетико-дедуктивний метод в управлінні складними проєктами дозволяє систематично перевіряти та вдосконалювати гіпотези на основі результатів експериментів, оцінювати, які технології будуть найбільш ефективними для виконання конкретних завдань проєкту, і надає їм можливість мінімізувати ризики невдач.

На ефективне застосування гіпотетико-дедуктивного методу в управлінні складними проєктами вказує і огляд літературних джерел. Так в [56] гіпотези створюються для тестування, а далі результати порівнюються з експериментами або спостереженнями.

В роботі [57] розглядається використання гіпотетико-дедуктивного методу в інформаційних системах та його обмеження. Автори також пропонують альтернативні підходи до управління гіпотезами в дослідженнях та висказують власну думку стосовно наповненості методу та його складових.

Метод A/B тестування використовується для перевірки гіпотез у контексті змін продукту чи користувацького інтерфейсу. Гіпотеза перевіряється шляхом порівняння двох варіантів продукту: контрольної версії (A) та експериментальної (B). A/B тестування дозволяє зібрати статистичні дані, які підтверджують або спростовують ефективність змін [58]. До процес A/B тестування відносять такі етапи:

– формулювання гіпотези про зміни, які можуть покращити продукт;

– розробка двох версій продукту (A і B);

– збір даних про взаємодію користувачів з кожною версією;

– статистичний аналіз результатів для прийняття рішення про впровадження змін.

У статті [59] представлена модель для підтримки A/B тестів на рівні класів і компонентів програмного забезпечення. Автори розробили фреймворк для впровадження експериментів у реалістичному веб-проєкті, показуючи

його ефективність у зборі інформації для ухвалення рішень. В [60] розглядається метод тестування гіпотез для оцінки стартап-проєктів, де автори використовують A/B тестування для швидкого визначення ризиків і привабливості стартапів, що дозволяє приймати обґрунтовані рішення.

Підхід на основі Канвасу Гіпотез структурує процес управління гіпотезами у вигляді канвасу, де кожна гіпотеза описується у певному форматі. Канвас (англ. *Canvas*) — це структурована візуальна схема, для систематизації, аналізу та побудови елементів бізнесу, проєкту чи ідеї. Канвас складається з різних блоків, кожен з яких відповідає за певні аспекти діяльності або концепції, що впорядковує інформацію [61].

Структура кавансу для управління гіпотезами матиме такі складові [62, 63]:

- гіпотеза це припущення, яке необхідно перевірити;
- критерії успіху це метрики які підтвердять успішність гіпотези;
- ризики це потенційні проблеми, які можуть вплинути на результат;
- дії, що необхідно зробити для перевірки гіпотези.

Така модель систематизує роботу команди над кількома гіпотезами одночасно і допомагає правильно виконати їх перевірку. Автори [64] і [65] досліджували використання підходів бізнес-канви до написання дисертації PhD. В [62] було застосовано підходи Design Thinking та бізнес-канви для метою вирішення проблем впродовж виконання дисертаційних досліджень. В результаті чого було сформовано рекомендації для визначенні тем та управління інноваціями та дослідницькими комунікаціям за допомогою Canvas бізнес-моделі.

В [65] Canvas Model Research Project застосовувалась як інструмент візуалізації процесів управління академічних дослідницьких проєктів та дисертацій. Ці інструменти надають дослідникам можливість створити стратегію, яка дозволить авторам проєкту прогнозувати потреби, невдачі, цілі проєкту та адаптувати його до умов дослідження. В [66] автори розглядають інженерію гіпотез у програмній розробці, що керується експериментами.

Розробка гіпотез — це нова дисципліна для розробки програмного забезпечення на основі експерименту, яка зосереджується на належній обробці гіпотез для керівництва експериментами та покращення взаємодії з користувачем. Для цього автори використовують модель HYPEX (Гіпотезо-експериментально керована розробка). Вона розроблена на основі проблем, виявлених під час кейс-стадій у компаніях з розробки програмного забезпечення та складається з шести практик.

1. Генерація функцій у беклозі, які можуть бути цінними для користувачів і які слід перевірити через експерименти.

2. Вибір найбільш пріоритетної функції і створення гіпотез про те, як компанія вважає, що ця функція створить цінність для користувача.

3. На третьому етапі команда виокремлює «найменшу частину функції», яка додає цінність користувачеві (так звану мінімально життєздатну функцію або MVF), і забезпечує інструментами для збору даних.

4. Під час аналізу розривів, компанія аналізує різницю між очікуваною і фактичною поведінкою користувачів. Якщо вони не збігаються, команда розробляє гіпотези щодо причин, які використовуватимуться на наступних етапах.

5. Аналіз створених гіпотез та їх пріоритизація. Якщо найімовірніша гіпотеза полягає в тому, що функція не відповідає потребам користувачів, команда повинна розглянути можливість створення альтернативної функції.

6. Якщо виявляється, що функція не задовольняє користувачів, наступна практика полягатиме в «створенні альтернативної реалізації».

Вибір методу залежить від типу проєкту, рівня та якості досліджень, наявних ресурсів та потреб у швидкості перевірки та реалізації гіпотез. З урахуванням виконаного огляду та аналізу складності і ІТ-проєктах, а також враховуючи процедуру управління гіпотезами в ІТ-проєктах, було сформоване визначення складного ІТ-проєкту, яке буде викоритсовуватися в дисертаційному дослідлекні.

Визначення 1.4. Управління складним ІТ-проєктом — це процес динамічного керівництва розробкою інформаційних систем в умовах високої невизначеності, спрямований не на сліпе виконання жорсткого плану, а на постійне зниження невизначеності та верифікацію управлінських і технічних рішень. Цей процес передбачає перехід від детермінованого (лінійного) планування до ітераційного управління на основі висування та перевірки гіпотез, через такі функції:

Ідентифікація зон невідомості: виявлення дефіциту компетенцій, ризиків сі складності та бізнес-невизначеностей на ранніх етапах.

Ініціювання експериментального циклу: організація процесів швидкого прототипування, створення тестових реалізацій (MVP) та апробації рішень для накопичення досвіду команди.

Сценарне моделювання та аналіз: розробка альтернативних траєкторій розвитку проєкту замість спроб прийняти одне безальтернативне прогнозоване рішення.

Адаптивне коригування: гнучка зміна тактики розробки, архітектурних підходів або складу технологій на основі результатів тестування гіпотез.

Трансформація знань: конвертація початкової невизначеності проєктного середовища у перевірені компетенції, стійкі підходи та прогнозовані результати для успішного завершення проєкту.

Головна зміна парадигми управління: від «контролю виконання плану» до «управління процесом дослідження, щоб знайти правильний план».

1.4. Огляд інформаційних засобів управління складними ІТ - проєктами

1.4.1 Інформаційні системи для управління дослідженнями

Проведення дисертаційних досліджень в межах 122 спеціальності «Комп’ютерні науки» потребує дослідити існуючи інформаційні систем для управління ІТ-проєктами, тому що практичне рішення дисертаційної проблеми буде спрямоване у напрямку створення інформаційної технології

для планування, організації, контролювати та координувати роботу команд, що управляють складними IT-проєктами з науковими дослідженнями та експериментальними розробками. Ось деякі з найпоширеніших:

1. Jira [67]. Використовують для управління проєктами в середовищах Agile (Scrum, Kanban), Має розвинену систему звітів і можливості інтеграції з іншими інструментами для DevOps. Має мождивості для планування, відстеження завдань, управління вимогами, релізами.

2. Trello [68] це Канбан-система для візуального управління завданнями. До особливостей цієї системами можна вднести простий інтерфейс для створення дошок із завданнями, що допомагає в управлінні проєктами завдяки візуалізації статусу кожної задачі.

3. Asana позиціонують як інструмент для управління завданнями та робочими процесами в проєкті, що уключає структурування завдань, створення підзадач, контроль строків виконання, управління календарями. Добре підходить для команд, які працюють над кількома проєктами одночасно [69].

4. Microsoft Project зазвичай позиціонують для проєктів що використовують класичну, водоспадну модель. Має потужні інструменти для планування, управління ресурсами та моніторингу проєктів, створює діаграми Ганта, контролює ресурси, може управляти портфелем проєктів та робити розподіл робочих навантажень [70]. Зазначимо, що проєкти які виконуються виключно за гнучкою технологією не рекомендують до планування в Microsoft Project.

5. Monday.com як інструмент для управління робочими процесами та проєктами має інтерактивні дошки для планування та контролю завдань, забезпечує підтримку різних робочих просторів, має зручне налаштування під різні типи проєктів [71], до яких можна віднести : Scrum або Kanban для Agile-розробки; проєкти з налаштування або оновлення серверів, мереж, систем безпеки; відстеження етапів впровадження нової інфраструктури чи переносу даних в інфраструктурних IT-проєктах; управління процесами CI/CD

(безперервна інтеграція та впровадження) в проєктах DevOps та автоматизації; управління проєктами з перенесення даних між системами або платформами; підтримку та технічне обслуговування IT-систем; в інноваційних та науково-дослідницьких IT-проєктах, відстеження гіпотез, експериментів, прототипування нових технологій, планування дослідницьких етапів і тестування нових рішень та ін.

6. ClickUp це інтегроване рішення для управління завданнями, документообігом та командною роботою в IT-проєктах. Воно підтримує кілька видів представлення даних (Канбан, діаграми Ганта, списки), забезпечує інтеграцію з іншими інструментами [72].

7. Redmine — це потужна платформа для управління складними IT-проєктами завдяки гнучкості, можливості кастомізації та підтримці різних плагінів [73]. Redmine підтримує ітераційні підходи, як то Scrum, Kanban на всіх етапах розробки. Redmine є відкритим кодом і випущений згідно з умовами GNU General Public License v2 (GPL).

8. Basecamp призначений для управління командами та робочими процесами в IT-проєктах [74]. Може створювати завдання, виконувати контроль строків, проводити обговорення та забезпечує спільне використання файлів.

9. Wrike призначений для управління великими командами і проєктами [75]. Окремо може управляти завданнями, створювати діаграму Ганта, звіти, розподіляти ресурси та забезпечує інтеграцію з іншими корпоративними системами.

10. GitLab це інструмент для розробки програмного забезпечення, що пов'язане з DevOps [76]. Вбудовані функції забезпечують безперервну інтеграцію та розгортання (CI/CD), GitLab ідеально підходить для впровадження великих та складних проєктів, де важливим є швидкі ітерації та автоматизація.

11. Zoho Projects добре підходить для управління IT-проєктами, зі створення вебсайтів, розробки програмного забезпечення та підтримки клієнтів, де важливо координувати виконання завдань різних команд [77].

12. Smartsheet це простота електронних таблиць із функціональністю інструментів для управління проєктами. Smartsheet дозволяє легко структурувати завдання, контролювати терміни виконання та розподіляти робочі ресурси [78]. Вбудовані діаграмам Ганта дозволяють відстежувати прогрес проєкту. Інтеграція з сервісами Google Drive та Microsoft Office забезпечує співпрацю команди в режимі реального часу та спрощує комунікацію.

Ці системи широко використовуються в IT-компаніях для координації команд, оптимізації робочих процесів та покращення продуктивності. Кожна з аналізованих систем має свої переваги та недоліки, залежно від специфіки проєкту та вимог команди, які було зведено в таблиці 1.3.

Таблиця 1.3 – Програмні засоби з управління IT-проєктами

Система	**Гнучкість у кастомізації під IT-проєкти**	**Інтеграція з інструмента-ми DevOps**	**Можливості колаборації команд**	**Підходить для Agile (гнучкі технології)**	**Підходить для водоспадної моделі**
Jira	Висока	Так	Високі	Так, забезпечує Agile-дошки та звіти	Так, але потребує кастомізації
Trello	Середня	Ні	Середні	Так, проста Канбан-дошка	Ні, не підходить
Asana	Висока	Так	Високі	Так, є підзадачі та проєкти	Ні, краще для гнучких
Microsoft Project	Низька	Ні	Низькі	Ні, орієнтована на традиційне управління	Так, спеціалізується на водоспадній моделі
Monday.com	Висока	Так	Високі	Так, підтримує гнучкі процеси	Так, але більше фокусується на гнучкості

Продовження табл. 1.3

Система	**Гнучкість у кастомізації під IT-проєкти**	**Інтеграція з інструмента-ми DevOps**	**Можливості колаборації команд**	**Підходить для Agile (гнучкі технології)**	**Підходить для водоспадної моделі**
ClickUp	Висока	Так	Високі	Так, гнучка адаптація	Так, адаптується для будь-якої методології
Redmine	Висока	Так	Середні	Так, можна налаштувати плагінами	Так, орієнтована на класичне управління
Basecamp	Середня	Ні	Середні	Ні, орієнтована на простіші проєкти	Так, але не оптимальна
Wrike	Висока	Так	Високі	Так, є Agile функціонал	Так, підходить
GitLab	Висока	Так	Високі	Так, інтеграція з DevOps	Так, підтримує ітерації
Zoho Projects	Середня	Ні	Середні	Обмежена підтримка, підходить, але з певними обмеженнями	Для водоспадної моделі — підходить краще
Smartsheet	Середня	Ні	Середні	Так, є Канбан і можливості планування	Так, підтримує традиційне управління

Джерело: розроблено автором

Основними факторами, що впливають на вибір, є підтримка колаборації, управління ресурсами, можливість документування досліджень та ефективне відстеження прогресу кожного етапу. Нижче представлено порівняльний аналіз програмних засобів, які найбільш підходять для таких завдань, враховуючи специфіку проєктів, що потребують експериментальних підтверджень і гіпотез.

Jira є одним із найпопулярніших інструментів для управління IT-проєктами, особливо завдяки своїй гнучкості та інтеграції з підходами Agile (Scrum, Kanban). Це робить її відмінним вибором для наукових IT-проєктів, де необхідно проводити різні етапи досліджень і експериментів. У Jira можна

використовувати епіки для представлення великих дослідницьких напрямків або експериментальних гіпотез, а окремі історії користувача або завдання — для опису кожного експерименту чи кроку [79]. Також система надає можливість детальної звітності та інтеграції з DevOps для контролю версій, що дозволяє ефективно керувати всіма етапами досліджень [80].

Microsoft Project дозволяє управляти ресурсами, завданнями та часовими у багатозадачних наукових проєктах. Через діаграму Ганта можна відстежувати виконання складних завдань, що важливо для великих дослідницьких проєктів з тривалими експериментальними циклами. Для опису досліджень у Microsoft Project можна організувати кожне дослідження як підпроєкт, де кожен експеримент оформлюється як завдання з датами початку та завершення.

Asana пропонує простий інтерфейс для управління проєктами, що робить його ефективним для команд, які працюють паралельно над різними етапами проєктів. Основною перевагою Asana є можливість створення підпроєктів та підзадач для кожного окремого дослідження або етапу [81]. Для опису досліджень можна створити проєкт для кожного дослідницького напрямку та структурувати його за допомогою завдань і підзавдань.

Wrike забезпечить взаємодію з різними командами або відділами у дослідницьких проєктах. Інтерфейс Wrike підтримує кілька режимів візуалізації (діаграми Ганта, Канбан), що дозволяє ефективно керувати складними дослідницькими проєктами. Для опису досліджень і експериментів у Wrike можна створювати етапи проєкту для кожної фази дослідження, де експеримент може бути окремим завданням, з детальним описом змінних.

Smartsheet поєднує простоту таблиць з можливостями планування і управління, що робить його корисним для дослідницьких команд. Цей інструмент забезпечує зручні функції відстеження і дозволяє структурувати складні проєкти, додаючи елементи для управління завданнями. Для наукових досліджень у Smartsheet можна використовувати таблиці для детального опису кожного етапу експериментів, включаючи змінні та метрики. Система також

має функції для спільної роботи, що робить її ефективною для команд з великою кількістю учасників.

ClickUp пропонує широкий спектр можливостей для управління проєктами, включаючи підтримку різних форматів відображення даних (списки, діаграми Ганта, Канбан) [82]. Це робить його гнучким інструментом для дослідницьких команд. У ClickUp можна використовувати списки завдань для кожного дослідження, де кожне завдання представляє окремий експеримент.

Redmine, як відкритий інструмент, є гнучким рішенням для управління науковими проєктами, особливо в академічних установах або наукових лабораторіях. Ця система дозволяє налаштувати підхід під конкретні вимоги наукового проєкту [78, 83]. Для опису досліджень у Redmine можна створювати проєкти для кожного дослідницького напряму та використовувати підпроєкти для представлення різних експериментів. Наприклад CosmoBots.eu має плагін для Redmine [82], який дозволяє управляти вимогами, включаючи автоматичну та миттєву ієрархію та діаграми залежностей, імпорт/експорт з електронних таблиць, повну взаємодію з інші інструменти, що використовують REST API.

Таким чином, для управління складними, науковими IT-проєктами, найбільш ефективними інструментами будуть Jira, Microsoft Project, Wrike і Redmine [84]. Вони забезпечують гнучкість, інтеграцію з іншими системами, можливість контролю ресурсів і управління експериментами, що є важливим для успішного завершення проєктів. Результати аналізу прграмних засобів, що розглянуто у цьому пункті зведено у таблицю 1.4.

Для складних наукових проєктів, що включають дослідження**,** Jira, Microsoft Project та Wrike забезпечують найбільш потужний функціонал. Вибір залежить від того, наскільки проєкт інтегрований з IT-середовищем, наскільки важлива співпраця між віддаленими командами, і які ресурси потрібні для управління.

Таблиця 1.4 – Аналіз програмних засобів для управління IT-проєктами з елементами досліджень та експериментів

Система	Планування та управління дослідженнями	Гнучкість адаптації під IT-дослідження	Можливості колаборації	Інтеграція з інструментами DevOps	Можливість документування результатів
Jira	Високий рівень структурованості для експериментів через епіки та завдання	Висока, багато плагінів для адаптації	Високі, є можливості для командної роботи і інтеграція з іншими інструментами	Так, повна підтримка DevOps інструментів	Так, підтримка звітів і документів
Microsoft Project	Потужні можливості планування, але складне налаштування для наукових проєктів	Низька, більше для традиційного управління	Низькі, не підходить для спільних наукових експериментів	Ні, орієнтована на управління традиційними проєктами	Так, зручне документування, але громіздка система
Asana	Гнучкість у завданнях, але більше підходить для невеликих проєктів	Середня, але підходить для малих досліджень	Середні, підходить для команд середнього розміру	Ні, обмежена інтеграція з DevOps	Так, зберігання документів і файлів
Wrike	Гнучке управління етапами проєкту і дослідженнями	Висока, добре налаштовується	Високі, багато можливостей для командної роботи	Так, інтеграція з DevOps для IT-досліджень	Так, інтеграція з документами та звітами
ClickUp	Гнучке управління, адаптується під будь-який проєкт	Висока, інтуїтивне налаштування	Високі, розширені функції колаборації	Так, повна інтеграція	Так, створення і зберігання документації
Redmine	Можливе кастомне налаштування для досліджень	Висока, але потребує технічної кастомізації	Середні, налаштовується під командну роботу	Так, інтегрується з Git та іншими DevOps інструментами	Так, підтримка документування через плагіни

Джерело: розроблено автором

1.4.2 ІС для управління гіпотезами

За результатами аналізу предмету дисертаційного дослідження процедуру управління гіпотезами уключено до меж наукового пошуку. Тому необхідно провести аналіз інформаційних систем та інструментів, спеціально адаптованих для управління гіпотезами в проєктах. Аналіз існуючих інформаційних систем для управління гіпотезами в ІТ проєктах є важливим задля розробки практичних рішен.

Різноманітність програмних засобів на ринку дозволяє обрати, що найкраще відповідає потребам як традиційних ІТ-проєктів, так і проєктів, що включають повний цикл науково-дослідної роботи та експериментального підтвердження гіпотез. Airfocus [85] пропонує простий та інтуїтивно зрозумілий спосіб управління гіпотезами, зокрема, завдяки інструментам для пріоритизації завдань і тестування. Важливою перевагою цього рішення є його інтеграція з іншими популярними інструментами управління завданнями, такими як Jira, Trello та Asana. Craft.io [86] надає можливість організації та відслідковування гіпотез у процесі розробки продукту, що робить його корисним для команд, що залучають користувацькі історії та експерименти до процесу розробки. Jira [67] з плагінами для управління гіпотезами підтримує Agile-методології. Використання плагінів, таких як Jira Product Discovery або Experiment Board, дозволяє налаштувати Jira для потреб управління гіпотезами. Experiment Engine [87] є спеціалізованим інструментом для управління гіпотезами у маркетингових проєктах, зокрема через підтримку A/B-тестування та інтеграцію з аналітичними платформами. Проте обмеженість у застосуванні для інших типів проєктів, зокрема ІТ-розробок, робить Experiment Engine менш придатним для комплексного управління гіпотезами в ІТ-сфері. Notion [88] пропонує гнучкість у створенні власних структур для управління гіпотезами, завдяки використанню баз даних та шаблонів. Відсутність автоматизованих функцій для тестування або інтеграції з інструментами управління завданнями є перешкодою для складних та інтегрованих рішень. Trello [68] з адаптацією для управління гіпотезами є

простим у використанні інструментом з інтуїтивно зрозумілим інтерфейсом. Проте обмеженість функціоналу без використання додаткових плагінів або автоматизації робить його менш придатним для великих проєктів із складними експериментальними циклами.

Confluence [89], як додатковий інструмент до Jira, дозволяє створювати централізовану базу знань для команди, що є особливо корисним для документування гіпотез, їх опису та результатів тестування. Інтеграція з Jira забезпечує повний цикл управління гіпотезами, проте це рішення може бути надто складним для команд, які не мають великого досвіду роботи з такими інструментами. Результати аналізу програмних засобів що мають функціонал для управління гіпотезами в IT-проєктах, подані у таблиці 1.5.

Таблиця 1.5 – Аналіз програмних засобів для управління гіпотезами в IT-проєктах

Система	**Можливості для управління гіпотезами**	**Інтеграція з іншими інструментами**	**Гнучкість кастомізації під потреби проєкту**	**Можливості колаборації**	**Документування гіпотез і результатів**
Airfocus	Потужні інструменти для пріоритизації та тестування гіпотез	Так, інтеграція з Jira, Trello та іншими	Висока, можливість налаштування під специфічні вимоги	потужні інструменти для командної роботи	Так, можливість створення звітів та документування тестів
Craft.io	Зручна інтеграція гіпотез із користувацькими історіями	Так, інтеграція з інструментами управління продуктом	Висока, зручна кастомізація під кожен проєкт	Високі можливості для спільного управління гіпотезами	Так, зручне документування гіпотез і тестувань
Jira (з плагінами)	Широкий вибір плагінів для гнучкого управління гіпотезами	Так, інтеграція з DevOps та багатьма іншими інструментами	Висока, можливість додавання плагінів і налаштування процесів	Високі, зручна робота для команд різного розміру	Так, підтримка документів та звітів через плагіни
Experiment Engine	Інструменти для створення, тестування та аналізу гіпотез	Так, підтримка аналітичних платформ та A/B тестування	Середня, фокус на маркетингові та цифрові проєкти	Середні, більше орієнтовані на маркетингові команди	Так, можливість створення звітів та аналітики

Продовження таблиці 1.5

Система	Можливості для управління гіпотезами	Інтеграція з іншими інструментами	Гнучкість кастомізації під потреби проєкту	Можливості колаборації	Документування гіпотез і результатів
Notion	Гнучка база даних для документування та управління гіпотезами	Так, інтеграція з багатьма інструментами через API	Висока, можна налаштувати будь-які робочі процеси	Високі, спільне редагування та управління	Так, гнучке зберігання та організація документації
Trello (адаптована)	Проста адаптація для візуалізації та управління гіпотезами	Так, обмежена інтеграція з іншими інструментами через Power-Ups	Середня, кастомізація через розширення та додатки	Середні, більше для малих команд	Обмежене, можливе через додаткові плагіни
Confluence (спільно з	Глибока інтеграція з Jira для управління гіпотезами та документами	Так, повна інтеграція з Jira та іншими продуктами Atlassian	Висока, налаштування шаблонів та інтеграцій з Jira	Високі, інтеграція з Jira для командної роботи	Так, потужна система документування через Confluence

Джерело: розроблено автором

1.5. Постановка задачі дослідження

Аналіз сучасних ІТ-проєктів, за літературними джерелами та власним досвідом здобувача, показав наявність окремої групи, яка характеризується присутністю досліджень та експериментів продовж життєвого циклу ІТ-проєкту. Ці складові обумовили необхідність перевірки прийнятих проєктних рішень задля забезпечення їх успішного завершення. Це обумовило актуальність дисертаційного дослідження та необхідність вирішення наступних задач:

1) Розробити інформаційну модель складного ІТ-проєкту з НДДКР.
2) Розробити концептуальну модель управління складними ІТ-проєктами в умовах невизначеності.
3) Розробити модель управління гіпотезами в складних ІТ-проєктах.
4) Розробити метод управління складними ІТ-проєктами в умовах невизначеності.
5) Розробити метод формування гіпотез складних в ІТ-проєктах

6) Розробити рефакторинг-метод управління складними ІТ-проєктами

7) Розробити ІС управління складними ІТ-проєктами, яка поєднає в собі усі зазначені методи та моделі

Висновки до розділу 1

В першому розділі проведено аналіз та виконано класифікацію складних ІТ-проєктів. Визначено їх характеристики, особливості поведінки та фактори, що формують або додають складності. Сформульовано визначення складного ІТ-проєкту, яке віднесено наукової новизни.

1. Проведено аналіз особливостей управління складними ІТ-проєктами. Показано, що для проєктів такого типу найбільш ефективним буде застосування гнучких технік розробки. Тим не менш, для ІТ-проєктів, що матимуть наукові дослідження у своєму життєвому циклі, або експерименти для підтвердження висунутих в результаті дослідження гіпотез, доцільним може бути застосування водоспадної моделі, саме для етапу дослідження або експерименту.

2. Окремо розглянуто ІТ-проєкти, що матимуть елементи науково-дослідних та дослідно-конструкторських розробок. Показана їх сутність та межі застосування зазначених термінів, у тому числі невизначеності та складності.

3. Аналіз літературних джерел показав, що під час проведення наукових досліджень, висуваються гіпотези, які мають бути доведені або спростовані під час життєвого циклу проєкту через проведення досліджень та експериментів. Тому в першому розділі також було розглянуто та проаналізовано гіпотези, їх ознаки та сутність, розглянуто моделі та методи управління гіпотезами в ІТ-проєктах.

4. Проведений огляд сучасних програмних засобів з управління ІТ-проєктами, що матимуть у себе функціонал для ефективного проведення наукових досліджень та експериментальних розробок. Окремо розглянуто та

проаналізовано програмні засоби для управління гіпотезами при розробці ІТ-проєктів.

5. Результати досліджень першого розділу опубліковані в роботах [2, 3, 4, 31, 42].

Список використаних джерел у розділі 1

1. Чорна М.В. (2003). Проєктний аналіз. Харків, Консум, 228 с.

2. І. Путій, А. Косенко, О. Бондар. Управління проєктами створення радіотехнічних пристроїв як складними системами // Project, Program, Portfolio Management. P3M-2023: Тези доп. VIII Міжнар. науково-практ. конф., м. Одеса, 1–2 груд. 2023 р. Одеса, 2023. – С. 201-204

3. Путій І.Д., Бондар О.А., Мартиненко С.С. (2024). Особливості управління дослідженнями в складних ІТ проєктах. Кібербезпека, робототехніка, автоматика, комп'ютерна інженерія. КРАКІН'24. Матеріали 59-ї науково-технічної конференції студентів та молодих вчених ІШІР (13-15 травня 2024 р.). – Одеса: Національний університет «Одеська політехніка», С. 5 – 7.

4. Путій І.Д., Бондар О.О., Скопін Р.П. Особливості проєктів створення радіотехнічних продуктів. Збірка праць МНПК «Інтелектуальні інформаційні системи в управлінні проєктами та програмами», Коблево, 12–15 вересня 2023 р. — Харків: ХНУРЕ, 2023, С. 167-169.

5. Jeff Thull. (2003). Mastering the complex sale : how to compete and win when the stakes are high! Published by John Wiley & Sons, Inc., Hoboken, New Jersey. 220 p. ISBN 0-471-43151-6

6. Spikes. SAFe Studio. URL : https:// www.scaledagile framework.com/spikes/

7. Dwayne Campbell. (2023). How to Master Spikes for Agile Product Development: A Comprehensive Guide. URL : https://thinkproductgroup.com/how-to-master-spikes-for-agile-product-development-a-comprehensive-guide/

8.Liubava Chernova, Iryna Zhuravel, Serhii Chernov, Lyudmila Chernova, Nataliia Kunanets. (2024). Management of the IT project as a complex system. Proceedings of the 5th International Workshop IT Project Management (ITPM 2024), May 22, 2024, Bratislava, Slovak Republic, 114 – 125. URL : CEUR-WS.org/Vol-3709/paper10.pdf

9. Remington, K Pollack, JB. (2008). Complex Projects: What are they and how can we manage them more effectively? Australian Institute of Project Management. Conference Proceeding of the 2008 AIPM Project Management Conference, 2008, pp. 1 – 10. URL : http://hdl.handle.net/10453/17630

10. Frederico Steffen Neto. (2015). Knowledge evolution on complex projects: A scientific production analysis. Engineering, Business. URL : https://www.semanticscholar.org/paper/KNOWLEDGE-EVOLUTION-ON-COMPLEX-PROJECTS%3A-A-ANALYSIS-Neto/280334da670932d6553465d203483ab9da2985c7

11. David Usifo. (2024). Using Spikes to Manage Risks and Uncertainty in Scrum and Agile. URL : https://deeprojectmanager.com/spikes-in-scrum/

12. І.О. Близнюкова П.О. Тесленко, О.Б. Данченко. Інструменти управління ІТ-проєктами з інноваціями. Збірка тез VIII Міжнародної НТК «Інформатика. Культура. Технології» ІКТ-2021. — Одеса : ІКС, 2021. — С. 84-86.

13. Близнюкова І.О. Метод формування креативної команди ІТ-проєкту. Вісник НТУ «ХПІ». Серія: Стратегічне управління, управління портфелями, програмами та проєктами. Зб. наук. праць. / Нац. техн. ун-т «ХПІ», Харків, 2023. № 1(7). – С. 12–19.

14. Крамський, С. О. (2017). Застосування комплексної оцінки ризик-орієнтованих засобів в управлінні інноваційними ІТ-проєктами. Управління розвитком складних систем. 29. С. 71 – 77.

15. Жмаєва, Ю. В., & Чала, Л. Е. (2018). Гібридний метод оцінювання складності ІТ проєктів. Біоніка інтелекту, 2(91), 99-107.

16. Robert C. Martin. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Published by Pearson, ISBN: 0132350882

17. Hashimi, H., & Gravell, A. (2020). Spikes in Agile Software Development: An Empirical Study. In IEEE International Conference on Computational Science and Computational Intelligence, CSCI-2020. pp. 1715-1721.

18. What is Spike in Scrum? https://www.visual-paradigm.com/scrum/what-is-scrum-spike/

19. Lee, S. W., Kim, H. K., & Lee, R. Y. (2008). Enterprise process model for extreme programming with CMMI framework. Computer and Information Science, 169-180.

20. O.Perminova, M.Gustafsson, K.Wikström (2008). Defining uncertainty in projects – a new perspective. International Journal of Project Management, Volume 26, Issue 1, 73-79. https://doi.org/10.1016/j.ijproman.2007.08.005.

21. Uncertainty vs. Risk: Key Concepts in Project Management. https://slm.mba/mmpo-002/uncertainty-vs-risk-in-project-management/

22. Ranasinghe, U., Jefferies, M., Davis, P., & Pillay, M. (2021). Conceptualising Project Uncertainty in the Context of Building Refurbishment Safety: A Systematic Review. Buildings, 11(3), 89. https://doi.org/10.3390/buildings11030089

23. Marinho M., Sampaio S., Lima T., Moura H. (2014). A Systematic Review of Uncertainties in Software Project Management. International Journal of Software Engineering & amp, vol. 5, no. 6, 1–21. https://doi.org/10.5121/ijsea.2014.5601.

24. The Uncertainty Principle. Stanford Encyclopedia of Philosophy. URL : https://plato.stanford.edu/entries/qt-uncertainty/

25. C. E. Shannon (1948). A Mathematical Theory of Communication. The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656

26. Prigogine I., Stengers I. (1984). Order out of chaos. A new dialogue between people and nature. A Bantam Book. 385.

27. Haji-Kazemi, S., Andersen, B., & Krane, H. P. (2013). A Review on Possible Approaches for Detecting Early Warning Signs in Projects. Project Management Journal, 44(5), 55-69. https://doi.org/10.1002/pmj.21360

28. Dokas, I. M., Feehan, J., & Imran, S. (2013). EWaSAP: An early warning sign identification approach based on a systemic hazard analysis. Safety science, 58, 11-26. https://doi.org/10.1016/j.ssci.2013.03.013.

29. Macedo, K., Marinho, M., & Santos, S. (2019). Uncertainty management in software projects: A case study in a public company. Journal of Convergence Information Technology (JCIT). Volume14, Number1, 61 – 67. https://arxiv.org/pdf/1902.05835

30. Weick, K. E. (1995). Sensemaking in organizations. Thousand Oaks, CA: Sage publications.

31. Путій І.Д., Бондар О.А. (2024). Складність IT-проєктів з науково-дослідною та дослідно-конструкторською розробкою . коблево 2024

32. Alex Hannaway. (2025) R&D Projects: Difference Between Projects And R&D. EmpowerRD. URL : R&D Projects: Difference Between Projects And R&D Projects

33. Morcov S., Pintelon L., Kusters R. (2021). IT Project Complexity Management Based on Sources and Effects: Positive, Appropriate and Negative. Proceedings of the Romanian Academy - Series A: Mathematics, Physics, Technical Sciences, Information Science. 329 – 336. URL : https://www.researchgate.net/publication/342314173_IT_Project_Complexity_Management_Based_on_Sources_and_Effects_Positive_Appropriate_and_Negative

34. Morandi, V. (2013). The management of industry–university joint research projects: how do partners coordinate and control R&D activities?. J Technol Transf 38, 69–92 https://doi.org/10.1007/s10961-011-9228-5

35. Ilbahar, E., Çebi, S., & Kahraman, C. (2021). Risk assessment of R&D projects: a new approach based on IVIF AHP and fuzzy axiomatic design. J. Intell. Fuzzy Syst., 42, 605-614. https://doi.org/10.3233/jifs-219215.

36. Shin, J., Lee, S., & Yoon, B. (2018). Identification and Prioritisation of Risk Factors in R&D Projects Based on an R&D Process Model. Sustainability, 10, 972. https://doi.org/10.3390/SU10040972.

37. Šmidovnik, T., & Grošelj, P. (2023). Solution for convergence problem in DEMATEL method: DEMATEL of finite sum of influences. Symmetry, 15(7), 1357. https://doi.org/10.3390/sym15071357

38. Luppino, R; Hosseini, MR; Rameezdeen, R (2014). Risk management in research and development (R&D) projects: the case of South Australia. Deakin University. Journal contribution. https://hdl.handle.net/10536/DRO/DU:30075627

39. Li, H., Chen, R., & Zhang, X. (2022). Uncertain Public R&D Project Portfolio Selection Considering Sectoral Balancing and Project Failure. Sustainability. https://doi.org/10.3390/su142315774.

40. Leal, L., Ribeiro, A., Romão, V., Amaral, G., Altmann, R., Kahn, R., Pacci, B., Avó, M., Salerno, M., Plonski, G., & Zancul, E. (2021). R&D approach based on multiple partners and Design Thinking, Lean Startup, and Agile concepts: case study in the electricity sector. . https://doi.org/10.14488/bjopm.2021.003.

41. Idowu, S., Osman, O., Strüber, D., & Berger, T. (2022). On the Effectiveness of Machine Learning Experiment Management Tools. 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 207-208. https://doi.org/10.1145/3510457.3513084.

42. Путій І. Д., Тесленко П. О. (2025). «Канвас» методу формування гіпотез для складного ІТ-проєкту. Управління розвитком складних систем, 64, 120–127. DOI: 10.32347/2412-9933.2025.64.

43. Шумка М. (2004). Гіпотеза. http://dspace.wunu.edu.ua/bitstream/316497.4881.1.%B0%И3%B1%96%B0%ИА%B0%ИУ%B1%82%B0%И5%B0%И7%B0%И0%20%B0%ИС%B0%И0%B1%82%B0%И5%B1%80%B1%96%B0%И0%B0%ИИ%B0%И8%20%B0%И4%B0%ИУ%20%B0%ИИ%B0%И5%B0%ИФ%B1%86%B1%96%B1%97ю3ВА

44. Колесников О.В. (2008). Основи наукових досліджень. Навч. посібник. Харків. УкрДАЗТ, 170.

45. A., Bulajic, a., Stamatović, M., & Cvetanović, S. (2012). The importance of defining the hypothesis in scientific research. International Journal of Educational Administration and Policy Studies, 4, 170-176. https://doi.org/10.5897/IJEAPS12.009.

46. Gonçalves, G., Fernandes, J. A., & Gonçalves, J. J. (2020). Learning Hypothesis Testing Through a Project Work Methodology. In Developing Technology Mediation in Learning Environments (pp. 221-238). IGI Global.

47. Гороховатський В.О., Творошенко І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник. НУРЕ, 92. DOI: 10.30837/978-966-659-298-2

48. Dmytro Shestakov (2021). The Hypotheses Testing Method for Evaluation of Startup Projects. Journal of Economics and Management Sciences. https://doi.org/10.30560/jems.v4n4p47.

49. Hannu Kivijärvi (2020). Theorizing IT Project Success: Direct and Indirect Effects in a Hierarchical Framework. International Journal of Information Technology Project Management, 11(1), 71-98. https://doi.org/10.4018/ijitpm.2020010105

50. Miecznikowski, J., & Wang, J. (2022). Error control in tree structured hypothesis testing. Wiley Interdisciplinary Reviews: Computational Statistics, 15. https://doi.org/10.1002/wics.1603

51. Roy, J., Duclos-Hindié, N., & Dessureault, D. (1997). Efficient cluster management algorithm for multiple-hypothesis tracking. , 3163. https://doi.org/10.1117/12.279526.

52. Eric Ries. (2011). The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful. Crown Currency; First Edition. 336. ISBN-10 : 9780307887894

53. Ximenes, B., Alves, I., & Araújo, C. (2015). Software Project Management Combining Agile, Lean Startup and Design Thinking. , 356-367. https://doi.org/10.1007/978-3-319-20886-2_34.

54. Yordanova, Z. (2017). Knowledge transfer from lean startup method to project management for boosting innovation projects' performance. International Journal of Technological Learning, Innovation and Development, 9, 293. https://doi.org/10.1504/IJTLID.2017.10010040.

55. Jeremy Nelson (2016). Becoming a Lean Library. Lessons from the World of Technology Start-ups. Published by Elsevier, 148. ISBN: 978-1-84334-7798

56. Nola, R. (2007). Theories of Scientific Method: The hypothetico-deductive method. , 170-184. https://doi.org/10.1017/UPO9781844653881.008.

57. Siponen, M., & Klaavuniemi, T. (2020). Why is the hypothetico-deductive (H-D) method in information systems not an H-D method?. Inf. Organ., 30, 100287. https://doi.org/10.1016/j.infoandorg.2020.100287.

58. Лях, І. М., Білак, Ю. Ю., Данько-Товтин, Л. Я., & Станишевський, В. В. (2015). Використання перевірки статистичних гіпотез в інформаційно-технічній сфері. A/B тестування та доцільність його застосування. Квалілогія книги, (2), 82-85.

59. Souza, W., Pereira, F., Albuquerque, V., Melegati, J., & Guerra, E. (2022). A Framework Model to Support A/B Tests at the Class and Component Level. 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), 860-865. https://doi.org/10.1109/COMPSAC54236.2022.00136.

60. Shestakov, D. (2021). The Hypotheses Testing Method for Evaluation of Startup Projects. Journal of Economics and Management Sciences. https://doi.org/10.30560/jems.v4n4p47.

61. Kronsbein, T., & Müller, R. (2019). Data Thinking: A Canvas for Data-Driven Ideation Workshops. , 1-10. https://doi.org/10.24251/HICSS.2019.069.

62. Silva, H., & Cardoso, A. (2019). Research Project Model Canvas. Computer Science and Information Technology. https://doi.org/10.13189/CSIT.2019.070301.

63. Habermann, F., & Schmidt, K. (2020). The project canvas: five years evolution of a project management instrument. International Journal of Management Practice. https://doi.org/10.1504/ijmp.2020.10027281.

64. Hoffmann, M., & Drath, R. (2022). How to survive a PhD – using Design Thinking methods and the Business Model Canvas. 2022 IEEE Global Engineering Education Conference (EDUCON), 1652-1657. https://doi.org/10.1109/EDUCON52537.2022.9766528.

65. Silva, H., & Cardoso, A. (2019). Research Project Model Canvas. Computer Science and Information Technology. https://doi.org/10.13189/CSIT.2019.070301.

66. Melegati, J., Wang, X., & Abrahamsson, P. (2019). Hypotheses Engineering: First Essential Steps of Experiment-Driven Software Development. 2019 IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE), 16-19. https://doi.org/10.1109/RCoSE/DDrEE.2019.00011

67. Maximize Collaboration and Efficiency with Atlassian Jira (2023). URL : https://www.consumersearch.com/technology/maximize-collaboration-efficiency-atlassian-jira-try-free?utm_content=params%3Aad%3DdirN%26qo%3DserpIndex%26o%3D740007%26ag%3Dfw10&ueid=6D33CCF1-9C8F-46A9-853D-22994DCD312E&origq=atlassian+jira+confluence

68. Trello для стартапів. URL : https://trello.com/teams/startups

69. Asana. URL : https://cloudfresh.com/ua/produkty/asana/

70. Microsoft Project. URL : https://www.microsoft.com/uk-ua/microsoft-365/project/project-management-software

71. Monday workdocs:where words ignite workflows. URL : https://monday.com/

72. ClickUp. URL : https://clickup.com/features/portfolios

73. Redmine. URL : https://www.redmine.org/projects/redmine

74. Basecamp. URL : https://basecamp.com/

75. Wrike. URL : https://www.wrike.com/partnertrial/?utm_source=partner&utm_medium=referral_program&utm_campaign=partnerstack_other&pscd=get.wrike.com&ps_partner_key=c2VtYW50aWNsYWJz&sid=1-g-Cj0KCQjwrp-3BhDgARIsAEWJ6SyDcLKxx_bLMXeOZo3CzYO2vC_Fl026HHmJToEPw_h_GAZgwUley4UaAiUgEALw_wcB&gad_source=1&gclid=Cj0KCQjwrp-3BhDgARIsAEWJ6SyDcLKxx_bLMXeOZo3CzYO2vC_Fl026HHmJToEPw_h_GAZgwUley4UaAiUgEALw_wcB&ps_xid=h9XgSVTlhTWu6o&gsxid=h9XgSVTlhTWu6o&gspk=c2VtYW50aWNsYWJz

76. GitLab. URL : https://about.gitlab.com/

77. Zoho Projects. URL : https://www.zoho.com/projects/

78. Smartsheet URL : https://www.smartsheet.com/

79. Amarasekara, T., Isurindi, H., Navanjana, E., Gamage, O., Samarakoon, U., & Kugathasan, A. (2021). Project Zone : An Advanced Undergraduate Project Management System For Software Development. 2021 21st International Conference on Advances in ICT for Emerging Regions (ICter), 183-188. https://doi.org/10.1109/ICter53630.2021.9774820.

80. Diamantopoulos, T., Nastos, D., & Symeonidis, A. (2023). Semantically-enriched Jira Issue Tracking Data. 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), 218-222. https://doi.org/10.1109/MSR59073.2023.00039.

81. Puliak, O., & Myronenko, N. (2023). PROSPECTS OF USING MODERN TASK MANAGERS FOR IT PROJECT MANAGEMENT. Academic Notes Series Pedagogical Science. https://doi.org/10.36550/2415-7988-2023-1-208-213-218.

82. Vaz-Cedillo, J., González-Rocha, R., Torres-Gil, M., Rodríguez-González, N., Fernández-Izquierdo, P., Quintero-Nehrkorn, J., Hernando, Y., Núñez-Cagigal, M., García, F., Vega-Reyes, N., Galarreta, C., Cózar-Castellano, J., Martínez, Á., González-Cava, J., Femenía-Castellá, B., Padilla-Hernández, C., Carballo-Martín, M., & González-Pérez, F. (2022). CosmoSys-Req: a free open-source requirements

management tool. , 12187, 121871K - 121871K-20. https://doi.org/10.1117/12.2629628.

83. Carvalho, E., Malcher, P., & Santos, R. (2020). A Survey Research on the Use of Mobile Applications in Software Project Management. Proceedings of the XIX Brazilian Symposium on Software Quality. https://doi.org/10.1145/3439961.3439963.

84. Evangelista, E., Sofianti, T., & Baskoro, G. (2022). Analytical Hierarchy Process (AHP) for Selection of Project Management Software to Support Remote Working: A Case Study at Logistics Division of a National EPC Company in Indonesia. Proceedings of the 2022 International Conference on Engineering and Information Technology for Sustainable Industry. https://doi.org/10.1145/3557738.3557828.

85. A platform built for the new way of doing product management. URL : https://airfocus.com/

86. End-to-end product management platform with best practices built-in URL : https://craft.io/

87. Product experimentation skillsfrom idea to scale URL : https://experimentengines.com/

88. Notion: Write. Plan. Collaborate. URL : https://translate.google.com/?sl=en&tl=uk&text=Write.%20Plan.%20Collaborate.%0AIn%20Notion%2C%20work%20feels%20better%20(and%20gets%20done%20faster).%20With%20a%20little%20help%20from%20AI.&op=translate

89. Confluence documentation URL : https://www.refined.com/use-case/documentation-sites?utm_term=&utm_campaign=Refined+-+Dynamic&utm_source=adwords&utm_medium=ppc&hsa_acc=9066151006&hsa_cam=19923863263&hsa_grp=148601193038&hsa_ad=676333357349&hsa_src=g&hsa_tgt=dsa-19959388920&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAjwgfm3BhBeEiwAFfxrG1sZ_WhyF9Q332TE46oAdxUZDoiUbaTaMuvbsTIEclTm-RoS5GzGYRoCw0EQAvD_BwE

РОЗДІЛ 2. МОДЕЛІ СКЛАДНИХ ІТ-ПРОЄКТІВ В УМОВАХ НЕВИЗНАЧЕНОСТІ

2.1. Інформаційна модель складного ІТ-проєкту з науково-дослідницькою та дослідно-конструкторською складовою

Інформаційна модель складного ІТ-проєкту з науково-дослідницькою та дослідно-конструкторською складовою є структурованим представленням основних елементів і взаємозв'язків, що забезпечують системне управління розробкою. Інформаційна модель використовується для формалізації складних процесів в умовах невизначеності, що включають різноманітні аспекти реалізації складних ІТ-проєктів. Вона дозволяє визначити складові, їх функції, взаємодію та вплив на результат. Використання цього терміну обумовлене необхідністю інтегрованого підходу до планування, досліджень, управління ризиками, вибору технологій та архітектурних рішень. Інформаційна модель дозволяє систематизувати процес розробки та забезпечити взаємозв'язок між усіма етапами проєкту [1].

У суміжному просторі системного аналізу комп'ютерних наук та управління проєктами, можна виділити такі варіації інформаційних моделей.

1. Структурна інформаційна модель – відображає компоненти системи та їхні зв'язки у вигляді графів, схем або діаграм.

2. Процесна інформаційна модель – акцентує увагу на потоках інформації між компонентами системи та правилах її обробки.

3. Логічна інформаційна модель – деталізує зв'язки між даними, їх логічні залежності та структуру інформаційних об'єктів.

4. Фізична інформаційна модель – описує, як інформація зберігається та передається на рівні конкретної реалізації (бази даних, файлові системи).

5. Динамічна інформаційна модель – моделює зміну інформаційних потоків у часі, що важливо для аналізу поведінки систем.

6. Доменна інформаційна модель – специфічна для конкретної предметної області, наприклад, фінансова, медична чи промислова.

Доменна інформаційна модель – це модель, яка описує структуру, зв’язки та правила обробки інформації у конкретній предметній області (домені). Вона визначає, які дані існують у межах цього домену, як вони взаємодіють між собою, які процеси ними керують та які обмеження накладаються на їх використання [2]. Основна мета такої моделі – уніфікація представлення інформації для узгодженої роботи всіх учасників проєкту, включаючи системи, що взаємодіють між собою.

Домен, або предметна область, в у контексті інформаційних моделей – це специфічна сфера діяльності або система, в межах якої функціонує інформаційна модель. Наприклад, у сфері охорони здоров’я домен охоплює дані про пацієнтів, медичні записи, діагнози та процедури. В контексті IT-проєктів домен включатиме управління вимогами, командну взаємодію, розгортання програмного забезпечення, безпеку тощо.

Зміст доменної інформаційної моделі містить такі сутності [3, 4].

1. Об'єкти домену – ключові сутності, які описують систему: користувачі, процеси, документи, компоненти системи.

2. Атрибути об'єктів – характеристики кожного об'єкта.

3. Зв’язки між об'єктами – як взаємодіють сутності, наприклад, один стейкхолдер або член команди може мати кілька ролей у проєкті.

4. Інформаційні потоки – опис руху даних у системі, їх джерела та кінцеві точки.

5. Процеси обробки даних – алгоритми роботи з інформацією, правила її зміни, доступу та збереження.

6. Обмеження та бізнес-правила – правила, які визначають алгоритм роботи об'єктів.

7. Моделі взаємодії з іншими доменами – як одна інформаційна модель інтегрується з іншими системами.

Для складного IT-проєкту доменна інформаційна модель складатиметься з таких елементів, або піддоменів:

– домен управління проєктом (планування, завдання, контроль термінів);

– домен стейкхолдерів: розробники, менеджери, тестувальники, дослідники;

– домен коду та архітектури: структура вихідного коду, бібліотеки, інтеграції);

– домен безпеки: контроль доступу, аутентифікація, захист даних.

Кожен із цих доменів може мати свою інформаційну модель, що описує його сутності, процеси та взаємозв’язки. Наприклад, об'єкт «користувач» у домені безпеки має атрибути «роль», «рівень доступу», «статус». А між ним та об'єктом «система аутентифікації» є зв'язок «проходження перевірки доступу». Зазначимо, що структурна та доменна інформаційні моделі мають певну схожість, оскільки обидві описують компоненти реального об’єкта, але вони мають різні цілі, рівень абстракції та застосування. Різницю між структурною та доменною інформаційною моделлю розглянемо через сутності: призначення, рівень абстракції, об'єкти моделювання, зв'язки між елементами [5].

1. Призначення інформаційних моделей:

– структурна інформаційна модель описує статичну архітектурну організацію системи, її компоненти та взаємозв’язки між ними;

– доменна інформаційна модель фокусується на предметній області, визначає її ключові сутності, правила обробки даних, процеси та бізнес-логіку в межах конкретного домену.

2. Рівень абстракції моделей:

– структурна модель це технічне представлення, яке показує, як побудована система, її компоненти та як вони між собою з'єднані;

– доменна модель це концептуальне представлення інформації про предметну область, призначена для аналізу вимог, узгодження між елементами структри, що мають різну природу.

3. Об'єкти моделей:

– структурна модель складається з системних компонентів, таких як модулі, бази даних, API, тощо;

– доменна модель складається з бізнес-сутностей, таких як користувачі, транзакції, документи, процеси, тощо.

4. Зв'язки між елементами моделей:

– структурна модель описує зв’язки з точки зору фізичного або логічного з’єднання між компонентами;

– доменна модель описує бізнес-зв’язки між сутностями.

У якості висновка можна сказати, що структурна інформаційна модель показує «як побудована система» та її технічну організацію, натомість доменна інформаційна модель описує «що ця система обробляє», які дані та бізнес-правила у ній діють. Тобто, вони доповнюють одна одну, але не є тотожними. Тому в роботі розроблено саме доменну інформаційну модель для виокремлення сутностей та елементів складних ІТ-проєктів, їх взаємодію між собою. В даному випадку, доменом є складний ІТ-проєкт, в якому складність долається через проведення наукових досліджень, пошуку та перевірки шляхом розробки гіпотез та/або прототипу в межах дослідно-конструкторських розробок через визначення ефективності пропонованих проєктних рішень як логічна схема до дій. Схема цієї доменної інформаційної моделі подана на рисунку 2.1.

Розглянемо основні елементи доменної інформаційної моделі складного ІТ-проєкту з НДДКР [6]. Стейкхолдери складного ІТ-проєкту (елемент 1 – далі по тексту скорочено: Е1) є ключовими учасниками процесу, що формують бачення, визначають стратегічні цілі та очікування від проєкту. Вони включають замовників, кінцевих користувачів, інвесторів, регуляторні органи та технічних експертів, дослідників тощо. Їхня участь критично важлива для встановлення реалістичних вимог та забезпечення узгодженості між всіма сторонами складного ІТ-проєкту [7]. Саме стейкхолдери формують поле невизначеності у складних ІТ-проєктах.

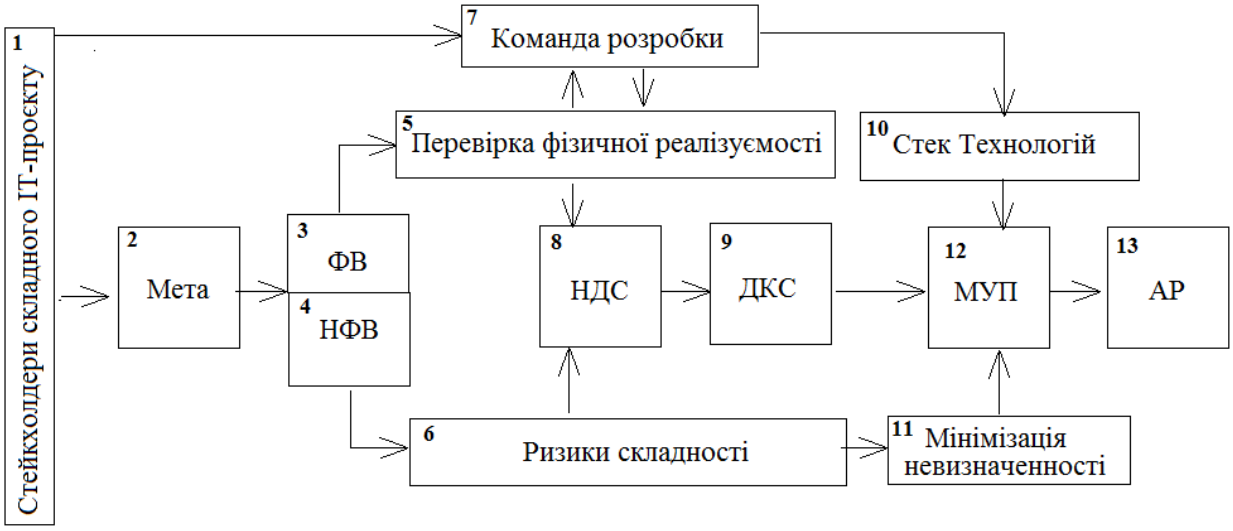


Рисунок 2.1 – Схема доменної інформаційної моделі складного ІТ-проєкту [1]

Мета проєкту (E2) визначає усі інші складові моделі. Узгоджує очікування стейкхолдерів, встановлює критерії успіху та окреслює етапи реалізації. Мета має бути досяжною, вимірюваною та орієнтованою на кінцевий результат. Функціональні вимоги (E3), як елемент поля невизначеності, формують набір можливостей і функціоналу, які повинен мати продукт проєкту. Вони є основою для проєктування архітектури, вибору технологічного стеку та оцінки ефективності реалізації. Вимоги охоплюють бізнес-логіку, інтеграцію з іншими системами, алгоритми обробки даних та користувацький інтерфейс.

Нефункціональні вимоги (E4) включають параметри продуктивності, масштабованості, безпеки, доступності, сумісності та інших параметрів, які визначають якість кінцевого продукту та відповідність його стандартам.

Перевірка фізичної реалізуємості (E5) є ключовим етапом, що визначає можливість команди реалізувати проєкт з точки зору наявних технологій, компетенцій та ресурсів. В E5 невизначеність формує управлінські дії через необхідність додаткового навчання, розробку нових алгоритмів чи досліджень для досягнення цілей проєкту. Це дозволяє вчасно виявити потенційні проблеми та мінімізувати ризики. Ризики складності (E6), це калькульовані

прояви невизначеності. Цей елемент визначає потенційні загрози, що виникають через складність проєкту. До таких ризиків можуть належати невизначеність у вимогах, складність інтеграції, залежність від зовнішніх факторів, технічні обмеження та нестабільність технологічного середовища. Оцінка ризиків дозволяє розробити стратегії їхньої мінімізації та забезпечити стійкість та успішне завершення проєкту.

Команда розробки (E7) є основним виконавчим ресурсом IT-проєкту. Вона визначає компетенції, необхідні для реалізації, розподіл обов'язків, організацію комунікації та управління процесами розробки. Ефективне управління командою має нівелювати ризики через зменшення невизначеності. Науково-дослідницька складова (НДС) (E8) передбачає виконання досліджень для розв'язання невизначеностей, розробки нових підходів, тестування гіпотез та пошуку оптимальних рішень. Вона є важливим елементом складних проєктів, в яких необхідні інновації або відсутні готові рішення.

Дослідно-конструкторська складова (ДКС) (E9) включає прототипування, моделювання, тестування та оптимізацію технічних рішень. Її основною метою є підтвердження життєздатності концепцій, які були розроблені на етапі науково-дослідницької діяльності, та перевірка їхньої ефективності в реальних умовах. Технологічний стек (E10) визначає набір мов програмування, фреймворків, баз даних, серверних технологій та інших компонентів, які будуть використані в розробці. Вибір технологій має відповідати вимогам продуктивності, безпеки, гнучкості та масштабованості.

Мінімізація невизначеності (E11) є підходом до управління ризиками шляхом впровадження механізмів прототипування, тестування, розбиття завдань на менші ітерації, використання методологій швидкої розробки та впровадження інструментів прогнозування ризиків. Знижує вплив невизначеності та підвищує передбачуваність результатів. Методологія управління та розробки (МУП) (E12) проєкту обирається на основі специфіки завдань, рівня складності, команди та бізнес-вимог. Вона може включати

Agile, Scrum, Kanban, Waterfall або їхні комбінації для забезпечення ефективного управління процесами розробки.

Архітектурне рішення (AP) (E13) є завершальним етапом побудови моделі. Воно визначає логічну, фізичну та інформаційну структуру продукту проєкту, забезпечує баланс між продуктивністю, безпекою, зручністю використання та масштабованістю.

Математична модель такої доменної інформаційної моделі складного IT-проєкту з НДДКР слугує формальним інструментом для опису його структури, взаємозв'язків і процесів обробки інформації в межах конкретної предметної області. Серед основних типів математичних моделей, які можуть бути застосовані для опису інформаційних потоків у таких проєктах, виділяють: теорію графів, яка дозволяє описати вузли системи та зв'язки між ними у вигляді орієнтованих графів [8]; мережі масового обслуговування, що моделюють черги та обробку заявок у системах із різною пропускною здатністю [9]; стохастичні процеси, такі як марковські ланцюги, які корисні при моделюванні ймовірнісних переходів між станами [10]; динамічні системи на основі диференціальних рівнянь для аналізу змін параметрів у часі [11]; теорію інформації для вимірювання кількості інформації та визначення ефективності каналів передачі [12]; мережі Петрі, що добре моделюють паралелізм та взаємозалежності у процесах [13]; алгебру логіки, або булеву алгебру, яка дозволяє описувати логічні умови, правила і залежності у вигляді формул [14]; теорію категорій та онтологічні підходи для концептуального узагальнення сутностей і відношень [15].

Серед зазначених підходів найдоцільнішим для побудови математичної моделі доменної інформаційної структури складного IT-проєкту з НДДКР є модель на основі булевої алгебри. Це зумовлено тим, що предметна область такого проєкту характеризується складною логічною структурою, наявністю умов доступу, переходів між етапами, залежністю між процесами та перевірок виконання визначених критеріїв. Булева алгебра дозволяє компактно та формально описати ці залежності у вигляді логічних формул. До переваг

булевої моделі належать її формальна строгость, можливість використання в автоматизованих системах перевірки умов (наприклад, у CASE-засобах), універсальність опису логіки взаємодії та ефективність реалізації у програмному середовищі.

Водночас до недоліків відносять її статичність і обмежену придатність до аналізу динамічних або ймовірнісних змін, такі як поведінка у часі або випадкові збої. Саме тому інші типи моделей, попри свою корисність у специфічних задачах, менш придатні для побудови цілісної доменної моделі. Наприклад, теорія графів добре підходить для структурного аналізу, але не виражає логіки бізнес-правил. Стохастичні та динамічні моделі є складними для реалізації у вигляді правил прийняття рішень. Теорія інформації орієнтована на кількісні характеристики, а не на семантичну або логічну структуру предметної області. Таким чином, булева модель краще формалізує логічну поведінку сутностей, їх взаємодію та зв’язки в доменній інформаційній моделі складного ІТ-проєкту, що має науково-дослідницьку та дослідно-конструкторську складову.

Формальна булева модель для всієї доменної інформаційної моделі складного ІТ-проєкту з НДДКР представляє домени як множини сутностей, пов’язаних логічними умовами. Для зручності, кожен домен описаний через змінні-сутності та логічні формули, що визначають зв’язки між ними:

Для моделі прийняті такі позначення:

- E1 — Стейкхолдери
- E2 — Мета проєкту
- E3 — Функціональні вимоги
- E4 — Нефункціональні вимоги
- E5 — Перевірка фізичної реалізуємості
- E6 — Ризики складності
- E7 — Команда розробки
- E8 — Науково-дослідницька складова

– E9 — Дослідно-конструкторська складова

– E10 — Технологічний стек

– E11 — Мінімізація невизначеності

– E12 — Методологія управління та розробки

– E13 — Архітектурне рішення

Взаємозв’язки формалізовані наступним чином:

1. Визначення мети проєкту базується на бачені стейкхолдерів:

$$E2 = f(E1),$$

тобто мета формується на основі інтересів усіх стейкхолдерів.

2. Вимоги формуються з мети:

$$E3 \wedge E4 = f(E2).$$

Функціональні та нефункціональні вимоги задаються цільовим баченням.

3. Перевірка фізичної реалізуємості залежить від вимог і команди:

$$E5 = E3 \wedge E4 \wedge E7 \wedge E10.$$

Проєкт фізично реалізуємий, якщо вимоги відповідають можливостям команди і технологіям.

4. Ризики складності виникають, якщо реалізуємість не доведена або вимоги нестабільні:

$$E6 = \neg E5 \vee \Delta(E3 \vee E4).$$

Ризики зростають при відсутності реалізуємості або змінності вимог.

5. Якщо існують ризики або недостатність компетенцій, активуються НДДКР:

$$E8 \vee E9 = E6 \wedge \neg E5.$$

НДДКР виконуються, коли існує складність і проєкт не може бути реалізовним стандартними засобами.

6. Результати НДДКР впливають на остаточний стек і мінімізують невизначеність:

$$E10 = f(E8 \wedge E9),$$

$$E11 = E8 \wedge E9 \wedge f(E6).$$

Технології та зменшення невизначеності прямо залежать від НДДКР.

7. Вибір методології управління залежить від характеру вимог і типу ризиків:

$$E12 = f(E3 \wedge E4 \wedge E6 \wedge E10).$$

Методологія узгоджується з вимогами, ризиками, технологіями.

8. Архітектурне рішення формується на основі: вимог, реалізуємості, результатів НДДКР, мети:

$$E13 = E3 \wedge E4 \wedge E5 \wedge E8 \wedge E9 \wedge E2.$$

Архітектура проєкту логічно завершує цикл.

Узагальнена логічна формула моделі матиме наступний вигляд:

$$E13=f(E1,f(E1)\rightarrow E2,f(E2)\rightarrow(E3\wedge E4),(E3\wedge E4\wedge E7\wedge E10)\rightarrow E5,(\neg E5\vee\Delta(E3\vee E4))\rightarrow E6,(E6\wedge\neg E5)\rightarrow(E8\vee E9),(E8\wedge E9)\rightarrow(E10\wedge E11),f(E3,E4,E6,E10)\rightarrow E12,f(E3,E4,E5,E8,E9,E2)\rightarrow E13)E13 = f(E1, f(E1) \rightarrow E2, f(E2) \rightarrow (E3 \wedge E4), (E3 \wedge E4 \wedge E7 \wedge E10) \rightarrow E5, (\neg E5 \vee \Delta(E3 \vee E4)) \rightarrow E6, (E6 \wedge \neg E5) \rightarrow (E8 \vee E9), (E8 \wedge E9) \rightarrow (E10 \wedge E11), f(E3, E4, E6, E10) \rightarrow E12, f(E3, E4, E5, E8, E9, E2) \rightarrow E13)$$

Ця булева модель показує взаємозалежність усіх ключових елементів доменної інформаційної структури. Вона може бути використаною для логічного аналізу взаємодії сутностей, валідації послідовності процесів та побудови базових обчислювальних процедур для системи підтримки прийняття рішень у межах складного ІТ-проєкту з НДДКР.

Таким чином, інформаційна модель складного ІТ-проєкту з науково-дослідницькою та дослідно-конструкторською складовою системно представляє всі його процеси, визначає ключові аспекти його реалізації, забезпечує ефективне управління невизначеністю та знижує ризики складності. Вона дозволяє інтегрувати науково-дослідницькі та дослідно-конструкторські підходи в процес розробки, що є необхідним для створення інноваційних та технологічно складних продуктів.

2.2 Концептуальна модель управління складними ІТ-проєктами в умовах невизначеності

Концептуальна модель — це абстрактне уявлення системи або процесу, що відображає основні компоненти та взаємозв'язки між ними, допомагає виявити та зрозуміти сутність явища та його подальшу логіку розвитку. Термін «концептуальна» походить від латинського «conceptio», що означає «зачаття», «поняття» або «система», підкреслює ідею формування загального уявлення про об'єкт або явище.

Українські та зарубіжні вчені надають різні визначення концептуальної моделі залежно від галузі дослідження. У філософії концептуальна модель розглядається як інструмент для структурування та аналізу абстрактних понять і ідей. У технічних науках вона використовується для формалізації системних уявлень, що допомагає виявити ключові елементи та їх взаємодію.

Наприклад, у статті «Концептуальна модель бази даних збірки М. Номиса «Українські приказки, прислів’я і таке інше» автори І.М. Кульчицький та Н.Б. Осідач описують процес створення концептуальної моделі для лексикографічної бази даних, що сприяє збереженню культурної спадщини України [16].

Застосування концептуальних моделей у проєктуванні та управлінні організаційно-технічними системами дозволяє визначити структуру системи, взаємозв'язки між її компонентами та основні функції. Це сприяє ефективному плануванню, розробці та впровадженню систем, забезпечує узгодженість між етапами проєкту та комунікацію між його учасниками.

В управлінні проєктами концептуальна модель має достатньо широке використання від організації проєктного офісу, формування ризиків стейкхолдерів, опису портфельного управління до елементів цифрового суспільства [17-22].

В нашому випадку, результатом концептуального моделювання складного ІТ-проєкту з НДДКР є формалізована структура управління, що визначає ключові компоненти, їх взаємозв’язки та процеси реалізації. Це

дозволить оптимізувати ресурсне планування, розподілити ролі та зони відповідальності між учасниками проєкту, встановити механізми прийняття рішень та впровадити процедуру оцінки готовності проєкту до наступних етапів. Проблема, для вирішення якої розробляється концептуальна модель є невизначеність під час розробки продукту проєкту. Модель має виявити точки ризиків складності задля зменшення невизначеності через застосування Spikes, наукових досліджень, PoC, дослідно-конструкторських розробок згенерованих рішень, прототипування та інших методів. Вона забезпечує логічну інтеграцію наукових досліджень у процес розробки, сприяє швидкому адаптуванню проєкту до змінних умов і дозволяє ефективно трансформувати результати НДДКР у робочі технічні рішення.

Наявність в проєкті науково-дослідної складової унеможливлює заздалегідь планувати остаточну структуру проєкту, тому що її елементи будуть залежати від результатів дослідження, експериментальних випробувань, тощо. Крім того, існує варіант, коли вже затверджену структуру проєкту може бути змінено у зв’язку з результатами поточних досліджень. Це вимагатиме у загальній структурі управління проєктом процедуру перепроєктування, для якої вирішено використовувати термін «рефакторинг».

Термін «рефакторинг» застосований з ІТ галузі, де він означає процес покращення внутрішньої структури коду без зміни його зовнішньої поведінки. Це метод оптимізації, який робить програму більш читабельною, ефективною та підтримуваною, але без додавання нового функціоналу [23].

До методів рефакторингу відносять:

- заміну довгих методів на менші функції, так званий «чистий код» з принципом DRY;
- перейменування змінних і класів для покращення читабельності;
- виділення повторюваних блоків у загальні модулі;
- перехід від монолітної архітектури до модульної або мікросервісної.

Таким чином, можемо стверджувати, що рефакторинг – це процес безперервного покращення якості коду, який робить його простішим у

розумінні та супроводі. Він є важливою частиною життєвого циклу розробки, особливо в довготривалих ІТ-проєктах.

Визначення 2.1. ***Рефакторинг в управлінні ІТ-проєктом*** може бути інтерпретований як процес перегляду, оптимізації або покращення процесів, архітектури, методів управління та розробки без зміни функціональності проєкту.

Це аналогічно технічному рефакторингу коду [23], коли структура покращується для більшої ефективності, але зовнішня поведінка залишається незмінною. Зазначимо, що процедура рефакторингу вже набула широкого застосування. Наприклад до управління базами даних [24], або проведення судових експертиз (експертних досліджень) [25], тощо.

Рефакторинг до системи управління проєктами необхідний для можливого корегування системи під час проведення наукових досліджень та після них, під час випробувань прийнятих рішень та тестування прототипів. Рефакторинг включатиме:

– оновлення методології управління (наприклад, перехід з Waterfall на Agile через зміни у вимогах, або за результатами досліджень);

– оптимізацію робочих процесів команди (перерозподіл ролей, зміна структури спринтів, навчання, перевірка технологій розробки);

– поліпшення архітектури проєкту (перехід на мікросервісну архітектуру для масштабованості, проведення додаткових досліджень, проведення перевірки обраних рішень);

– автоматизацію процесів розгортання (впровадження CI/CD);

– покращення комунікацій між різними підрозділами команди, стейкхолдерів, замовників, тощо.

Розроблена модель зробить управління проєктом прогнозованим, гнучким і науково обґрунтованим, що важливо для складних ІТ-розробок [26]. В результаті дисертаційних досліджень була запропонована така структура концептуальної моделі управління складними ІТ-проєктами в умовах невизначеності (рисунок 2.2).

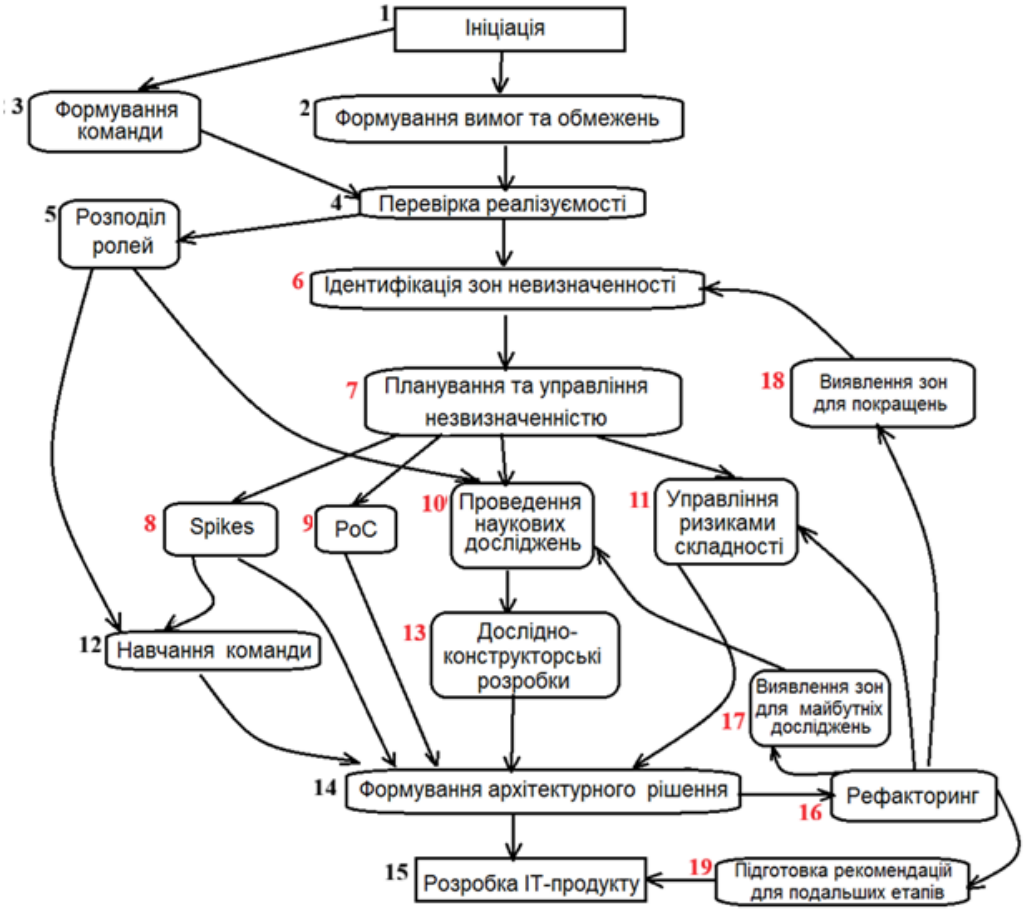


Рисунок 2.2 – Концептуальна модель управління складним IT-проєктом з НДДКР в умовах невизначеності [26]

На рисунку червоним кольором позначені номери блоків моделі, які запропоновані автором, та відсутні у стандартних моделях управління IT-проєктами. Новизна розробки полягає у впровадженні низки заходів для роботи з невизначеністю в проєкті. Вона стартує з ідентифікації невизначеності (блок 6 – дали Б6) та завершується рефакторингом управління проєктом (після зниження невизначеності) та розробкою рекомендацій до майбутніх досліджень та етапів проєкту (Б 16 –19).

Концептуальна модель управління складним IT-проєктом з НДДКР складається з таких елементів. Першим елементом моделі є класична «ініціація» (Б1). Завершується модель елементом «Розробка IT-продукту» (Б15). Тобто модель містить елементи, що пов’язані із ідентифікацією та управлінням невизначеністю, що пов’язана зі складністю, проведенням

наукових досліджень для її подолання, та дослідно-конструкторських розробок для перевірки отриманих результатів перед передачею їх у розробку. Далі розробка здійснюється за методологією управління ІТ-проєктами яка буде оптимальною для конкретного проєкту і цей етап розробки не входить до концептуальної моделі.

Ініціація проєкту передбачає визначення стратегічної мети та ключових цілей проєкту, окреслення стейкхолдерів та їхніх очікувань, попередній аналіз бізнес-вимог (Б1).

Формування вимог та обмежень уключає збір функціональних та нефункціональних вимог, оцінку технологічних, ресурсних та часових обмежень, ідентифікацію критичних факторів успіху проєкту (Б2).

Формування команди уключає створення команди управління, розробки та дослідницької групи. Визначення їх ключових виконавців та обов'язків (Б3).

Перевірка реалізуємості проєкту уключає аналіз наявних та доступних технологій, методів розробки. Оцінку компетенцій команди та потреба у її навчанні. Визначення необхідності проведення науково-дослідницьких робіт (Б4).

По завершенні виконується розподіл ролей і розробка комунікаційної стратегії та ідентифікація зон невизначеності, планування та управління ними (Б5).

Для управління невизначеністю (Б6–7) в роботі запропоновано використовувати наступні підходи:

1. Spikes (Б8).

2. Застосування PoC (Б9).

3. Проведення наукових досліджень (Б10).

4. Управління ризиками складності. Як було заначено у першому розділі в роботі розглядається складність реалізації проєкту (Б11).

PoC (Proof of Concept) — це метод експериментальної перевірки концепції, яка дозволяє довести технічну реалізовність функціоналу перед

повноцінною розробкою. Щоб довести, що функціонал продукту можна реалізувати, необхідно створити тестовий сценарій із мінімальним набором можливостей, що імітує ключові процеси майбутньої системи. Це допомагає оцінити можливі технічні обмеження та вибрати оптимальну архітектуру. Для перевірки працездатності обраної технології в реальних умовах створюється прототип або тестове середовище, в якому аналізуються продуктивність, сумісність та безпека. Це дозволяє оцінити ефективність рішення під реальними навантаженнями. PoC також дозволяє виявити слабкі місця та ризики перед основним впровадженням. Для переконання інвесторів або замовників у прибутковості ідеї використовуються результати PoC, які демонструють життєздатність концепції, підтверджують ефективність технології та оцінюють потенційний фінансовий вплив на бізнес.

Навчання команди застосовується для ліквідації прогалин у компетенціях та навичок для розробки продукту проєкту або для проведення досліджень (Б12).

Дослідно-конструкторські розробки в продовж яких теоретичні гіпотези перевіряються через прототипування та тестування (Б13). Вони можуть виконуватися після проведення наукових досліджень, або як різновид Spikes.

Після проведення наукових досліджень, Spikes, навчання команди, дослідно-конструкторських розробок та оцінки ризиків складності, формується архітектурне рішення продукту складного IT-проєкту (Б14). У якості зворотного зв’язку в моделі запропоновано рефакторинг як процедура перепроєктування структури проєкту (Б16). Рефакторинг не змінює кінцеву мету проєкту, але значно покращує процес його реалізації, а значить і має збільшити імовірність успішного завершення.

В моделі процедуру рефакторингу розподіллено на три складові. 1) Виявлення зон для покращень (Б18). 2) Виявлення зон для майбутніх досліджень (Б17), які треба провести в проєкті, тому що попередні не змогли вирішити ідентифіковані проблеми. 3) Підготовка рекомендацій для реалізації подальших етапів проєкту (Б19).

Цією складовою завершується модель управління складним ІТ-проєктом з НДДКР в умовах невизначеності, а саме, процедура управління проєктом повертається до стандартного етапу «Розробка ІТ-продукту» (Б15), коли невизначеність зменшена, що забезпечило можливість прийняття прогнозованих проєктних рішень.

Результатом концептуального моделювання складного ІТ-проєкту з НДДКР є формалізована структура управління, що визначає ключові компоненти, їх взаємозв'язки та процеси реалізації. Це дозволяє оптимізувати ресурсне планування, розподілити ролі та зони відповідальності між учасниками проєкту, встановити механізми прийняття рішень і впровадити ефективні методи оцінки готовності до наступних етапів. Концептуальна модель допомагає зменшити невизначеність шляхом визначення точок ризиків складності та їх мінімізації через дослідження, формування та тестування прототипів, застосування Spikes, PoC, навчання команди та інших методів. Вона забезпечує логічну інтеграцію наукових досліджень у процес розробки, сприяє швидкому адаптуванню проєкту до змінних умов і дозволяє ефективно трансформувати результати НДДКР у робочі технічні рішення. Розроблена модель робить управління проєктом прогнозованим, гнучким і науково обґрунтованим, що є важливим для складних розробок зі складністю реалізації.

2.3. Модель управління гіпотезами в складних ІТ-проєктах з НДДКР в умовах невизначеності

В цьому підрозділі подано результати розробки моделі управління гіпотезами в складних ІТ-проєктах, які потребують виконання НДДКР. Вона є складовою логічної схеми управління складними ІТ-проєктами, яка була розроблена у 2.2. Модель базується на адаптивному підході, що поєднує науково-дослідні методи, експериментальну перевірку та використання систем штучного інтелекту для прийняття обґрунтованих проєктних рішень та враховує особливості ІТ-проєктів [27].

Проблема в управлінні складним ІТ-проєктом може виникати через високу невизначеність, складність архітектури, відсутність стандартних рішень, компетенцій та досвіду команди або потребу у впровадженні інноваційних технологій коли ефективність рішень не є наперед відомою. Наприклад, труднощі можуть полягати у виборі оптимальної технологічної архітектури, визначенні найефективнішої методології розробки або прогнозуванні ефективності нового рішення. Найбільш ефективним інструментом для цього етапу є Data Mining, оскільки дозволяє аналізувати великі обсяги даних, знаходити закономірності та ідентифікувати потенційні проблемні зони.

З урахуванням концептуальної моделі загального управління складними ІТ-проєктами (рисунок 2.2), структура моделі управління гіпотезами [27], пропонується в такому складі (рисунок 2.3).

1. Блок формування проблеми в управлінні ІТ-проєктом.

2. Блок визначення вхідних даних стосовно складності виконання проєкту, досвіду і компетенцій команди, наявності засобів розробки та обмежень по строкам, бюджету, кількості команди та її попереднього досвіду.

3. Блок визначення змінних та факторів впливу з технології управління ІТ-проєктом за методологіями PMBoK, SWEBOK, Scrum та ін.

4. Блок формування гіпотези щодо вирішення проблеми управління складним ІТ-проєктом.

5. Блок розробки плану перевірки гіпотези.

6. Блок розробки плану експериментів, прототипів рішень та тестування прототипів.

7. Аналіз отриманих результатів.

8. Блок перевірки рівня невизначеності.

9. Блок формування проєктного рішення (вирішення означеної проблеми управління ІТ-проєктом).

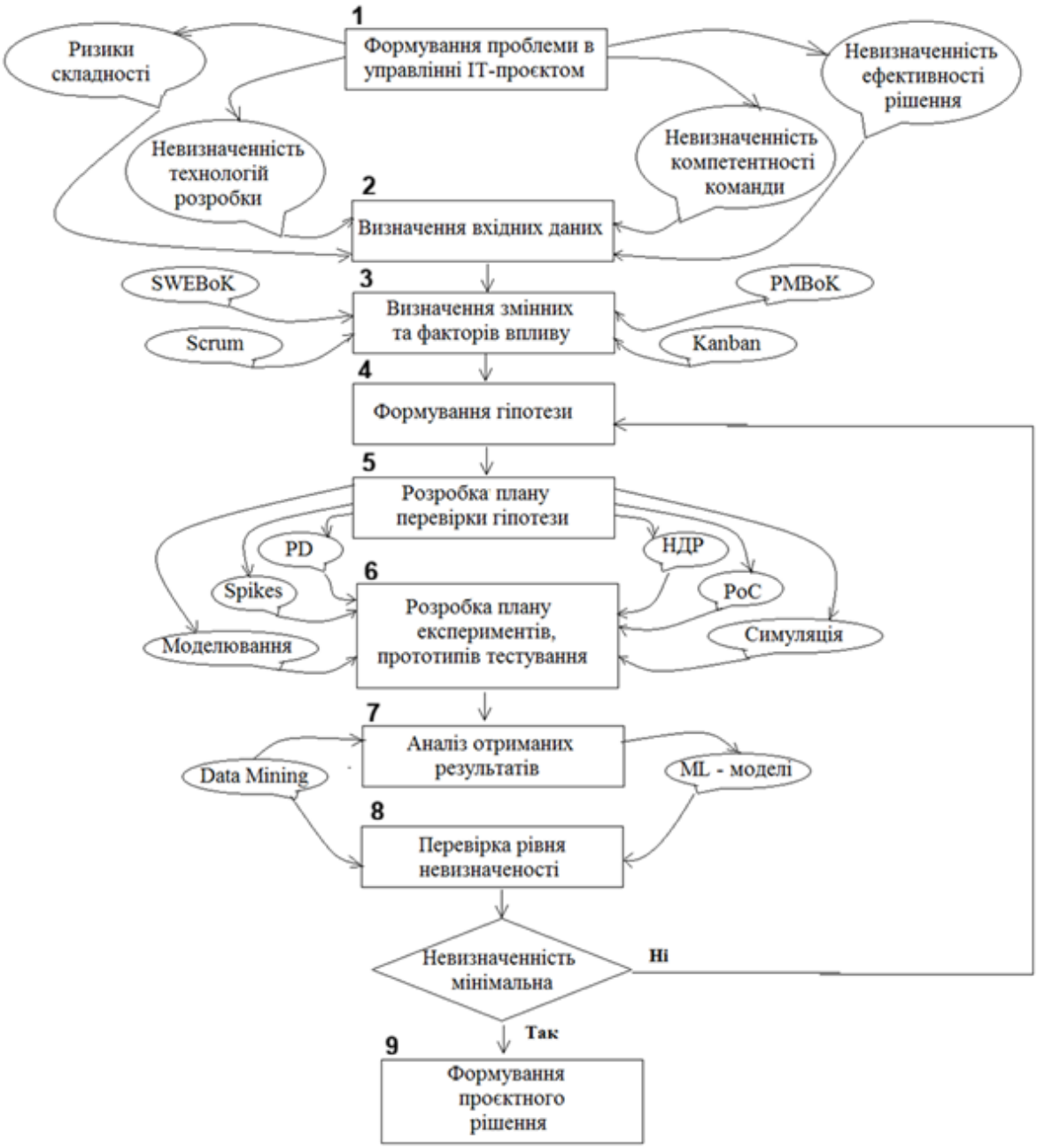


Рисунок 2.3 – Модель управління гіпотезами складними ІТ-проєктами з НДДКР. Джерело: розроблено автором

На рисунку 2.3 блоки моделі пронумеровані. Праворуч та ліворуч від них зображені: умови, за якими здійснюється перехід до наступного етапу; допоміжні інструменти; технології розробки. Зв’язки між блоками представлені двома типами: безпосередньо між блоками (2 – 5; 6 – 7) та через допоміжні інструменти (1 – 2; 5 – 6; 7 –8). Між блоками 8 – 9 зв'язок

організовано через перевірку умови про те, що невизначеність зменшилася та є можливість прийняте обґрунтоване проєктне рішення.

Складні IT-проєкти, як і будь-які проєкти, розпочинаються з етапу ініціації в результаті якого формується загальне бачення продукту проєкту та оцінка можливостей команди, щодо його реалізації. Не зважаючи на те, що за основу прийнята ітеративна технологія розробки, в моделі формуються проблеми з управління таким проєктом, які конкретизуються у ризики складності та невизначеності з: компетенції та навичок команди, технології розробки (їх відсутності, або відсутності досвіду у команди розробки), ефективності рішень (чи задовольнить результат роботи майбутнього продукту проєкту).

На основі цього у другому блоці формуються вхідні дані про складність проєкту, що включають оцінку рівня інноваційності, аналіз вимог, досвід команди та наявність необхідних ресурсів. Критичними факторами є компетенції фахівців, доступність технологічного стеку, часові та бюджетні обмеження. AI-моделі прогнозування, такі як Random Forest або Gradient Boosting, допомагають оцінити ризики, пов'язані з недостатньою компетентністю команди або можливими затримками у розробці.

Третій блок формує змінні і фактори впливу з методологій управління проєктами. Це управління ризиками, командою, стейкхолдерами, змістом, це рівень зміни вимог, ступінь невизначеності та адаптивність методології. Тому, елементами блоку є методології PMBoK, SWEBOK, Scrum, Kanban. Вони пропонують різні підходи до управління проєктними змінними. Для формування змінних та факторів впливу запропоновано використовувати алгоритми кластеризації, K-Means або DBSCAN, які дозволяють аналізувати вплив різних чинників на результати виконання проєкту.

Четвертим блоком моделі є формування гіпотези щодо вирішення проблеми управління IT-проєктом. Процедура передбачає встановлення припущення, яке можна перевірити експериментально. Наприклад, якщо застосувати новий підхід до балансування навантаження в мікросервісній

архітектурі, то швидкість обробки запитів може зрости на 20%. Використання Bayesian Networks дозволяє встановлювати причинно-наслідкові зв’язки між змінними та прогнозувати ймовірність підтвердження гіпотези.

Формулу гіпотези сформулюємо в такій редакції: «Якщо ми використовуємо X-підхід для управління проблемою Y, то досягнемо Z-результату з певною ймовірністю».

Після цього, в п’ятому блоці формується план перевірки гіпотези. Він включає вибір метрик оцінки ефективності, методів тестування та необхідних ресурсів. Елементами блоку є:

- проведення науково-дослідних робіт (НДР);
- моделювання;
- симуляція;
- Proof of concept;
- Spikes;
- PD, прототипування (Prototype Development).

Наприклад, для оцінки продуктивності можна використовувати A/B тестування або навантажувальне тестування. Використання Reinforcement Learning дозволяє знаходити оптимальні рішення через послідовні ітерації, оцінюючи відхилення від бажаного результату.

На основі цього, в шостому блоці плануються експерименти, розробка та тестування прототипів в межах створення мінімально життєздатного продукту (MVP). Використання симуляційних моделей, таких як Monte Carlo, допомагає прогнозувати результати експерименту та оцінювати ризики.

В сьомому блоці заплановано проведення аналізу отриманих результатів через порівняння емпіричних даних із запланованими (бажаними) показниками. Тут, як і в кожному блоці моделі, для підвищення значущості отриманих результатів будуть використані моделі та методи інтелектуального аналізу даних (Data mining) та системи штучного інтелекту (наприклад, моделі ML).

Отриманий результат — це результат перевірки гіпотези, яка формувалася на початку проєкту для зменшення рівня невизначеності. Тому, наступним етапом перевіряється, чи зменшили проведені дії рівень цієї невизначеності. Якщо «так», то реалізація проєкту продовжується за результатами проведених досліджень. Якщо «ні», то треба повернутися до блоку формування нової гіпотези.

Прийняття проєктного рішення залежить від отриманих експериментальних результатів. Якщо гіпотеза підтвердилася, можна впроваджувати рішення у процедуру управління проєктом, якщо ні — необхідно скоригувати підхід. Використання багатокритеріального аналізу рішень (AHP, TOPSIS) допоможе оцінювати альтернативні варіанти з урахуванням різних параметрів. Тому модель підтримує ітераційність та адаптивність, які забезпечують багаторазове тестування та покращення рішень на основі гнучких методів розробки.

Математична модель, що описуватиме управління гіпотезами в складних ІТ-проєктах з НДДКР має бути гібридною та об’єднувати елементи булевої логіки, байєсівських мереж, скінченних автоматів, алгоритмів машинного навчання та теорії прийняття рішень [28].

1. Вхідними даними для моделі є:

- P — сукупність проблем управління в проєкті;
- D — вихідні дані про складність проєкту (компетенції команди, технології та обмеження);
- V — набір змінних і факторів впливу з методологій управління (PMBoK, Scrum тощо);

2. Вихідними даними є:

- R — прийняте проєктне рішення щодо реалізації або зміни гіпотези
- U — зменшення рівня невизначеності (успішність перевірки гіпотези)

3. Формування гіпотези H залежить від проблеми P, даних D та факторів V:

$$H = f(P, D, V),$$

де H — гіпотеза;

P — проблема управління;

D — вихідні дані про складність;

V — змінні та фактори впливу.

4. Ймовірність істинності гіпотези H за даними E (експерименти) визначимо за формулою Байєса [29]:

$$P(H|E) = [P(E|H) * P(H)] / P(E),$$

де $P(H|E)$ — апостеріорна ймовірність гіпотези;

$P(E|H)$ — ймовірність спостереження E при істинності H;

$P(H)$ — апріорна ймовірність гіпотези;

$P(E)$ — ймовірність спостереження E.

3. Стан скінченного автомата на k-му кроці:

$$S\_\{k+1\} = \delta(S\_k, I\_k),$$

де S_k — поточний стан;

I_k — вхідні дані (результати перевірки гіпотези);

δ — функція переходу між станами.

4. Функція втрат (похибки) при оцінці гіпотези через ML-модель:

$$L(H) = \sum(\hat{y}_i - y_i)^2,$$

де $L(H)$ — функція втрат;

$\hat{y}_i$ — прогнозоване значення моделі;

y_i — фактичне значення за результатами експерименту.

5. Інтегральна оцінка гіпотези методом AHP (аналітичного ієрархічного процесу):

$$R = \sum(w_i * s_i),$$

де R — рейтинг гіпотези;

w_i — ваговий коефіцієнт критерію;

s_i — оцінка гіпотези за критерієм i.

6. Успішність гіпотези U як зниження невизначеності:

$$U = U_0 - \Delta U,$$

де U — поточний рівень невизначеності;

U_0 — початковий рівень;

ΔU — зміна рівня невизначеності внаслідок перевірки гіпотези.

7. Правило продовження або оновлення гіпотези:

$$R = H, \text{ якщо } P(H|E) \geq \theta; \text{ інакше } R = H',$$

де R — рішення щодо гіпотези;

θ — порогове значення ймовірності;

H' — оновлена або нова гіпотеза.

Застосування систем ШІ та ІАД забезпечують автоматизацію процесу перевірки гіпотез та скорочують час ухвалення рішень та мінімізують помилки через людський фактор. Таким чином, модель управління гіпотезами в складних ІТ-проєктах дозволяє ефективно вирішувати проблеми, пов’язані з високою невизначеністю, інтегруючи сучасні технології штучного інтелекту та інтелектуального аналізу даних у процес управління розробкою.

Висновки до розділу 2

У другому розділі були розроблені та формалізовані три ключові моделі, які є основою управління складними ІТ-проєктами в умовах невизначеності. Це: інформаційна модель складного ІТ-проєкту, концептуальна модель управління складним ІТ-проєктом, а також модель управління гіпотезами в складних ІТ-проєктах з НДДКР.

1. Інформаційна модель складного ІТ-проєкту з НДДКР забезпечує цілісне уявлення про структуру системи «складний ІТ-проєкт з НДДКР», взаємозв’язки між її компонентами, джерела невизначеності та логіку обробки

інформації. Особлива увага була приділена доменній інформаційній моделі, яка дозволяє детально описати специфіку предметної області, відобразити сутності, атрибути, бізнес-правила та потоки даних, що є критично важливим для проєктів з науково-дослідницькими та дослідно-конструкторськими компонентами.

Завдяки формалізації логічних взаємозв'язків у вигляді булевих формул і структур, модель дозволяє інтегрувати науково-дослідні завдання без порушення загальної логіки управління.

2. Концептуальна модель управління складними IT-проєктами узагальнює процеси управління з урахуванням таких елементів як управління ризиками складності, перевірка реалізуємості, PoC, Spikes, прототипування, навчання команди, рефакторинг. Це дозволяє адаптувати процес управління до змінних умов та результатів досліджень. Модель орієнтована на формування адаптивної системи управління, яка реагує на динаміку проєктного середовища. Особливу роль у концептуальній моделі відіграє процедура рефакторингу, яка дозволяє змінювати структуру управління без зміни кінцевої мети проєкту.

3. Модель управління гіпотезами є ключовим елементом адаптації в умовах високої невизначеності. Вона дозволяє формувати, перевіряти та обґрунтовувати проєктні рішення через гіпотези, що перевіряються на основі результатів досліджень, експериментів та моделей штучного інтелекту. Застосування Data Mining, ML, байєсівських мереж, скінченних автоматів і теорії прийняття рішень дозволить автоматизувати прийняття рішень на основі об'єктивних даних.

4. Розроблена гібридна математична модель управління гіпотезами забезпечує узгоджене функціонування усіх компонентів, визначення рівня невизначеності та обґрунтування переходу між етапами проєкту. Таким чином, усі три моделі утворюють узгоджену систему, яка дозволяє управляти складним IT-проєктом з високим ступенем невизначеності, забезпечуючи

інтеграцію результатів НДДКР у загальну логіку управління та формулювати ефективні проєктні рішення.

5. Результати досліджень другого розділу опубліковані в роботах [1, 6, 7, 26, 27].

Список використаних джерел у розділі 3

1. Путій Ілля, Бобровський Олексій. (2024). Інформаційна модель складного ІТ-проєкту з дослідницькою складовою. Project, Program, Portfolio Management (P3M-2024). Тези доповідей IX Міжнародної науково-практичної конференції Том 1. Одеса : ІШІР, 316-319

2. Istiadi, L. E. Nugroho and P. I. Santosa, (2014). An agent model of domain based information seeking on personal knowledge organizer. 2014 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia, pp. 259-263, DOI: 10.1109/ICITSI.2014.7048274.

3. Proper, H.A., Guizzardi, G. (2024). Understanding the Variety of Domain Models: Views, Programs, Animations, and Other Models. SN COMPUT. SCI. 5, 861 https://doi.org/10.1007/s42979-024-03163-y

4. Mani, N., Helfert, M., Pahl, C., Nimmagadda, S.L., Vasant, P. (2018). Domain Model Definition for Domain-Specific Rule Generation Using Variability Model. In: Vasant, P., Litvinchev, I., Marmolejo-Saucedo, J. (eds) Modeling, Simulation, and Optimization . EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-319-70542-2_4

5. Georg Heidenreich, Pantelis Angelidis. (2010) An Approach towards Semantic Interoperability using Domain Models. Conference: Electronic Journal of Health Informatics. Volume: 5. URL : https://www.researchgate.net/publication/274954770_An_Approach_towards_Semantic_Interoperability_using_Domain_Models

6. Путій І.Д., Бондар О.А. (2024). Складність ІТ-проєктів з науково-дослідною та дослідно-конструкторською розробкою. Міжнародна науково-

практична конференція «Інформаційні системи в управлінні проєктами та програмами», Коблево, ХНУРЕ, 197 –200.

7. Путій І. Д., Тесленко П. О. (2024). Аналіз сучасних підходів до управління складними ІТ-проєктами. Управління розвитком складних систем, 59, 81 – 89. URL : https://urss.knuba.edu.ua/zbirnyk-59. ISSN 2219-5300

8. Євтух П. Побудова моделей сенсорних мереж та їх оцінювання методами теорії графів / Євтух П., Карпінський В., Кінах Я. // Вісник ТНТУ. — 2010. — Том 15. — № 4. — С.146-154

9. Теорія систем масового обслуговування : навч. посібник / А. Л. Литвинов ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 141 с. ISBN 978-966-695-473-5

10. K. Kolesnikova, T. Olekh, D. Lukianov, D.J. Obenewaa, Markov Principles of Project Effectiveness. SIST 2021 - 2021 IEEE International Conference on Smart Information Systems and Technologies, 2021, 9465881.

11. Огірко, І. В., Ясінський, М. Ф., & Ясінська-Дамрі, Л. М. (2015). Жорсткі і м’які математичні моделі та їх застосування. Наукові записки [Української академії друкарства]. Серія: Технічні науки, (1), 102-117.

12. Теорія інформації і кодування: курс лекцій [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра за спеціальністю 124 «Системний аналіз» / КПІ ім. Ігоря Сікорського ; уклад.: А.Є.Коваленко. 247. https://ela.kpi.ua/bitstreams/56dac98e-5c52-439c-a0ba-8acc3919af5b/download

13. Супруненко, О. О., & Онищенко, Б. О. (2022). Аналіз прихованих помилок у моделях програмних систем на основі мереж Петрі. Electronic Modeling, 44(2), 38-50.

14. Ракитянська, Г. Б., and Б. В. Прус. Моделювання надійності програмного забезпечення на основі алгебри алгоритмів і нечіткої логіки. Diss. Одеський національний технологічний університет, 2023.75-78

15. Demianenko, V. (2019). Онтологічні засади формалізації інформаційних джерел у е-освітніх середовищах. ScienceRise: Pedagogical Education, (6 (33)), 39-45.

16. Кульчицький І.М., Осідач Н.Б. (2011) Концептуальна модель бази даних збірки М. Номиса “Українські приказки, прислів’я і таке інше” URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/jun/3408/ 2858.pdf? utm_source=chatgpt.com

17. Бабаєв, В. М., & Фесенко, Т. Г. (2010). Концептуальна модель організації офісу управління будівельними проєктами в перспективі проєктного менеджменту. Восточно-европейский журнал передовых технологий, 1(3 (43)), 9-11.

18. Євдокимова, А. В. (2011). Концептуальна модель активності громад для задач формування портфелю грантових проєктів соціально-економічного розвитку громад, які підтримуються міжнародними організаціями. Управління проєктами та розвиток виробництва, (4), 134-139.

19. Веренич, О. В. (2015). Концептуальна модель формування ментального простору. Управління розвитком складних систем, 23 (1), 39-43.

20. Бакуліч, О. О., & Севост’янова, А. В. (2019). Концептуальна модель балансу ризиків (можливостей та загроз) стейкхолдерів проєктів вітроенергетики. Вчені записки Університету «КРОК», (3) 55), 143-150.

21. Бушуєв, С., Бушуєв, Д., Бушуєва, В., & Бушуєва, Н. (2021). Концептуальна модель цифрового сліду проєктів в умовах цифровізації суспільства. Управління розвитком складних систем, (46), 12-18.

22. Близнюкова І.О., Данченко О.Б., Тесленко П.О., Заруцький С.О. (2021). Концептуальна модель креативного управління командою ІТ проєкту. Тези доповідей VI Міжнародної науково-практичної конференції Р3М-2021, Одеса: ІШІР, 81 – 83.

23. Мартін Роберт (2019). Чистий код: створення, аналіз, рефакторинг. Харків.: Фабула, 416. ISBN 978-617-09-5285-1

24. Струзік В.А., Грибков С.В., Чобану В.В. (2020). Категорія рефакторинг доступу. Наукові праці НУХТ Том 26, 2, 31 – 49. DOI: 10.24263/2225-2924-2020-26-2-4

25. Кривоножко, Г. Є., Голікова, О. В., & Заікіна, Т. В. (2020). Значення рефакторінгу під час виконання судових експертиз (експертних досліджень) та проведення розрахунків на основі моделі COCOMO II. Експерт: парадигми юридичних наук і державного управління, 3 (9), 84-90. https://doi.org/10.32689/2617-9660-2020-3(9)-84-90

26. Путій, І. Д., & Тесленко, П. О. (2025). Концептуальна модель складного ІТ-проєкту. Таврійський науковий вісник. Серія: Технічні науки, (2), 148-154. DOI: https://doi.org/10.32782/tnv-tech.2025.2.16

27. Путій І. Д., Тесленко П. О. (2025). «Канвас» методу формування гіпотез для складного ІТ-проєкту. Управління розвитком складних систем, 64, 120–127. DOI: 10.32347/2412-9933.2025.64.

28. Коваленко А. А., Кучук Г. А. (2018). Методи синтезу інформаційної та технічної структур системи управління об’єктом критичного застосування. Сучасні інформаційні системи, 2, (1), 22–27.

29. Кашкевич, С. О., Єфименко, О. В., Троцько, О. О., Гаман, О. В., & Шишацький, А. В. (2024). Огляд методів штучного інтелекту в інтересах вирішення завдань самоорганізаці. In The 13th International scientific and practical conference “Information and innovative technologies in the development of society”(April 02–05, 2024) Athens, Greece. International Science Group. 283-289.

РОЗДІЛ 3. МЕТОДИ УПРАВЛІННЯ СКЛАДНИМИ ІТ-ПРОЄКТАМИ В УМОВАХ НЕВИЗНАЧЕНОСТІ

3.1. Канвас методу формування гіпотез для складного ІТ-проєкту

Канвас був запропонований О. Остервальдером у моделі Business Model Canvas [1] для менеджменту, де необхідно забезпечити чітке бачення ключових компонентів складних систем [2].

В дисертації канвас розглядається як підхід для впорядкування процесу створення та перевірки гіпотез у складних ІТ-проєктах. Він дозволяє трансформувати припущення у структуровані елементи, які можна тестувати, вимірювати та використовувати в управлінні [2].

Гіпотеза у класичному науковому розумінні є припущенням, яке потребує перевірки шляхом дослідження або експерименту. В управлінні складними ІТ-проєктами гіпотези набувають особливої ваги, адже дозволяють перевіряти реалізованість тих чи інших технічних рішень ще до того, як будуть витрачені значні ресурси на їх впровадження. Використання гіпотез в управлінні складними ІТ-проєктами має знизити невизначеність, оптимізувати витрати та знизити проєктні ризики, що пов’язані з його реалізацією. Мова йде не про маркетингові або бізнесові припущення щодо поведінки стейкхолдерів, а саме про технічні гіпотези, які спрямовані на перевірку архітектурних, інженерних та процесних рішень у розробці складних ІТ-продуктів. В рамках дисертаційного дослідження було запропоноване таке визначення технічної гіпотези.

Визначення 3.1. ***Технічна гіпотеза для управління складними ІТ-проєктами*** визначається як формалізоване припущення про працездатність, ефективність або доцільність конкретного технічного, алгоритмічного, програмного рішення, яке можна перевірити за допомогою експерименту, прототипування або обмеженого впровадження [2].

Важливим є те, що йдеться не просто про метод формування гіпотез, а саме про ***канвас*****,** оскільки він дозволяє систематизувати усі ключові елементи:

– контекст;
– проблему;
– передумови;
– критерії успішності;
– методи перевірки;
– очікувані результати.

Канвас виступає як інтерактивний шаблон, який уніфікує підхід до створення та перевірки гіпотез, зменшує ризик втрати важливої інформації та підвищує узгодженість між членами команди. Отже, під ***канвасом методу формування гіпотез у складних ІТ-проєктах*** слід розуміти структуровану сукупність, що дозволяє перетворювати технічні припущення у перевірені формулювання з чітко визначеними вхідними даними, умовами тестування та критеріями успіху [2].

Принципи побудови канвансу О. Остервальдера [1] було адаптовано наступним чином [2]:

1) Customer Segments → Команда + стейкхолдери проблеми, це ті, хто формує проблему і генерує гіпотези, а також «власників вимог» (business analysts, product owner).

2) Value Proposition → Цінність гіпотези. Головне питання: «Навіщо цю гіпотезу формуємо? Яку нову користь вона дасть ІТ-проєкту?»

3) Channels → Канали доставки гіпотези. Як гіпотеза знижує ризик чи наближає завершення ІТ-проєкту. Це фактично «аргумент» для її існування.

4) Customer Relationships → Відносини між групами. Тут варто говорити про координацію між тими, хто формує проблему, і тими, хто перевірятиме гіпотезу. Це — «взаємодія між ролями» команди складного ІТ-проєкту.

5) Revenue Streams → Результати формування гіпотези. *Expected Results*. Тобто який «артефакт» буде на виході (описана гіпотеза у певному форматі). Це не гроші, а «вихідна цінність».

6) Key Resources → Ресурси. це дані, люди, знання, середовища, які потрібні для формування гіпотези.

7) Key Activities → Дії. Аналіз проблеми, формулювання, документування (далі тут буде сформовано алгоритм формування гіпотез).

8) Key Partnerships → Зовнішні учасники. Іноді справді потрібні зовнішні консультанти, інші команди чи навіть вендори (окрім «внутрішньої команди складного ІТ-проєкту). Якщо таких немає — просто блок порожній.

9) Cost Structure → Витрати. Це час, людино-години, ресурси для воркшопів чи досліджень. Блок показує, що формування гіпотез теж коштує.

Графічне зображення канвансу подано на рисунку 3.1.

Рисунок 3.1 – Канвас методу формування гіпотез для складного ІТ-проєкту. Джерело: розроблено автором

Таким чином, фокус технічних гіпотезах в ІТ-проєктах переміщено на перевірку життєздатності рішень у процесі розробки (архітектури, технологій, компетенцій команди). Це відповідає технології Spikes у Agile/Scrum, але через Canvas додано більшої формалізації [3, 4].

Сформулюємо універсальний шаблон для формування гіпотез «Hypothesis Canvas 2.0 для ІТ-проєктів» (формування):

1. Проблемна зона (Problem Area)

Що саме викликає сумнів чи невизначеність?

Вхідні дані: вимоги, архітектурні обмеження, user stories, backlog. → Формулювання: «Неясно, чи можливо …» / «Немає підтвердження, що …».

2. Технічна гіпотеза (Hypothesis Statement)

Яке припущення (проблему) ми висуваємо для перевірки?

Форма: «Якщо ми застосуємо [технологія/метод], то отримаємо [результат/ефект] в [контексті]».

3. Критичність (Amplitude / Criticality)

Наскільки важлива ця проблема для успіху проєкту?

Шкала: низька (можна обійтись без формування гіпотези), середня (є ризики, але ця проблема не блокує проєкт), висока (без формування гіпотези та подальшій її перевірки ІТ-проєкт не реалізується).

4. План формування (Formation plan)

Яким способом ми формуватимемо гіпотезу?

Варіанти: аналіз проблем і вимог, аналіз очікувань стейкхолдерів, порівняння User story, gap analysis, аналіз сумісності, review архітектури.

5. Вхідні ресурси (Inputs / Resources)

Що потрібно для формування гіпотези?

Люди (компетенції команди), середовище (dev tools, knowledge base), дані (історичні рішення, попередні проєкти), час.

6. Критерії успіху формування (Formation Success Criteria)

Що вважатиметься підтвердженням того, що гіпотеза сформульована коректно?

Приклади: сформульовано у валідуємій формі «якщо… то…», однозначно описана невизначеність, визначено параметри критичності, гіпотеза прийнята командою до подальшої перевірки.

7. Очікуваний вихід (Expected Outcome)

Який результат ми прогнозуємо від етапу формування?

Форма: «Ми отримали гіпотезу, яка чітко описує проблему/невизначеність і може бути передана на перевірку через HD-метод або spike-експеримент».

8. Наслідки (Implications)

Що робимо далі залежно від якості сформованої гіпотези?

– Якщо гіпотеза відповідає критеріям успіху → передаємо її у backlog на етап перевірки.

– Якщо ні → уточнюємо формулювання / створюємо додаткову гіпотезу.

Ключова відмінність від бізнесових Hypothesis Canvas у тому, що в них «цінність» розуміється не як для стейкхолдера/ринку, а як успішність реалізації технічного рішення [5].

Саме для цього запропоновано блок Критичності (Amplitude), щоб відсіювати гіпотези, які не потребують глибоких перевірок.

Канвас промарковано як «2.0», щоб показати, що це модифікований варіант, адаптований саме під контекст складних ІТ-проєкт сформульованих в предметній області.

В класичній літературі та практиці існує лише Hypothesis Canvas (який умовно можна вважати *1.0*), який використовують у Lean Startup, маркетингу та UX-дослідженнях [6]. Він виглядає наступним чином: формулювання гіпотези, припущення про цінність для користувача, метод перевірки, метрики успіху. Це більше про бізнес та поведінку клієнтів.

Запропонований «Hypothesis Canvas 2.0» це розширена версія, де замість бізнес-цінності ми працюємо з технічною життєздатністю, додано блок «Критичність (Amplitude)», щоб одразу відсікати неважливі гіпотези. Запропоновано «План перевірки» для з'ясування її фізичної реалізуємості.

Тобто 1.0 → бізнес і ринок, а 2.0 → технічна верифікація в ІТ-проєктах. Цей «2.0» не є офіційним терміном, це скоріше авторське позначення версії, щоб підкреслити відмінність і не плутати з класичним Lean Canvas чи Hypothesis Canvas у маркетингу [2]. Алгоритм «генерації гіпотез у складних ІТ-проєктах створено як послідовність керованих функцій (КФ), що відповідатиме п.4 «Hypothesis Canvas 2.0 для ІТ-проєктів», який має назву «План формування». Кожна КФ — це структурована операція, яку можна

застосовувати багаторазово на різних етапах ІТ-проєкту. Такий підхід відповідає ідеї гіпотетико-дедуктивного методу, але обмежується стадією формування гіпотез [7].

Алгоритм «Генерація гіпотез у складних ІТ-проєктах» поданий на рисунку 3.2.

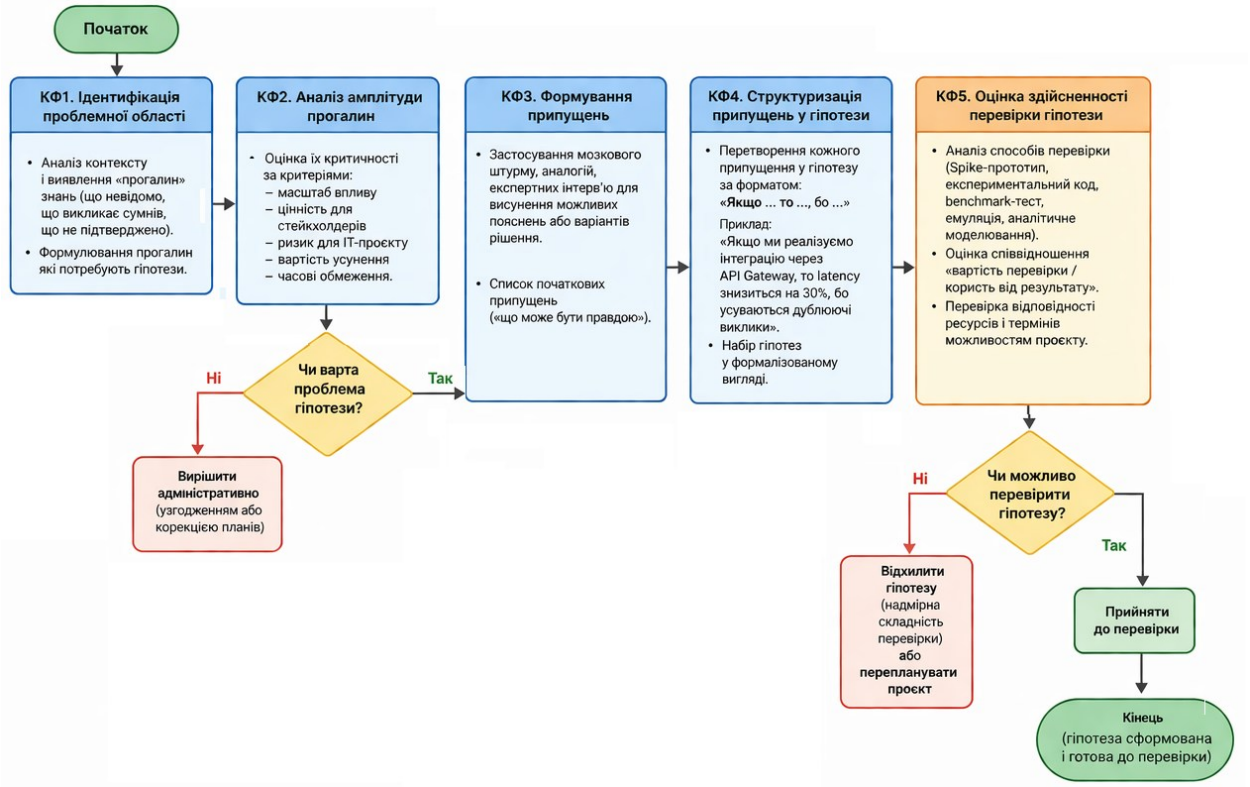


Рисунок 3.2 – Алгоритм формування гіпотез. Джерело: розроблено автором

КФ1. Ідентифікація проблемної області.

Вхідні дані: вимоги замовника (якщо є), відгуки стейкхолдерів, бізнес-цілі, невирішені питання у проєктній документації. Джерела — бізнес-аналітика, опитування, інтерв'ю, backlog.

Проєктні дії: аналіз контексту і виявлення «прогалин» знань (що невідомо, що викликає сумнів, що не підтверджено).

Вихідні дані: формулювання прогалин, які потребують гіпотези.

КФ2. Аналіз амплітуди прогалин.

Вхідні дані: список прогалин (з КФ1).

Проєктні дії: оцінка їх критичності за критеріями (масштаб впливу, цінність для стейкхолдерів, ризик для IT-проєктах, вартість усунення, часові обмеження).

Вихідні дані: рішення — які прогалини потребують перетворення у гіпотези, а які можна вирішити адміністративно (наприклад, узгодженням або корекцією планів).

КФ3. Формування припущень.

Вхідні дані: проблемне питання (з КФ2), знання/досвід команди IT-проєкту, результати попередніх досліджень/проєктів, best practices (PMBOK, ITIL, SAFe, ISO/IEC 12207).

Проєктні дії: застосування мозкового штурму, аналогій, експертних інтерв'ю для висунення можливих пояснень або варіантів рішення.

Вихідні дані: список початкових припущень («що може бути правдою»).

КФ4. Структуризація припущень у гіпотези.

Вхідні дані: список припущень (з КФ3).

Проєктні дії: перетворення кожного припущення у гіпотезу за форматом Якщо … то …, бо … (наприклад: Якщо ми реалізуємо інтеграцію через API Gateway, то latency знизиться на 30%, бо усуваються дублюючі виклики).

Вихідні дані: набір гіпотез у формалізованому вигляді.

КФ5. Оцінка здійсненності перевірки гіпотези.

Вхідні дані: сформовані гіпотези (з КФ4), доступні ресурси проєкту (час, бюджет, компетенції, середовище).

Проєктні дії: аналіз способів, якими можна було б перевірити гіпотезу (наприклад, через spike-прототип, експериментальний код, benchmark-тест, емуляцію, аналітичне моделювання). Оцінка співвідношення «вартість перевірки / користь від результату». Якщо перевірка потребує ресурсів, що перевищують цінність проєкту, або термінів, несумісних із його реалізацією, гіпотеза відхиляється чи проєкт перепланується так, щоб уникнути цієї проблеми.

Вихідні дані: рішення щодо подальшої долі гіпотези — «прийнята до перевірки» / «відхилена через надмірну складність перевірки».

Таким чином, алгоритм матиме дві перевірки на доцільність гіпотез: КФ2 → фільтр за цінністю проблеми («чи варта вона гіпотези?»), КФ5 → фільтр за здійсненністю перевірки («чи можемо ми взагалі її перевірити?»).

Вигляд фінальної гіпотези [2].

Гіпотеза оформлюється у вигляді структурованого артефакту, готового до подальшої перевірки:

– Формулювання**:** *Якщо ... то ..., бо ...*
– Обґрунтування: які прогалини знань чи проблеми вона закриває.
– Критичність: низька / середня / висока.
– Статус: сформована (очікує перевірки).
– Місце у backlog: де саме вона впливає на планування проєкту.

Подальша робота з гіпотезою полягає у переході до фази перевірки наприклад, через Spike або експеримент, що було обґрунтовано у КФ5, однак це вже наступний процес управління, що виходить за межі даного алгоритму.

3.2 Двоетапний метод формування гіпотез в складних IT-проєктах

Спираючись на системну постановку задачі, розробимо алгоритм методу формування гіпотез для складних IT-проєктів, який охоплює повний цикл — від збору сигналів (баги, відгуки, побажання, backlog) до створення структурованого переліку гіпотез, що пройшли двоетапний механізм фільтрації (цінність → здійсненність) і готові для перевірки командою розробників. Беручі за основу «Hypothesis Canvas 2.0 для IT-проєктів» (підрозділ 3.1) [2], метод формування гіпотез для складних IT-проєктів з урахуванням двоетапного механізму фільтрації представлено у форматі swimlane diagram для 5 стовпців. Метод подано на рисунку 3.3.

Вхідними даними для методу будуть записи у backlog, user stories, тестові звіти, баг-репорти, фідбек користувачів, протоколи нарад, коментарі стейкхолдерів, технічні ризики, архітектурні обмеження.

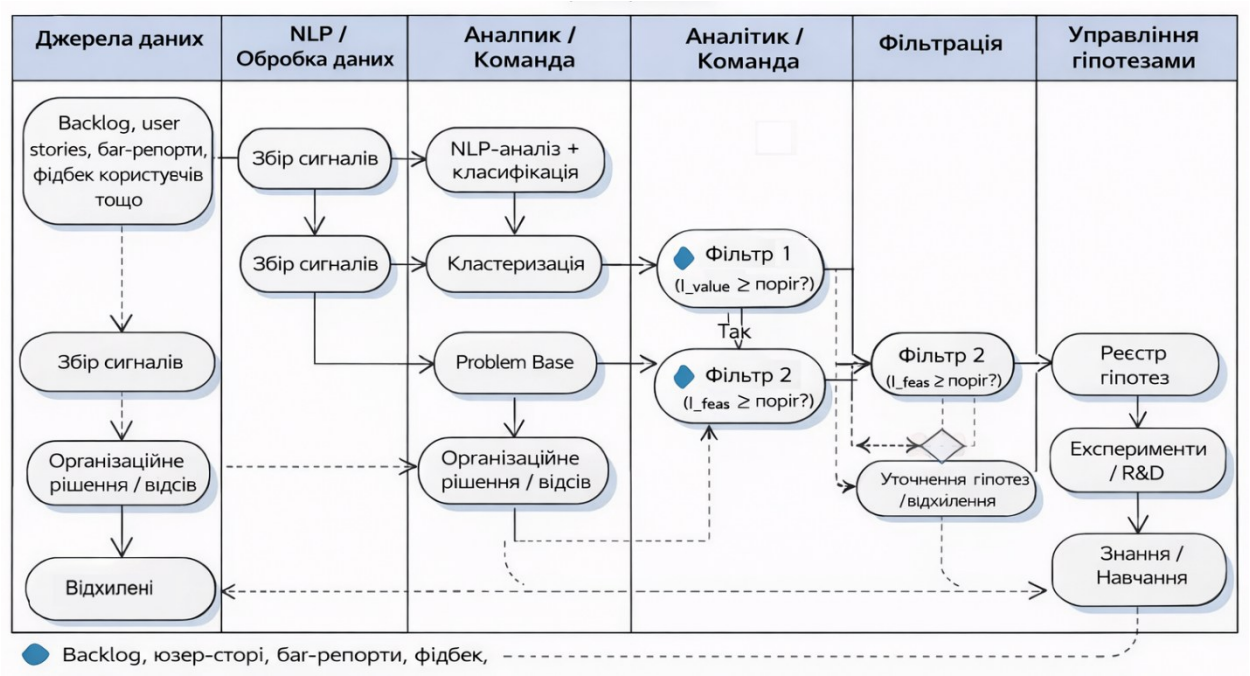


Рисунок 3.3 – Двоетапний метод формування гіпотез в складних IT-проєктах. Джерело: розроблено автором

Вихідними даними — структурований, погоджений перелік гіпотез, готових до перевірки командою R&D.

Етап 1. Збір первинних сигналів.

1. Збір усіх невизначеностей, проблем, помилок, зауважень, запитів, пропозицій та спостережень із різних джерел (Jira, Service Desk, Confluence, Slack, UX-форумів, опитувань).

2. Використання інструментів NLP-аналізу текстів для автоматичного вилучення змістових одиниць: невизначеність, проблема, ризик, побажання.

3. Попередня класифікація сигналів за типами: *функціональна проблема, продуктивність, UX, інтеграція, безпека, архітектура, бізнес-процес*.

4. Формування бази первинних проблем (Problem Base).

Приклад коду, який демонструє прототип такого рішення поданий в лістингу 3.1. Використано модель sentence-transformers для семантичного аналізу і просту класифікацію за подібністю до наперед визначених категорій.

Лістинг 3.1 – Код семантичного аналізу та класифікацію за визначених категорій

```
# Етап 1: Формування бази первинних проблем (Problem Base)
# для проєкту «SmartRetail 360»
from sentence_transformers import SentenceTransformer, util
import pandas as pd
# 1. Ініціалізація моделі семантичного аналізу
model = SentenceTransformer('all-MiniLM-L6-v2')
# 2. Попередньо визначені типи проблем
problem_types = [
    «функціональна проблема»,
    «продуктивність»,
    «UX»,
    «інтеграція»,
    «безпека»,
    «архітектура»,
    «бізнес-процес»
]
# 3. Приклад сирих сигналів (імітація Jira / Slack / Service Desk)
raw_signals = [
    «Під час великого навантаження частина подій Kafka губиться.»,
    «Користувачі скаржаться, що сторінки з рекомендаціями завантажуються надто довго.»,
    «Дані про замовлення не синхронізуються з SAP у реальному часі.»,
    «Інтерфейс для мобільних користувачів має проблеми з масштабуванням.»,
    «Немає чіткої політики зберігання персональних даних для країн ЄС.»,
    «Після оновлення з'явилися циклічні запити між сервісами, що перевантажують API.»,
    «Аналітики просять додати новий сценарій звіту про продажі.»
]
# 4. Отримання векторів для типів і сигналів
type_embeddings = model.encode(problem_types, convert_to_tensor=True)
signal_embeddings = model.encode(raw_signals, convert_to_tensor=True)
# 5. Класифікація сигналів за найвищою семантичною схожістю
classified = []
for i, signal in enumerate(raw_signals):
    cosine_scores = util.cos_sim(signal_embeddings[i], type_embeddings)
    best_idx = int(cosine_scores.argmax())
    problem_type = problem_types[best_idx]
    classified.append({
        «ID»: f»P{i+1:03d}»,
        «Сигнал / Проблема»: signal,
        «Тип проблеми»: problem_type,
        «Оцінка схожості»: float(cosine_scores[0][best_idx])
    })
# 6. Формування таблиці Problem Base
problem_base = pd.DataFrame(classified)
print(problem_base)
# 7. Збереження результату у CSV
```

problem_base.to_csv(«Problem_Base_SmartRetail360.csv», index=False, encoding=«utf-8-sig»)

print(«\n ✅ Файл Problem_Base_SmartRetail360.csv створено успішно.»)

Етап 2. Нормалізація та агрегація.

1. Виконання NLP-дедуплікації: виявлення дублюючих або семантично подібних проблем.

2. Групування проблем за близькістю контексту (кластеризація методами K-Means, UMAP).

3. Формування узагальненого переліку проблемних зон невизначеності, кожна з яких може стати джерелом потенційної гіпотези.

Приклад коду, який демонструє прототип такого рішення поданий в лістингу 3.2.

Лістинг 3.2 – Код для формування узагальненого переліку проблемних зон невизначеності

```
# Eman 2: Нормалізація та агрегація невизначеностей (Problem Base)
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
import matplotlib.pyplot as plt
# 1. Вхідні дані — проблеми з Emany 1
problem_base = [
    «Під час великого навантаження частина подій Kafka губиться.»,
    «Kafka не справляється з обробкою подій при понад 10 тис. запитів.»,
    «Дані про замовлення не синхронізуються з SAP у реальному часі.»,
    «SAP API викликає помилки при масових оновленнях.»,
    «Користувачі скаржаться, що сторінки з рекомендаціями завантажуються надто довго.»,
    «Інтерфейс для мобільних користувачів має проблеми з масштабуванням.»,
    «UI не адаптується для малих екранів.»,
    «Немає чіткої політики зберігання персональних даних для країн ЄС.»,
    «Відсутня система логування доступу до персональних даних.»,
    «Після оновлення з'явилися циклічні запити між сервісами, що перевантажують API.»
]
# 2. Векторизація текстів методом TF-IDF
vectorizer = TfidfVectorizer()
embeddings = vectorizer.fit_transform(problem_base)

# 3. Кластеризація схожих проблем (NLP-дедуплікація)
```

```
num_clusters = 4   # кількість узагальнених проблемних зон
kmeans = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)
kmeans.fit(embeddings)
# 4. Визначення представників кластерів
closest, _ = pairwise_distances_argmin_min(kmeans.cluster_centers_, embeddings)
generalized_zones = []
for cluster_idx in range(num_clusters):
    indices = [i for i, label in enumerate(kmeans.labels_) if label == cluster_idx]
    examples = [problem_base[i] for i in indices]
    representative = problem_base[closest[cluster_idx]]
    generalized_zones.append({
        «Кластер»: f»Зона {cluster_idx + 1}»,
        «Представницька проблема»: representative,
        «Кількість схожих проблем»: len(indices),
        «Приклади»: «; «.join(examples)
    })
# 5. Формування та збереження таблиці Problem Zones
zones_df = pd.DataFrame(generalized_zones)
zones_df.to_csv(«Problem_Zones_SmartRetail360.csv», index=False, encoding=«utf-8-sig»)
# 6. Візуалізація кластерів
plt.figure(figsize=(8, 5))
plt.scatter(range(len(problem_base)), kmeans.labels_, c=kmeans.labels_, cmap='viridis')
plt.title(«Кластеризація проблем SmartRetail 360 (Eman 2)»)
plt.xlabel(«Номер проблеми»)
plt.ylabel(«Номер кластера»)
plt.grid(True)
plt.show()
# 7. Вивід результатів
print(«=== Узагальнені проблемні зони SmartRetail 360 ===\n»)
print(zones_df)
print(«\n ✅ Дані збережено у файлі: Problem_Zones_SmartRetail360.csv»)
```

Етап 3. Первинне фільтрування (Фільтр 1 — цінність проблеми).

1. Для кожної проблеми визначається критичність (amplitude) за параметрами:

- вплив на цілі проєкту;
- масштаб наслідків (локальний / системний);
- ризик для якості / строків / вартості;
- кількість стейкхолдерів, яких зачіпає проблема.

2. Застосовується модель багатокритеріального аналізу (MCDA або AHP) для обчислення інтегрального індексу важливості I_value.

3. Проблеми, у яких I_value < порогу, відсіюються як такі, що не потребують формування гіпотез (можуть бути вирішені організаційно).

4. Решта проблем переходять у пул “кандидатів у гіпотези”.

Етап 4. Формування початкових гіпотез.

1. Для кожного відібраного кандидата формулюється попереднє припущення у форматі:

«Якщо ми застосуємо [рішення / технологію / підхід], то отримаємо [результат / ефект], бо [причина / обґрунтування]».

2. Використовуються методи генерації ідей:

– експертні сесії;
– мозкові штурми;
– аналіз аналогій;
– звернення до бази попередніх гіпотез.

3. Результатом є список попередніх гіпотез, пов’язаних із відповідними проблемами.

Етап 5. Оцінка здійсненності (Фільтр 2 — feasibility check).

1. Для кожної гіпотези виконується аналіз здійсненності перевірки — чи є можливість реально протестувати припущення у межах проєкту.

2. Оцінюються параметри:

– доступні ресурси (час, бюджет, люди, середовище, навички, досвід);
– складність експерименту;
– залежності від зовнішніх систем;
– очікувана вартість перевірки.

3. Застосовується модель прогнозування вартості та тривалості (наприклад XGBoost або Random Forest) для обчислення індексу здійсненності *I_feas.*

4. Якщо *I_feas < порогу*, гіпотеза маркується як така, що не може бути перевіреною, або надто витратна → повертається на уточнення або відхиляється.

5. Якщо *I_feas* ≥ *порогу*, гіпотеза переходить у перелік “допущених до перевірки”.

Етап 6. Узгодження та нормалізація гіпотез.

1. Виконується рев’ю гіпотез командою: архітектор, бізнес-аналітик, менеджер, QA, розробник.

2. Перевіряється однозначність формулювання, наявність посилань на вимоги, тестові критерії, критерії успіху.

3. Застосовується онтологічне зіставлення (ontology alignment) для уніфікації термінології та виключення семантичних колізій.

Етап 7. Формування фінального переліку гіпотез.

1. Усі гіпотези, що пройшли двоетапну фільтрацію (цінність + здійсненність), заносяться у реєстр гіпотез (Hypothesis Register).

2. Для кожної гіпотези вказується:

- формулювання у форматі *“Якщо ... то ... бо ...”*;
- обґрунтування (яку проблему вирішує);
- рівень критичності (низький / середній / високий);
- відповідальний власник;
- статус: сформована / очікує перевірки / у backlog;
- посилання на артефакти (issue ID, user story, документацію).

3. Реєстр передається команді R&D для проведення експериментів (spike, PoC, MVP).

Етап 8. Зворотний зв’язок і навчання

1. Після перевірки гіпотез результати (успішність / відхилення / нові проблеми) зберігаються у базі знань.

2. Алгоритм самонавчається — на основі даних минулих гіпотез уточнюються порогові значення *I_value* і *I_feas*.

3. Формується замкнений цикл безперервного вдосконалення (continuous hypothesis management).

3.3 Рефакторинг-метод управління складними ІТ-проєктами

Задля підвищення ефективності управління складними ІТ-проєктами в роботі запропоновано концепцію «рефакторинг-методу управління складними ІТ-проєктами» [8]. Термін «рефакторинг-метод» є авторським, що логічно походить від класичного поняття рефакторингу у програмній інженерії [9], але адаптований для управлінських процесів.

В науковій літературі відсутнє пряме визначення такого підходу, однак його зміст наближається до ідей безперервного вдосконалення процесів (continuous improvement) і адаптивного управління знаннями в ІТ-проєктах.

Рефакторинг в класичному розумінні — це процес поліпшення внутрішньої структури програмного коду без зміни його зовнішньої поведінки, перевіряючи це постійним **регресійним тестуванням** [9, 10]. Його мета — підвищення якості, читабельності, можливості підтримки та продуктивності системи. У сфері інформаційних технологій рефакторинг є складовою життєвого циклу розробки програмного забезпечення, тісно пов’язаною з Agile та DevOps-практиками.

3.3.1 Розробка процедури рефакторинг-методу управління ІТ-проєктами

В дисертації запропоновано застосовувати рефакторинг не до коду, а до процесів управління складними ІТ-проєктами (підрозділ 2.2). Сенс такого підходу полягає у безперервному виявленні та усуненні проблем в управлінні — таких, як неузгодженість вимог, дублювання завдань, інформаційний «вакуум» між командами або неефективні комунікаційні ланцюги, що пов’язані з проєктною невизначеністю [11]. Зміни вносяться до процесів, ролей, потоків комунікацій і навіть логіки прийняття рішень, але без зміни:

- мети проєкту;
- цільових показників (KPI, OKR);
- обсягу робіт;
- стратегічних бізнес-вимог.

«***Рефакторинг управління IT-проєктами***» в дисертації визначається як процес системного удосконалення управлінських структур, комунікацій і процесів у межах IT-проєкту, який спрямований на підвищення його ефективності, узгодженості та адаптивності через зменшення невизначеності.

Тобто мова йде про застосував принципу «safe refactoring» з програмної інженерії [12] до рефакторингу процесів управління IT-проєктами, а саме, до керованого та контрольованого рефакторингу управління.

Процедура рефакторингу в управлінні IT-проєктом — це системний процес удосконалення управлінських механізмів, спрямований на підвищення ефективності взаємодії команд, якості прийняття рішень та адаптивності проєкту до змін, без порушення його мети, задач і ключових параметрів.

На відміну від класичного рефакторингу коду, який змінює внутрішню структуру програмного забезпечення без зміни зовнішньої поведінки, рефакторинг управління IT-проєктами модифікує процеси, ролі та комунікаційні структури, зберігаючи незмінними стратегічні інваріанти проєкту.

Рефакторинг управління IT-проєктами уключає п’ять основних етапів і один контрольний контур перевірки інваріантів [11].

1. Ідентифікація зон для покращень

На цьому етапі здійснюється автоматизований аналіз інформаційних потоків (Jira, Confluence, Slack, Service Desk, звіти спринтів та команди) із використанням методів NLP та машинного навчання для виявлення проблемних точок управління. Визначаються зони покращення: організаційні, процесні, аналітичні, інженерні, комунікаційні, дослідницькі та навчальні. Результатом етапу є карта проблемних областей, що вимагають подальшого аналізу або формування гіпотез.

Перед рефакторингом в управлінні IT-проєктом потрібно «зафіксувати» «Baseline Management Configuration». Це еталонний набір параметрів, що визначають сутність цього проєкту. Він формально фіксується

в HMS або в системі управління проєктом перед початком циклу рефакторингу за шаблоном, який подано в таблл.3.1.

Таблиця 3.1 – Шаблон фіксації інваріантів проєкту «Baseline Management Configuration»

Категорія	Що фіксується
Мета проєкту	основна бізнес- або соціальна ціль
Основні задачі (scope)	верхньорівневі deliverables
Ключові показники (KPI / OKR)	метрики успіху
Архітектурна модель управління	структура ролей, взаємодій, каналів
Часові рамки та віхи (Milestones)	незмінні контрольні точки
Обмеження (Constraints)	технічні, бюджетні, правові

Додатковими об’єктами контролю є:

- зовнішні обмеження (compliance, стандарти);
- принципи управління ризиками;
- політика безперервності (continuity policy), елементи, які визначають «рамку стабільності» IT-проєкту.

2. Аналіз причинно-наслідкових зв’язків проблем

Для кожної ідентифікованої зони застосовуються методи причинного аналізу (Causal Loop Analysis, Knowledge Graph Reasoning), що дозволяють побудувати модель впливів між процесами, командами та результатами. Це дає змогу встановити, які фактори призводять до збоїв. Наприклад, затримки у комунікаціях, дублювання вимог чи неузгодженість архітектурних рішень.

Результатом є формалізована причинно-наслідкова карта (Causal Map) управлінських проблем.

3. Формування гіпотез для усунення проблем управління

На основі карти причин формуються технічні та організаційні гіпотези за допомогою HMS (Hypothesis Management System). Кожна гіпотеза описує можливе рішення у форматі:

Якщо змінити процес/структуру/комунікацію → то очікуваний ефект = Δ у KPI або продуктивності команди.

До гіпотез застосовується двоетапний механізм фільтрації:

1) за цінністю (наскільки вирішення проблеми важливе для цілей проєкту);

2) за здійсненністю (чи можливо перевірити гіпотезу з наявними ресурсами).

Результатом етапу є перелік відібраних, релевантних і перевірних гіпотез.

4. Перевірка ефективності запропонованих рішень

Гіпотези реалізуються у вигляді експериментів або обмежених змін: Spike-досліджень, симуляцій, A/B-перевірок управлінських сценаріїв. Для оцінки результатів застосовується аналітика ефективності (Performance Metrics, Time Series Analysis). Отримані дані передаються до HMS, де автоматично визначається успішність гіпотези. Непідтверджені рішення відхиляються або переформулюються.

5. Актуалізація структури управління.

Після перевірки ефективності гіпотез в систему управління IT-проєктами вносяться корекції:

1) оновлюються ролі, відповідальність, комунікаційні канали, workflow і backlog;

2) зміни синхронізуються з інструментами проєктного менеджменту (Jira, Azure DevOps, тощо);

3) оновлений backlog включає нові задачі, уточнені вимоги або змінені пріоритети.

6. Контур перевірки інваріантів (Management Regression Testing).

Кожен цикл рефакторингу супроводжується перевіркою того, що внесені зміни не порушили основні інваріанти IT-проєкту: мету, обсяг, ключові показники, часові рамки, обмеження та архітектурні принципи управління.

Система перевіряє:

$$G_{t+1} = G_t, \qquad S_{t+1} = S_t, | \, KPI_{t+1} - KPI_t \, | < \varepsilon_{th}$$

де : G – вектор цілей;

S – вектор задач (scope elements);

ε_{th} – допустиме відхилення.

Кожна зміна (refactoring patch) проходить валідацію:

$$\Delta R \rightarrow Test(G_{const}, S_{const})$$

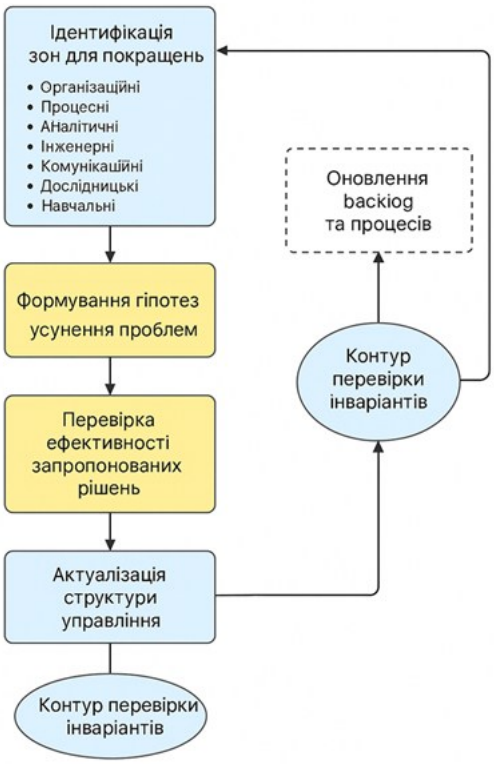
і якщо

$$Deviation(Goals, Scope) < \varepsilon_{th},$$

то зміни приймаються; якщо ж перевищує поріг — зміни відхиляються або потребують корекції. (ε_{th} – допустиме відхилення, наприклад, 2–3% від базових KPI).

Лише після успішної перевірки зміни приймаються, і рефакторинг вважається завершеним. Таким чином, процедура рефакторингу в управлінні IT-проєктами є безперервним циклом самооптимізації системи управління, який працює за принципом *«від невизначеності — через гіпотезу — до стабільного покращення»*. Вона інтегрується з циклами Agile/Scrum, оновлюючи не тільки backlog проєкту, а й модель управління ним, забезпечуючи її стійкість, узгодженість і відповідність стратегічним цілям. Структурна послідовність процедури представлена на рисунку 3.4.

Оскільки людський фактор часто є джерелом упередженості та ризиків або конфліктів, у складних IT-проєктах доцільно максимально автоматизувати етапи аналізу та прийняття рішень, де це можливо без втрати контексту. Для цього пропонується автоматизація трьох ключових процесів рефакторингу: (1) виявлення зон для покращення, (2) виявлення зон, що потребують додаткових досліджень, і (3) формування рекомендацій для подальших дій.



Загальний принцип кольорів

- 🟡 = *людина керує (Human-driven)*
- 💙 = *машина аналізує (AI-driven)*
- ⬜ = *дані зберігаються або передаються (Data artifacts)*

Рисунок 3.4 – Структура процедури рефакторингу в управлінні IT-проєктами. Джерело: розроблено автором

Для реалізації цих етапів доцільно залучати інтелектуальні моделі та інструменти, які представлені у таблиці 3.2.

Таблиця 3.2 – Моделі та методи забезпечення рефакторингу складних IT-проєктів

Модель / метод	Призначення	Приклади інструментів
Topic Modeling (LDA, BERT)	Виявлення тематичних кластерів проблем	spaCy, BERTopic
Text Similarity Analysis	Виявлення дублікатів вимог і гіпотез	SentenceTransformers, OpenAI Embeddings
Knowledge Graph Reasoning	Виявлення прогалин у знаннях та зв'язках між задачами	Neo4j, GraphDB
Active Learning	Автоматичне визначення напрямів для нових досліджень	Scikit-Learn, PyTorch
MCDA / AHP	Багатокритеріальний аналіз важливості проблем	Python-mcda, AHPy

Автоматизацію процедури рефакторингу представимо у вигляді послідовності етапів проведення робіт.

Етап 1. Виявлення зон для покращення (Improvement Zone Detection). Автоматичне виявлення зон, де спостерігається відхилення від планових показників управління уключає контур вхідної перевірки інваріантів — тобто система спочатку звіряє, які частини управлінської системи дають відхилення від інваріантів (KPI, scope, constraints).

Механізми автоматизації охоплюють:

- моніторинг KPI у Jira, Azure Boards, Confluence;
- аналіз сигналів «аномалій управління» (наприклад, надлишкові зміни пріоритетів, невідповідність velocity плану);
- використання Root Cause Analysis + Anomaly Detection ML-моделей.

Таким чином, процес починається з автоматичної перевірки, чи не порушено управлінські інваріанти.

Етап 2. Виявлення зон, що потребують додаткових досліджень (Exploratory Zone Detection) сформовано як процес формування та перевірки гіпотез, тобто — якщо система бачить, що причинно-наслідкові зв’язки не можуть бути доведені формально, вона автоматично створює задачу типу «Research Spike» у backlog.

Механізми автоматизації:

- Topic Modeling (BERT, LDA) для пошуку слабко описаних проблем;
- Graph Reasoning — виявлення відсутніх зв’язків між вимогами, командами чи артефактами;
- Active Learning — система пропонує «зони для подальших досліджень» на основі невизначеностей у даних.

Тобто, виявлення зон, що потребують додаткових досліджень інтегровано з Hypothesis Management System (HMS) — саме HMS керує переходом *невизначеність → гіпотеза → spike → backlog*.

Етап 3. Формування рекомендацій для подальших дій (Recommendation Generation) завершує вихідний контур перевірки інваріантів, тобто система формує рекомендації після підтвердження, що зміни не порушують інваріанти.

Механізми автоматизації:

- Rule-based моделі («якщо-знайдена-причина → то-покращення»);
- LLM-based генератори (на кшталт OpenAI API або LangChain) для формування текстових рекомендацій і оновлених backlog entries;
- Simulation Loop (Monte Carlo, sensitivity analysis) — перевірка стабільності пропозицій перед імплементацією.

Процес уключає етап регресійного тестування управлінських інваріантів перед прийняттям результату. Підсумкова структура автоматизації рефакторингу подана у таблиці 3.3.

Таблиця 3.3 – Структура автоматизації рефакторингу

№	Автоматизований процес	Нове уточнення	Інструменти
1	Виявлення зон для покращення	Через контур перевірки інваріантів та аномалії KPI	ML Anomaly Detection, RCA, KPI Dashboard
2	Виявлення зон для подальших досліджень	Інтегровано з HMS, автоматичне створення Research Spikes	BERT, Graph Reasoning, Active Learning
3	Формування рекомендацій	Через AI-регресійне тестування управлінських інваріантів	Rule-based + LLM Models, Monte Carlo Simulation

Доведення адекватності, достовірності та стійкості нового методу є ключовим етапом його наукової валідації та практичної прийнятності. Адекватність означає відповідність методу його предметній області, поставленим задачам і типу вхідних даних.

Для цього необхідно встановити, що внутрішня логіка методу, його припущення та модельні залежності узгоджуються зі структурою об’єкта дослідження, а також що отримані результати мають змістовну інтерпретацію у відповідному контексті. Достовірність методу визначається ступенем збігу між прогнозованими чи розрахунковими результатами і фактичними або

експертно визнаними значеннями, що підтверджується емпіричними тестами, статистичними критеріями узгодженості та показниками точності (Precision, Recall, F1).

Стійкість відображає здатність методу зберігати інваріантність результатів при варіаціях вхідних даних, шумі чи неповноті інформації. Її перевіряють шляхом аналізу чутливості та стохастичних експериментів типу Monte Carlo.

Отже, доказ адекватності, достовірності та стійкості передбачає не лише кількісну оцінку метрик, але й концептуальне обґрунтування відповідності методу своїй проблемній галузі, що перетворює його з гіпотетичного підходу на науково підтверджений інструмент пізнання та управління складними системами.

3.2.2 Валідація адекватності та стійкості методу рефакторингу управління ІТ-проєктах

Процес наукової валідації нового методу управління має на меті підтвердити, що він не лише коректно реалізує свої внутрішні алгоритмічні процедури, а й адекватно відображає закономірності предметної області, достовірно відтворює реальні управлінські процеси та стійко функціонує в умовах невизначеності.

Для методу рефакторингу управління складними ІТ-проєктами, який базується на застосуванні моделей інтелектуального аналізу даних (ІАД) та систем штучного інтелекту (ШІ), це означає підтвердження того, що система формує релевантні, узгоджені та відтворювані управлінські рішення, не порушуючи інваріанти цілей проєкту.

Валідація проводиться у трьох взаємопов’язаних площинах: адекватності, достовірності та стійкості результатів. Кожна з них перевіряється за допомогою кількісних і якісних показників, що відображають здатність системи коректно відтворювати процеси рефакторингу управління.

Адекватність (А) характеризує ступінь відповідності результатів, сформованих системою, експертним управлінським рішенням у тій самій

ситуації. Вона відображає, наскільки метод «розуміє» контекст IT-проєкту і формує логічно узгоджені пропозиції. Для цього використовується формула порівняння результатів, аналогічна метрикам оцінки моделі в машинному навчанні, але адаптована до управлінських процесів:

$$A = 1 - \frac{| R_{AI} - R_{exp} |}{R_{exp}}$$

де: R_{AI}— результат, запропонований системою ІАД/ШІ (кількість або індекс схвалених рекомендацій);

R_{exp}— результат, отриманий експертами вручну.

Формула є узагальненням метрики *Mean Relative Error*, застосованої до управлінських рішень. Значення (A) наближається до 1 при високій адекватності результатів.

Достовірність (D) визначається як спроможність системи правильно виявляти релевантні зони для рефакторингу, не генеруючи надлишкових або хибних рекомендацій. Для цього використовується модифікована F1-метрика, що походить з теорії виявлення сигналів і широко застосовується у Data Science:

$$D = F_1 = \frac{2PR}{P + R}$$

де P – *Precision*, точність визначення релевантних зон покращення;

R – *Recall*, повнота виявлення;

F_1 – узагальнений коефіцієнт достовірності (0–1).

Цей показник оцінює баланс між кількістю виявлених релевантних проблем та кількістю пропущених або хибних сигналів, аналогічно до оцінки класифікаційних моделей.

Для поглибленого аналізу достовірності застосовується також апарат теорії виявлення цілей (Detection Theory), який дозволяє обчислити ймовірності правильного виявлення (PD) та помилкової тривоги (PFA):

$$PD = \frac{TP}{TP + FN}, \qquad PFA = \frac{FP}{FP + TN}$$

де TP — кількість правильно виявлених релевантних проблем;

FP — кількість хибних виявлень;

FN — кількість пропущених проблем;

TN — кількість випадків, де система правильно не запропонувала змін.

Високе значення PD при низькому PFA свідчить про надійність методу у виділенні справді значущих гіпотез.

Стійкість (S) методу відображає його здатність зберігати стабільність результатів при зміні вхідних даних або зовнішніх умов.

Для цього використовується коефіцієнт варіації результатів за багаторазових запусків із різними параметрами, який є аналогом показника робастності у статистиці:

$$S = 1 - \frac{\sigma_R}{\mu_R}$$

де : σ_R— стандартне відхилення результатів рефакторингу за декількох експериментів;

μ_R— середнє значення результатів, ум. од.

Якщо $S \to 1$, метод є стійким до шумів, неповноти або суперечливості даних.

Для оцінки чутливості системи до параметрів проводиться аналіз впливу факторів (Sensitivity Analysis). Один з найефективніших підходів — метод Соболя [13], який дозволяє визначити частку дисперсії результату, що пояснюється варіацією кожного вхідного параметра:

$$S_i = \frac{V_i}{V_{total}}$$

де : S_i – індекс Соболя для параметра i;

V_i – часткова дисперсія, зумовлена параметром i;

V_{total} – загальна дисперсія результатів.

Значення $S_i > 0.5$ свідчить, що параметр суттєво впливає на результат рефакторингу, і модель повинна компенсувати його варіативність.

Для перевірки стійкості в динаміці проєкту застосовується імітаційне моделювання методом Monte Carlo, у якому виконується серія запусків моделі з випадковими варіаціями параметрів (наприклад: рівень шуму у даних, частота оновлення backlog, кількість стейкхолдерів). Результати статистично узагальнюються для побудови довірчих інтервалів оцінок (A, D, S).

Метод вважається науково валідованим, якщо отримані значення задовольняють порогові критерії:

$$A > 0.8, \qquad D > 0.85, \qquad S > 0.9$$

що означає адекватну інтерпретацію, високу точність та робастність результатів відповідно.

Валідація методу рефакторингу управління IT-проєктами поєднує кількісну метричну оцінку з концептуальною перевіркою відповідності моделі управлінським реаліям. Отримані показники свідчать про здатність запропонованого методу формувати достовірні, відтворювані та стабільні рішення навіть у середовищі з високим рівнем невизначеності — що робить його придатним для практичного застосування в системах підтримки управління складними IT-проєктами.

Таким чином, рефакторинг-метод управління складними IT-проєктами це концепція адаптивного удосконалення управлінських структур із

використанням інструментів ШІ, аналітики даних і семантичних технологій. Він дозволяє системно виявляти слабкі місця управління, автоматично формувати гіпотези покращень та генерувати рекомендації для управлінських рішень. Його впровадження підвищує стійкість, узгодженість і прозорість процесів у складних ІТ-проєктах.

3.4 Метод управління складними ІТ-проєктами в умовах невизначеності

Розробка та реалізація складних ІТ-проєктів характеризуються високим рівнем невизначеності, динамікою змін, множинністю стейкхолдерів і значною кількістю взаємопов’язаних компонентів. Традиційні підходи до управління проєктами, орієнтовані на фіксовані плани та послідовне виконання етапів, виявляються недостатньо ефективними в умовах постійного розвитку технологій та непередбачуваних факторів середовища. У таких випадках необхідним є метод, який поєднує гнучкість адаптивних моделей, науковий підхід до усунення невизначеності та можливість автоматизації управлінських процесів.

Запропонований метод управління складними ІТ-проєктами в умовах невизначеності базується на концептуальній моделі управління з НДДКР [14], доповненій трьома ключовими інструментами: Канвасом формування гіпотез [2], Двоетапним методом фільтрації гіпотез та Рефакторинг-методом управління ІТ-проєктах [11].

Зазначимо, що класичні етапи, такі як ініціація, формування вимог та команди, перевірка реалізуємості та розподіл ролей, в роботі не розглядаються. Метод починається з ідентифікації зон невизначеності, що формують основу для подальших досліджень, і завершується внесенням перевірених змін у структуру управління. Структура методу подана на рисунку 3.5.

Рисунок 3.5 – Метод управління складними IT-проєктами в умовах невизначеності. Джерело: розроблено автором

Першим кроком є ідентифікація зон невизначеності, під час якої система управління отримує сигнали з різних джерел: backlog, Jira, звітів команди, відгуків користувачів. За допомогою методів обробки природної мови (NLP) здійснюється автоматичне вилучення змістових одиниць: проблем, ризиків, суперечностей і прогалин у даних. На основі цього формується база проблем (Problem Base), що відображає зони, де відсутня визначеність або де спостерігаються ризики складності.

Для опису алгоритму, або послідовності методу використаємо псевдокод (pseudocode), як спосіб, наближеній до коду, але без дотримання

строгої синтаксичної структури конкретної мови програмування (Python, Java тощо). Зазвичай його використовують для пояснення логіки або послідовності дій алгоритму, не заглиблюючись у технічні деталі реалізації.

Було використано формальний, але зрозумілий стиль — між алгоритмічною нотацією та Python-подібним синтаксисом, щоб його можна було згодом легко перетворити на реальний код HMS-системи. Псевдокод 1 етапу методу поданий в лістінгу 3.3.

Лістінг 3.3 – Псвевдокод 1 етапу методу управління складними IT-проєктами

```
FUNCTION Identify_Uncertainty_Zones(data_sources):
  problem_base = []
  FOR source IN data_sources:
    raw_data = Extract_Text(source)
    processed_data = NLP_Preprocess(raw_data)
    issues = Extract_Issues(processed_data)
    problem_base.ADD(issues)
  classified_problems = Classify_By_Type(problem_base)
  RETURN classified_problems
```

Коментар: Здійснюється вилучення проблемних сигналів із backlog, Jira, Confluence, Service Desk тощо. NLP-модуль формує “Problem Base”, класифікуючи записи за категоріями (функціональні, UX, безпека, архітектура тощо).

На другому етапі виконується формування гіпотез на основі канвасу методу формування гіпотез, який було представлено в підрозділі 3.1. Кожна зона невизначеності описується у структурі: *проблема – гіпотеза – очікуваний результат – критерії перевірки – ресурси*. Такий формат дозволяє стандартизувати мислення команди, формалізувати припущення та забезпечити можливість подальшої перевірки. Структуровані гіпотези зберігаються у базі знань системи HMS, де вони можуть бути згруповані за семантичними зв’язками або технологічними напрямами. Псевдокод 2 етапу методу поданий в лістінгу 3.4.

Лістінг 3.4 – Псевдокод 2 етапу методу управління складними IT-проєктами

```
FUNCTION Generate_Hypotheses(problem_list):
  hypotheses = []
  FOR problem IN problem_list:
    hypothesis = CREATE Canvas_Hypothesis(
      Problem = problem.description,
      Hypothesis = «IF we apply [method] THEN [effect] BECAUSE [reason]»,
      Expected_Result = Predict_Outcome(problem),
      Success_Criteria = Define_Criteria(problem),
      Resources = Estimate_Resources(problem)
    )
    hypotheses.ADD(hypothesis)
  RETURN hypotheses
```

Коментар: Кожна проблема перетворюється в гіпотезу за шаблоном *«Якщо… то… бо…»*. Створюється структурована база даних гіпотез (HMS Knowledge Base).

Третім етапом є двоетапна фільтрація гіпотез, яка забезпечує відбір лише тих, що мають високу цінність для проєкту та можуть бути реалізовані на практиці. Перший фільтр оцінює вплив проблеми на стратегічні цілі за багатокритеріальною моделлю (MCDA), де обчислюється інтегральний індекс важливості (*I_{value}*), що враховує вплив, ризик, масштаб і кількість стейкхолдерів. Другий фільтр оцінює здійсненність через індекс (*I_{feas}*), який включає часові, фінансові та ресурсні обмеження. Гіпотези з низькими значеннями індексів автоматично відсіюються як надлишкові або нераціональні. Псевдокод 3 етапу методу поданий в лістингу 3.5.

Лістинг 3.5 – Псевдокод 3 етапу методу управління складними IT-проєктами

```
FUNCTION Generate_Hypotheses(problem_list):
  hypotheses = []
  FOR problem IN problem_list:
    hypothesis = CREATE Canvas_Hypothesis(
      Problem = problem.description,
      Hypothesis = «IF we apply [method] THEN [effect] BECAUSE [reason]»,
      Expected_Result = Predict_Outcome(problem),
      Success_Criteria = Define_Criteria(problem),
      Resources = Estimate_Resources(problem)       )
    hypotheses.ADD(hypothesis)
```

```
    RETURN hypotheses
```

Коментар: MCDA (Multi-Criteria Decision Analysis) обчислює індекс важливості. Feasibility_Index — оцінка здійсненності (ресурси, час, ризики). Зберігаються лише релевантні гіпотези.

Четвертий етап — перевірка гіпотез, у ході якої проводяться Spikes, Proof of Concept (PoC) та дослідно-конструкторські роботи. Ці дії дозволяють підтвердити або спростувати припущення на практиці, а також перевірити ефективність обраних технічних або управлінських рішень. Кожна гіпотеза отримує статус: *підтверджена*, *відхилена* або *потребує додаткових досліджень*. Це що дозволяє відстежувати прогрес і зберігати історію прийнятих рішень. Псевдокод 4 етапу методу поданий в лістингу 3.6.

Лістинг 3.6 – Псевдокод 4 етапу методу управління складними IT-проєктами

```
FUNCTION Test_Hypotheses(validated_hypotheses):
  FOR h IN validated_hypotheses:
    experiment = Design_Experiment(h)
    results = Execute_Experiment(experiment)
    evaluation = Evaluate_Results(results, h.Success_Criteria)
    IF evaluation.Passed:
      h.Status = «Confirmed»
    ELSE IF evaluation.Partial:
      h.Status = «Retest»
    ELSE:
      h.Status = «Rejected»
    Save_Results(h, results)
  RETURN validated_hypotheses
```

Коментар: Для кожної гіпотези створюється короткий експеримент (Spike або PoC). Результати зберігаються у HMS із зазначенням статусу.

П’ятий етап — рефакторинг управління IT-проєктами, який є основою адаптивності методу. На цьому етапі здійснюється виявлення зон для покращення (організаційних, комунікаційних, методологічних), визначення напрямів подальших досліджень і підготовка рекомендацій для наступних етапів. Запроваджується контур перевірки інваріантів — механізм, що забезпечує незмінність ключових характеристик проєкту, таких як мета, основні завдання та обмеження, навіть після внесення управлінських змін. Це

аналог «регресійного тестування» у програмному забезпеченні, але застосованого до управлінських процесів. Псевдокод 5 етапу методу поданий в лістингу 3.7.

Лістинг 3.7 – Псевдокод 5 етапу методу управління складними IT-проєктами

```
FUNCTION Refactor_Management_System(confirmed_hypotheses):
   improvement_zones = Detect_Improvement_Zones(confirmed_hypotheses)
   research_zones = Detect_Research_Zones(confirmed_hypotheses)
   recommendations = Generate_Recommendations(improvement_zones, research_zones)

   invariants = Load_Project_Invariants()
   IF NOT Violates_Invariants(recommendations, invariants):
      Apply_Changes_To_Management(recommendations)
   ELSE:
      Raise_Alert(«Invariant violation detected»)
   RETURN recommendations
```

Коментар: Рефакторинг — це адаптація процесів управління (не коду). Контур перевірки інваріантів гарантує, що мета, ключові завдання й обмеження проєкту залишаються незмінними.

Шостий етап — актуалізація структури управління, який передбачає перенесення результатів рефакторингу в операційну модель. Це включає оновлення backlog, планів спринтів, комунікаційних схем, ролей і відповідальностей. Усі зміни документуються, а їхня ефективність оцінюється на основі порівняльних метрик (наприклад, часу життєвого циклу вимог або швидкості прийняття рішень). Псевдокод 6 етапу методу поданий в лістингу 3.8.

Лістинг 3.8 – Псевдокод 6 етапу методу управління складними IT-проєктами

```
FUNCTION Update_Management_Structure(recommendations):
   backlog = Load_Current_Backlog()
   updated_backlog = Integrate_Changes(backlog, recommendations)
   Update_Sprint_Plans(updated_backlog)
   Update_Roles_And_Responsibilities(recommendations)
   Log_Changes_To_KnowledgeBase(updated_backlog)
   RETURN updated_backlog
```

Коментар: Усі зміни фіксуються в системі управління (наприклад, Jira, Azure DevOps). Backlog оновлюється, плани спринтів коригуються автоматично.

Завершальним є етап циклічного самонавчання, де система HMS накопичує дані про гіпотези, дослідження та результати рефакторингу. Використовуючи методи машинного навчання, система прогнозує нові зони невизначеності, підвищуючи стійкість і точність майбутніх ітерацій управління. Таким чином створюється замкнений адаптивний контур, у якому кожен цикл не лише зменшує невизначеність, але й підвищує зрілість системи управління. Псевдокод 7 етапу методу поданий в лістингу 3.9.

Лістинг 3.9 – Псевдокод 7 етапу методу управління складними IT-проєктами

```
FUNCTION Self_Learning_Cycle(history_data):
    patterns = Train_ML_Model(history_data)
    predicted_zones = Predict_New_Uncertainties(patterns)
    Save_Predictions(predicted_zones)
    RETURN predicted_zones
```

Коментар: Система HMS використовує машинне навчання для прогнозування майбутніх зон невизначеності та рекомендацій наступного циклу рефакторингу.

Псевдокод головного циклу методу поданий в лістингу 3.10.

Лістинг 3.10 – Псевдокод головного циклу методу управління складними IT-проєктами

```
FUNCTION Manage_Complex_ITP():
    problems = Identify_Uncertainty_Zones(data_sources)
    hypotheses = Generate_Hypotheses(problems)
    filtered_hypotheses = Filter_Hypotheses(hypotheses, T_value, T_feas)
    tested_hypotheses = Test_Hypotheses(filtered_hypotheses)
    recommendations = Refactor_Management_System(tested_hypotheses)
    new_backlog = Update_Management_Structure(recommendations)
    predicted_zones = Self_Learning_Cycle(HMS_History)
    RETURN new_backlog, predicted_zones
```

Коментар: Цей головний цикл описує повний адаптивний контур управління ІТ-проєктами, який безперервно зменшує невизначеність ІТ-проєкту, а результати первірки гіпотез — у структурні покращення проєкту.

Запропонований метод дозволяє інтегрувати наукове мислення, моделі, які представлені у другому розділі та автоматизацію в єдину управлінську структуру. Його особливістю є здатність не просто реагувати на зміни, а перетворювати їх на джерело вдосконалення. Внаслідок цього управління складними ІТ-проєктами набуває властивостей самокорекції, гнучкості та прогнозованості, що є важливим для інноваційних систем із високим рівнем технологічної невизначеності.

Висновки до розділу 3

У третьому розділі сформовано цілісну методологічну систему управління складними ІТ-проєктами, засновану на інтеграції наукових принципів гіпотетико-дедуктивного підходу, інструментів аналізу даних і концепції адаптивного рефакторингу управлінських процесів. Запропоновані методи дозволяють не лише зменшити невизначеність під час реалізації ІТ-проєктів, а й створити самонавчальну систему управління, здатну постійно вдосконалюватися на основі емпіричних результатів і зворотного зв’язку.

1. Розроблено канвас методу формування гіпотез («Hypothesis Canvas 2.0 для ІТ-проєктів»), який адаптовано до специфіки технічних і архітектурних рішень у складних ІТ-проєктах. Його відмінність від класичних бізнесових моделей полягає у спрямованості на технічну життєздатність і критичність гіпотез. Запропонований формат забезпечує уніфікацію процесу формування технічних припущень, забезпечує допомогу у прийнятті рішень і створює основу для подальшої автоматизації управлінських дій. Канвас представляє структурну інтерпретацію як перетворити невизначені знання в гіпотези з визначеними критеріями успішності.

2. Розроблено двоетапний метод формування гіпотез, який реалізує повний цикл перетворення неструктурованих сигналів (проблем, відгуків,

backlog-записів) у перевірені технічні гіпотези. Метод охоплює етапи збору, класифікації та нормалізації даних із застосуванням NLP, подальшу фільтрацію за двома критеріями — *цінністю проблеми* (*I_value*) та *здійсненністю перевірки* (*I_feas*). Завдяки використанню алгоритмів машинного навчання, таких як TF-IDF, K-Means і моделей подібності, створюється автоматизований контур аналізу, що мінімізує людський вплив та прискорює процес управлінських рішень. Метод забезпечує формування узгодженого реєстру гіпотез, готових до реалізації, і впроваджує елементи безперервного навчання через зворотний зв'язок системи HMS.

3. Представлено рефакторинг-метод управління складними IT-проєктами, що переносить концепцію рефакторингу з програмної інженерії до сфери управління проєктами. Запропонована процедура передбачає поетапне вдосконалення структури управління IT-проєктами без порушення інваріантів проєкту, а саме: його мети, обсягу, KPI та архітектурно-управлінських принципів. Метод включає шість етапів, серед яких: виявлення зон покращення, аналіз причинно-наслідкових зв'язків, формування гіпотез управлінських змін, перевірка ефективності, актуалізація структури управління та контур перевірки інваріантів. Визначено три процеси, що підлягають автоматизації: *виявлення зон для покращення*, *виявлення зон для подальших досліджень* та *генерацію рекомендацій*. Їх реалізація базується на поєднанні ML, графових імовірнісних моделей та LLM-алгоритмів.

4. Проведено валідацію адекватності, достовірності та стійкості методу рефакторингу, яка довела його наукову обґрунтованість і практичну придатність. Встановлено порогові критерії: адекватність $A > 0.8$, достовірність $D > 0.85$, стійкість $S > 0.9$, які підтверджують здатність системи формувати узгоджені та відтворювані рішення навіть за умов невизначеності даних.

5. В підрозділі 3.4 інтегровано всі попередні результати в єдиний метод управління складними IT-проєктами, який забезпечує повний цикл управління: від виявлення зон невизначеності до оновлення моделі

управління. Запропонований алгоритм (псевдокод) реалізує контур безперервного самонавчання системи управління, де результати перевірених гіпотез стають основою для подальших проєктних рішень.

6. Отже, розділ 3 формує науково-практичну основу нового типу управління складними ІТ-проєктами — адаптивно-інтелектуального управління, що поєднує гіпотетико-дедуктивну логіку, машинне навчання та принципи рефакторингу управлінських процесів. Такий підхід забезпечує стійкість системи до невизначеності, підвищує її когерентність і сприяє перетворенню управління на динамічну, самонавчальну систему, здатну до науково обґрунтованого вдосконалення у реальному часі.

7. Результати досліджень третього розділу опубліковані в роботах [2, 11].

Список використаних джерел у розділі 3

1. Ostenwalder A., Pigneur Y., Tucci C.L. (2005). Clarifying Business Models: Origins, Present, and Future of the Concept. Communications of AIS, Volume 15, 1 – 33. https://www.kth.se/social/files/546b8d75f276546614d2dffc/Osterwalder%2B% 282005%29.pdf?utm_source=chatgpt.com

2. Путій І. Д., Тесленко П. О. (2025). «Канвас» методу формування гіпотез для складного ІТ-проєкту. Управління розвитком складних систем, 64, 120–127. DOI: 10.32347/2412-9933.2025.64.

3. Гороховатський В.О., Творошенко І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник. НУРЕ, 92. DOI: 10.30837/978-966-659-298-2

4. Dmytro Shestakov (2021). The Hypotheses Testing Method for Evaluation of Startup Projects. Journal of Economics and Management Sciences. https://doi.org/10.30560/jems.v4n4p47.

5. Murray, A., & Scuotto, V. (2016). The Business Model Canvas. Symphonya. Emerging Issues in Management, (3), 94–109. https://doi.org/10.4468/2015.3.13murray.scuotto

6. Kemell, K., Elonen, A., Suoranta, M., Nguyen-Duc, A., Garbajosa, J., Chanin, R.M., Melegati, J., Rafiq, U., Aldaeej, A., Assyne, N., Sales, A., Hyrynsalmi, S., Risku, J., Edison, H., & Abrahamsson, P. (2020). Business Model Canvas Should Pay More Attention to the Software Startup Team. 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 342-345. https://doi.org/10.48550/arXiv.2102.06500

7. Гришко В.В. (2023). Методологія наукових досліджень у гуманітарній сфері в умовах діджиталізації та великих даних: проблеми, помилки та перспективи досліджень. Ефективна економіка, 8.1. URL: http://nbuv.gov.ua/UJRN/efek_2023_8_3

8. Путій І.Д., Бондар О.А. (2024). Складність IT-проєктів з науково-дослідною та дослідно-конструкторською розробкою. Міжнародна науково-практична конференція «Інформаційні системи в управлінні проєктами та програмами», Коблево, ХНУРЕ, 197 –200.

9. Ялова К.М., Бердник В.О. (2021). Рефакторинг програмного коду з патернами проєктування. Інформаційні технології. Збірка тез VIII Всеукраїнської науково-практичної конференції молодих науковців, 20 трав. 2021 р., м. Київ. ун-т ім. Б. Грінченка, 157 – 159. https://eportfolio.kubg.edu.ua/data/conference/6858/document.pdf#page=157

10. Струзік, В. А., Грибков, С. В., & Чобану, В. В. (2019). Визначення місця рефакторингу в сучасних методологіях розробки інформаційних систем. Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки. Том 30 (69) Ч. 1 № 5, 273-277. DOI https://doi.org/10.32838/2663-5941/2019.5-1/28

11. Ілля Путій, Павло Тесленко. (2025). Рефакторинг-метод управління складними IT-проєктами. Project, Program, Portfolio Management. P3M-2025: Тези доповідей VIII Міжнародної науково-практичної конференції. Одеса. ІШІР, 190 – 193. https://www.doi.org/10.5281/zenodo.18428325

12. Hodovychenko M. A., Kurinko D. D. (2025). Analysis of existing approaches to automated refactoring of object-oriented software systems. Herald of

Advanced Information Technology. Vol. 8 No.2. 179–196. DOI: https://doi.org/10.15276/hait.08.2025.11

13. Соловей, О. (2025). Зважена метрика чутливості для прогнозування затримки в KAFKA-кластері. Сучасний стан наукових досліджень та технологій в промисловості, 3 (33), 152-165. DOI: https://doi.org/10.30837/2522-9818.2025.3.152

14. Путій, І. Д., & Тесленко, П. О. (2025). Концептуальна модель складного ІТ-проєкту. Таврійський науковий вісник. Серія: Технічні науки, (2), 148-154. DOI: https://doi.org/10.32782/tnv-tech.2025.2.16

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ СКЛАДНИМИ ІТ-ПРОЄКТАМИ В УМОВАХ НЕВИЗНАЧЕНОСТІ

4.1. Розробка інформаційної системи управління складним ІТ-проєктом

4.1.1 Мета та задачі інформаційної системи

Метою системи управління складними ІТ-проєктами є забезпечення керованості, прогнозованості та адаптивності проєктів в умовах невизначеності, технологічної складності та динамічних змін середовища. Система спрямована на зниження рівня проєктних невизначеностей шляхом їх раннього виявлення, аналітичної інтерпретації та перетворення у керовані управлінські дії, що дозволяє утримувати проєкт у контрольованому стані протягом усього життєвого циклу [1]. Система має підтримати керівника проєкту не лише на рівні контролю виконання плану, але й на рівні прийняття обґрунтованих рішень в ситуаціях, де класичні методи управління є недостатніми через нестачу інформації або складність взаємозв'язків між компонентами проєкту.

Основними задачами системи є централізований збір і консолідація даних про хід ІТ-проєкту з різних інформаційних джерел, формування єдиної проєктної бази спостережень та забезпечення її актуальності. Система має ідентифікувати зони невизначеності та складності, виявляти приховані проблеми у вимогах, процесах і технологіях, а також оцінювати їхній потенційний вплив на результати проєкту. Важливою задачею є автоматизоване формування управлінських гіпотез щодо зниження виявлених невизначеностей, їх оцінка та пріоритизація з урахуванням ризиків, ресурсних обмежень і стратегічних цілей [2].

Система також повинна забезпечувати підтримку реалізації та перевірки обраних гіпотез, аналіз ефективності прийнятих рішень і вимірювання рівня зниження невизначеності. Окремою задачею є адаптація плану управління

проєктом на основі отриманих результатів і запуск повторних управлінських циклів. В цілому система має створювати інтелектуальне середовище підтримки рішень, яке допоможе керівнику складного ІТ-проєкту діяти проактивно, своєчасно реагувати на зміни та підвищувати ймовірність успішного завершення проєкту.

4.1.2 Розробка варіантів використання інформаційної системи управління складними ІТ-проєктами

Актори системи управління складними ІТ-проєктами визначаються як користувачі та зовнішні інформаційні системи, що взаємодіють із системою в межах адаптивного управлінського контуру [3]. Їх ідентифікація є необхідною для формування варіантів використання, оскільки саме актори ініціюють управлінські дії, надають вхідні дані або споживають результати аналітичної обробки. Межі системи визначаються через сукупність сценаріїв, у яких система виконує функції збору даних, аналізу невизначеностей, формування управлінських гіпотез та підтримки прийняття рішень [4].

Ключовим актором є керівник складного ІТ-проєкту, який використовує систему для моніторингу стану проєкту, аналізу рівня невизначеності, перегляду рекомендацій та ухвалення рішень щодо адаптації плану управління. Він ініціює управлінські цикли, визначає допустимі порогові значення ризиків і невизначеностей, а також затверджує або відхиляє запропоновані управлінські гіпотези. Іншим важливим актором виступає проєктна команда, яка опосередковано взаємодіє з системою через інструменти розробки та управління проєктами. Її дії, такі як виконання завдань, коміти коду, оновлення документації чи результати тестування, формують первинні дані для аналітичних модулів системи.

Окрему групу акторів становлять зовнішні інформаційні системи управління проєктами та розробкою, зокрема Jira, Git-репозиторії, CI/CD-платформи, системи управління знаннями та QA-інструменти. Вони виступають джерелами структурованих і напівструктурованих даних, які надходять до модуля збору та консолідації даних. Через API або журнали

подій ці системи передають інформацію про статуси задач, зміни вимог, дефекти, швидкість виконання та інші показники, що є необхідними для ідентифікації складностей і невизначеностей.

Аналітик проєкту або системний аналітик може розглядатися як окремий актор, який використовує систему для детального аналізу виявлених проблемних зон, інтерпретації результатів класифікації невизначеностей та уточнення управлінських гіпотез. Він взаємодіє з модулем формування та оцінки гіпотез, коригуючи їхні параметри або критерії пріоритизації. Інженер з підтримки системи виступає технічним актором, відповідальним за стабільність функціонування самої інформаційної системи, контроль коректності інтеграцій, оновлення моделей аналітики та моніторинг продуктивності.

Варіанти використання системи формуються (рисунок 4.1) навколо типових управлінських сценаріїв складного IT-проєкту [1].

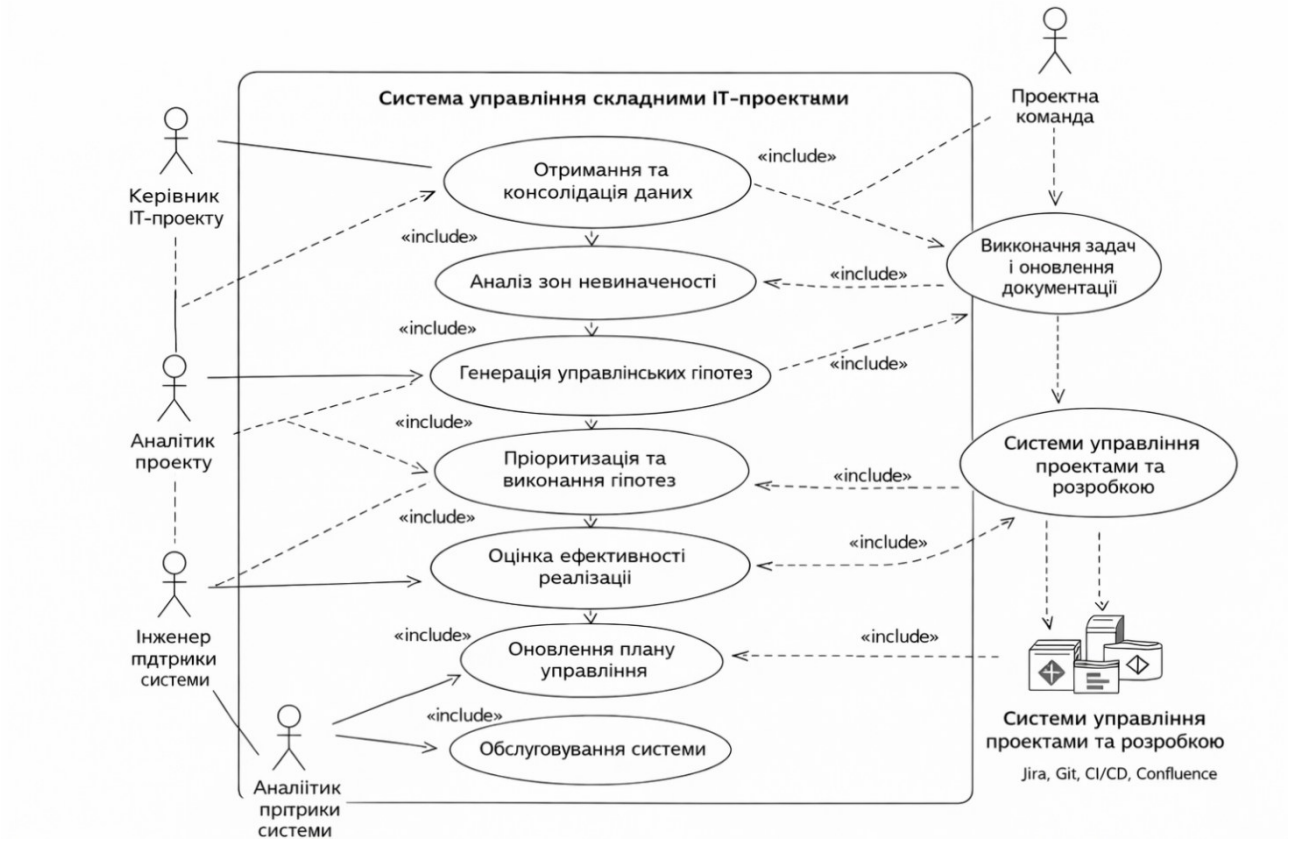


Рисунок 4.1 – Діаграма варіантів використання інформаційної системи. Джерело: розроблено автором

До них належить автоматизований моніторинг проєктного стану, виявлення зон невизначеності, генерація та оцінка управлінських гіпотез, аналіз ефективності реалізованих рішень і адаптація плану управління. Кожен варіант використання описує взаємодію одного або кількох акторів із відповідними модулями системи та визначає, яким чином система підтримує прийняття рішень у ситуаціях високої складності. Діаграма варіантів використання UML для такої системи формалізує ці сценарії, визначає відповідальність акторів і забезпечує узгодженість функціональних вимог із логічною структурою системи управління складними IT-проєктами [1].

4.1.3 Розробка діаграми класів інформаційної системи

Розробка класів для системи управління складними IT-проєктами базується на модульній архітектурі та принципах розділення відповідальностей та циклічної роботи з невизначеностями [5]. Діаграма класів системи управління складними IT-проєктами подана на рисунку 4.2 [1].

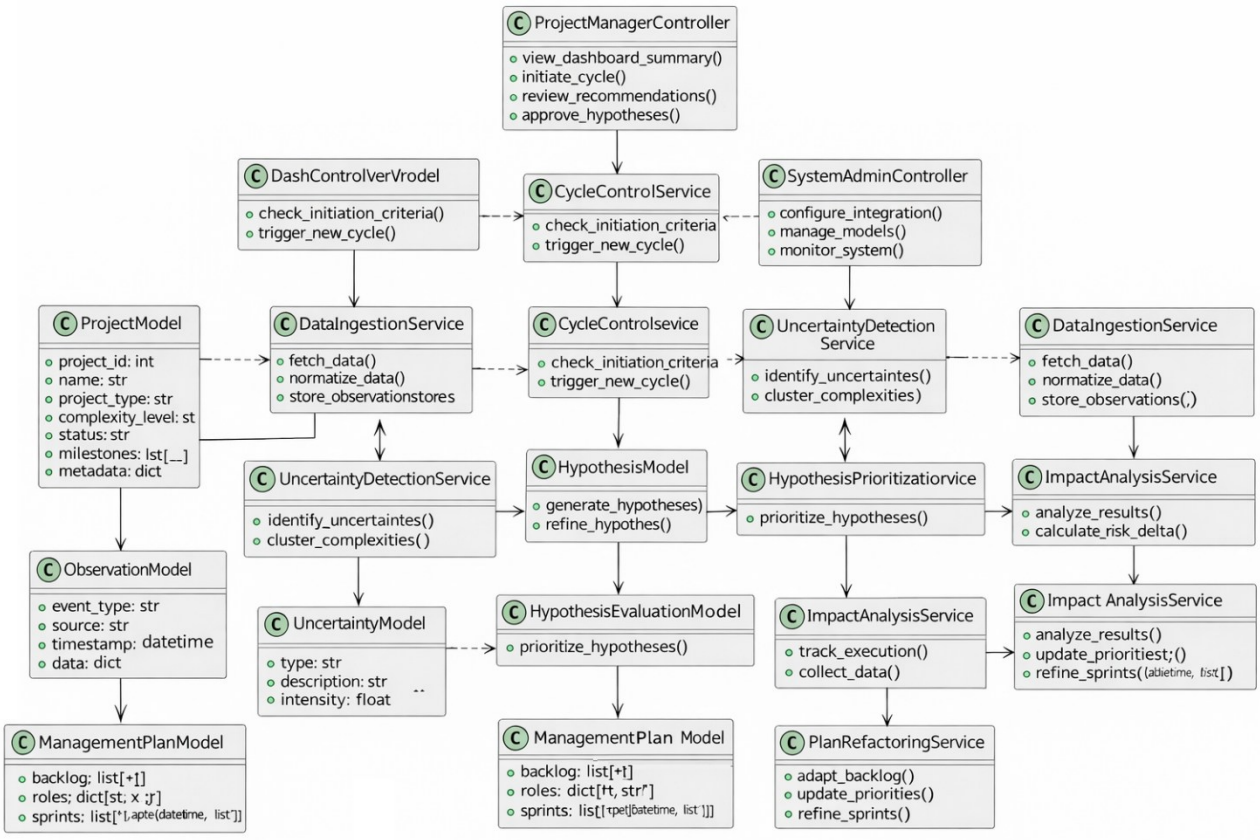


Рисунок 4.2 – Діаграма класів інформаційної системи управління складними IT-проєктами. Джерело: розроблено автором

ProjectModel – базова ORM-модель IT-проєкту, яка зберігає ідентифікатор, назву, тип проєкту, рівень складності, поточний стан, ключові віхи та метадані [1]. ObservationModel – модель проєктних спостережень, що описує зібрані факти з Jira, Git, CI/CD, QA-звітів, включаючи тип події, джерело, часову мітку та числові або текстові показники. DataIngestionService – сервіс збору та консолідації даних, відповідальний за інтеграцію з зовнішніми системами, нормалізацію форматів JSON та запис у проєктну базу спостережень. UncertaintyModel – структура даних для представлення виявлених невизначеностей і складностей, таких як нестабільні вимоги, технологічна новизна або аномальні затримки. UncertaintyDetectionService – сервіс ідентифікації невизначеностей, який використовує методи NLP, кластеризації та аналізу часових рядів для формування переліку проблемних зон. HypothesisModel – модель управлінської гіпотези з атрибутами типу, опису у форматі "*якщо–то–бо*", очікуваного ефекту та статусу перевірки. HypothesisGenerationService – сервіс формування гіпотез генерує можливі управлінські дії на основі виявлених невизначеностей та шаблонів з бази знань. HypothesisEvaluationModel – структура для зберігання оцінок гіпотез за критеріями цінності, здійсненності та ризику. HypothesisPrioritizationService – сервіс пріоритизації гіпотез, який реалізує багатокритеріальний аналіз та формує впорядкований список для реалізації. ExperimentExecutionModel – модель реалізації гіпотези, що фіксує виконані дії, використані ресурси, витрачений час і результати експерименту. ExperimentTrackingService – сервіс відстеження перевірки гіпотез і збору фактичних даних виконання. ImpactAssessmentModel – модель оцінки результатів, яка зберігає статус невизначеності, зміну ризику та досягнутий ефект. ImpactAnalysisService – сервіс аналізу результатів. Він порівнює стан системи до і після втручання. ManagementPlanModel – модель плану управління проєктом включає backlog, ролі, пріоритети та спринти. PlanRefactoringService – сервіс адаптації та рефакторингу плану управління на основі підтверджених гіпотез. CycleControlService – сервіс управління циклічністю, який ініціює нові ітерації

аналізу та контролює замкнений управлінський контур. DashboardViewModel – модель агрегованих показників для аналітичної панелі керівника з трендами невизначеності, прогресом та ефективністю рішень. ProjectManagerController – контролер взаємодії керівника проєкту з системою, що забезпечує доступ до аналітики та управлінських рішень. SystemAdminController – контролер адміністрування відповідає за налаштування інтеграцій, моделей аналізу та моніторинг працездатності системи.

Усі класи спроєктовані з дотриманням принципу єдиної відповідальності та підтримують розширюваність системи, що дозволяє масштабувати її під різні типи складних ІТ-проєктів і підключати нові аналітичні модулі без порушення цілісності архітектури.

4.1.4 Розробка логічного уявлення модулів інформаційної системи

Реалізація інформаційної системи базується на концепції адаптивного управління складними ІТ-проєктами. Основна ідея полягає у циклічному процесі виявлення невизначеностей, формування гіпотез щодо усунення невизначеностей, перевірки запропонованих дій та адаптації плану управління на основі отриманих результатів. Такий підхід дозволяє постійно знижувати рівень невизначеності, утримуючи проєкт у прогнозованому та контрольованому стані [6].

Процеси, що плануються для роботи ІС базуватимуться на методі управління складними ІТ-проєктами, який розроблено та представлено в підрозділі 3.4 дисертаційної роботи. Як було зазначено раніше, цей метод спирається на гнучкі адаптивні моделі, науковий підхід задля усунення невизначеностей складних ІТ-проєктів та має забезпечити можливість автоматизації управлінських процесів для забезпечення успішного завершення таких проєктів в умовах ризиків складності. Метод управління складними ІТ-проєктами уключає концептуальну модель управління з НДДКР, канвас формування гіпотез, двоетапний метод фільтрації гіпотез та рефакторинг-метод управління ІТ-проєктами.

Для реалізації цього бачення запропоновано структуру інформаційної системи, логічне уявлення якої представлено на рисунку 4.3.

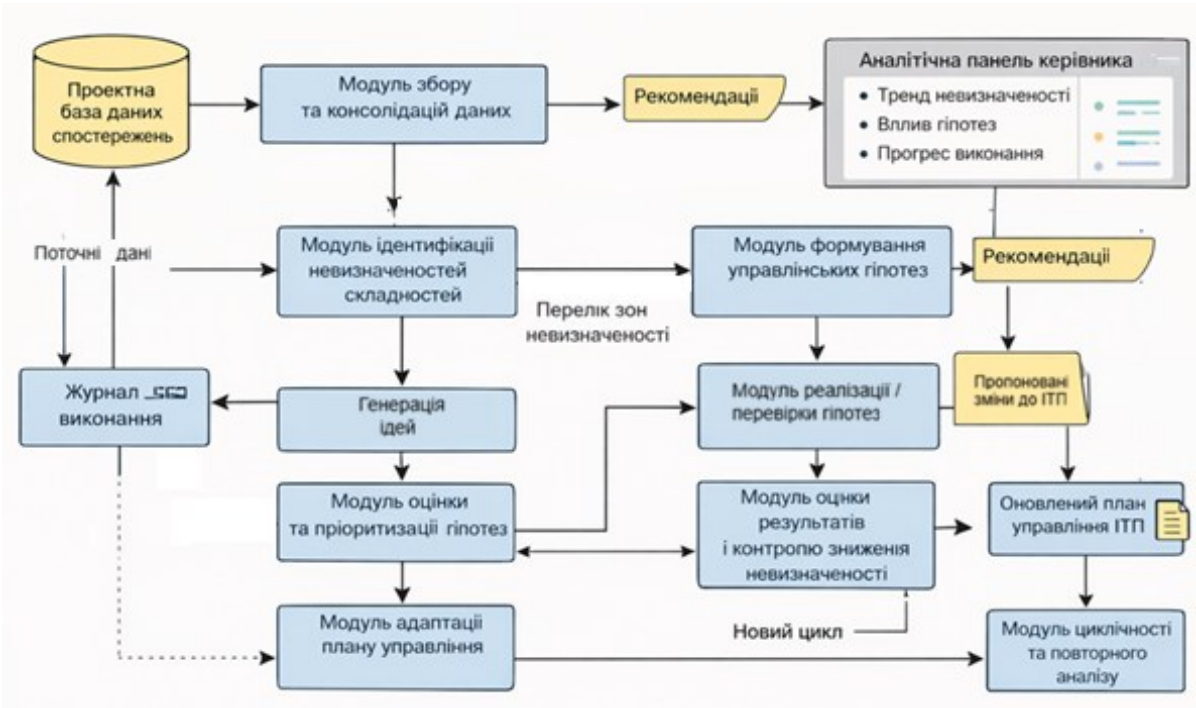


Рисунок 4.3 – Діаграма логічного уявлення модулів інформаційної системи (На рисунку позначено ІТП як ІТ-проєкт) . Джерело: розроблено автором

До складу структури включено наступні модулі [6]:

- модуль збору та консолідації даних;
- модуль ідентифікації невизначеностей і складностей;
- модуль формування управлінських гіпотез;
- модуль оцінки та пріоритизації гіпотез;
- модуль реалізації (перевірки) гіпотез;
- модуль оцінки результатів і контролю зниження невизначеності;
- модуль адаптації плану управління (рефакторинг);
- модуль циклічності та повторного аналізу;
- аналітична панель керівника.

Реалізація інформаційної системи управління складними IT-проєктами базується на концепції адаптивного, замкненого контуру управління, у якому дані, аналітика та управлінські рішення безперервно взаємопов'язані. Центральним елементом системи є проєктна база даних, що акумулює всі цифрові сліди життєвого циклу IT-проєкту. Вона формується модулем збору та консолідації даних, який отримує інформацію з операційних інструментів управління проєктом і розробкою: Jira, Git-репозиторіїв, Confluence, систем CI/CD, журналів виконання завдань, QA-звітів і логів тестування. Дані надходять у напівструктурованому вигляді у форматах JSON, CSV або через REST API та містять атрибути часу, відповідальних осіб, статусів, зв'язків між задачами, комітами, дефектами та релізами. На цьому етапі виконуються очищення, нормалізація, дедуплікація та часове вирівнювання даних із застосуванням методів ETL і базових статистичних перевірок якості. Результатом є уніфікована проєктна база спостережень, яка слугує єдиним джерелом даних для ІС.

Модуль ідентифікації невизначеностей і складностей працює безпосередньо з цією базою, використовуючи історичні та поточні дані про хід проєкту. Вхідними даними є тексти вимог, коментарі задач, журнали змін, метрики швидкості виконання, затримки та частота переробок. За допомогою методів NLP, тематичного моделювання, кластеризації та виявлення аномалій система визначає зони підвищеної невизначеності, такі як нечітко сформульовані вимоги, нестабільні компоненти, технологічна новизна або нетипові відхилення у термінах. На виході формується перелік зон невизначеності з кількісними оцінками їх типу, які передаються до наступного модуля.

Модуль формування управлінських гіпотез отримує ці зони як вхід і поєднує їх із знаннями про типові патерни управлінських дій, накопичені у базі [2]. Він використовує експертні системи та шаблони логічного виводу для генерації гіпотез у формалізованому вигляді «*якщо – то – бо*», де вказано очікуваний механізм ліквідації невизначеності. Результатом є набір

альтернативних управлінських гіпотез, структурованих у вигляді записів із параметрами дії, очікуваного ефекту та умов перевірки.

Далі модуль оцінки та пріоритизації гіпотез працює з цими записами, використовуючи багатокритеріальні моделі аналізу рішень. Вхідними даними є оцінки ризику, витрат, очікуваного виграшу, складності реалізації та стратегічної важливості. За допомогою методів MCDA та вагових моделей формується ранжований список гіпотез із числовими пріоритетами. На виході система передає обмежений перелік гіпотез, рекомендованих до перевірки.

Модуль реалізації або перевірки гіпотез інтегрується з операційними інструментами проєкту та відстежує виконання відповідних дій, таких як Spike-дослідження, Proof of Concept або зміни у процесах. Він збирає дані про фактичні витрати часу, ресурсів, статуси виконання та проміжні результати. Ці дані передаються до модуля оцінки результатів і контролю зниження невизначеності, який застосовує статистичний аналіз, порівняння до- і після дій та розрахунок індексів Uncertainty Index і Risk Δ. Результатом є кількісна оцінка ефективності кожної гіпотези.

Модуль адаптації плану управління використовує ці оцінки для автоматичного рефакторингу плану проєкту [7].

На вході він отримує підтверджені результати та оновлює backlog, ролі, пріоритети й календарні плани, синхронізуючи зміни з Jira та іншими системами. Модуль циклічності та повторного аналізу фіксує ці оновлення у проєктній базі даних і ініціює новий цикл управління, забезпечуючи безперервність процесу.

Аналітична панель керівника агрегує вихідні дані всіх модулів і відображає їх у вигляді дашбордів, які можуть містити: динаміку невизначеності, вплив управлінських дій і прогноз виконання проєкту. Тобто, набір даних, які будуть виведені на дашборді керівник проєкту обирає самостійно.

У підсумку, вся ІС надає менеджеру не лише статуси та плани, а й аналітично обґрунтовані рекомендації щодо зниження невизначеності. На

відміну від традиційних систем управління «звичайними» ІТ-проєктами, які фокусуються на контролі строків і задач, запропонована ІС працює з невизначеністю як керованою величиною та забезпечує адаптивність, прогнозованість і стійкість управління складними ІТ-проєктами.

4.2. Розробка синтетичного датасету для перевірки запропонованих підходів управління складним ІТ-проєктом в умовах невизначеності

4.2.1 Мета та задачі формування синтетичного дата сету

Формування навчального та демонстраційного датасету для системи управління складними ІТ-проєктами спрямоване не на відтворення реальних корпоративних даних, а на перевірку працездатності запропонованого управлінського механізму [8].

Для цієї розробки не потрібен класичний ML-датасет, орієнтований на навчання моделей за історичними прикладами. Натомість буде застосовано синтетичний, керовано-реалістичний датасет, який відтворює поведінку складного ІТ-проєкту в часі з урахуванням невизначеностей, наукових досліджень і практичних перевірок управлінських гіпотез.

Принцип формування такого датасету в тому, що він має репрезентувати не окремі записи даних, а цілісну історію проєкту. Розробка синтетичного датасету грунтується на викоритсанні засобів глибокого навчання. Один датасет відповідає одному проєкту, який розвивається в часі як послідовність станів. Кожен стан включає події, артефакти, управлінські рішення та наслідки від їх реалізації. Таким чином моделюється повний цикл управління складністю в умовах невизначеності:

- спостереження стану проєкту;
- виявлення невизначеностей;
- формування гіпотез;
- виконання дій для їх перевірки;
- аналіз результатів;

– перехід до нового стану проєкту або оновленого плану виконання проєктних робіт.

4.2.2 Формування базового шару даних навчального дата сету

Для формування навчального та демонстраційного датасету, необхідно визначити базовий шар даних спостережень, який імітує роботу реальних інструментів розробки та управління IT-проєктами.

Для подальшого моделювання фіксуємо проєкт-приклад: розробка складної корпоративної платформи BankDataHub управління цифровими даними для банківської групи, що поєднує інтеграцію даних, аналітичні сервіси, регуляторну відповідність і R&D-компоненти.

Для проєкту обрано такий перелік типових джерел:

– Jira (issues, epics, статуси, зміни scope);
– Git (commits, churn, frequency);
– CI/CD (build failures, test coverage);
– QA (defects, severity);
– Комунікації (Confluence, Slack — агреговано).

Саме для такого типу проєкту характерні високий рівень невизначеності, активна зміна вимог, дослідницькі гіпотези та тісна взаємодія між інженерними, аналітичними й управлінськими процесами, тому обраний набір джерел даних є репрезентативним.

Jira використовується як основне джерело управлінських і процесних подій. Дані з Jira відображають стан робіт над проєктом у вигляді задач, епіків, змін статусів і коригувань обсягу робіт. Сенс цих даних полягає у фіксації того, як проєкт планується та виконується. Наприклад, issue може переходити зі статусу “In Progress” у “Blocked”, що сигналізує про управлінську проблему або технічну невизначеність. Приклад подієвого запису для системи управління може виглядати наступним чином (лістинг 4.1). Тут і далі, записи подано в нотаціях JSON (JavaScript Object Notation).

Лістинг 4.1 – Приклад подієвого запису для Jira

{"source":"jira", "event_type":"status_change", "issue_id":"BDH-142", "from":"In Progress", "to":"Blocked", "timestamp":"2026-03-14T10:32:00Z", "epic":"Data Ingestion", "assignee":"team_backend"}

Git слугує джерелом технічних спостережень щодо інтенсивності та стабільності розробки. Commit означає зафіксовану зміну в коді, churn відображає обсяг змінених рядків коду за певний період, а frequency показує частоту комітів як індикатор активності команди.

Наприклад, різке зростання churn може свідчити про архітектурні експерименти або переробку рішень. Приклад подієвого JSON-запису подано у лістингу 4.2.

Лістинг 4.2 – Приклад подієвого запису для Git

{"source":"git", "event_type":"commit", "commit_id":"a9f3c21", "author":"dev_analytics", "lines_added":420, "lines_deleted":310, "churn":730, "timestamp":"2026-03-15T18:05:00Z", "module":"Data Pipeline"}

CI/CD надає дані про стабільність процесу постачання продукту. Тут фіксуються збірки, збої, тривалість виконання та покриття тестами. Наприклад, серія невдалих білдів може сигналізувати про технічний ризик або поспішне впровадження гіпотези. Запис для системи управління подано у лістингу 4.3.

Лістинг 4.3 – Приклад подієвого запису для CI/CD

{"source":"cicd", "event_type":"build_result", "pipeline":"backend-main", "status":"failed", "test_coverage":62, "duration_sec":540, "timestamp":"2026-03-16T02:11:00Z"}

QA є джерелом інформації про якість продукту та наслідки управлінських рішень. Дані включають дефекти, їх серйозність і статуси виправлення. Наприклад, поява дефектів з критичною важливістю після експериментальної зміни архітектури може підтверджувати або спростовувати гіпотезу. Приклад запису подано у лістингу 4.4.

Лістинг 4.4 – Приклад подієвого запису для QA

{"source":"qa", "event_type":"defect_reported", "defect_id":"DEF-77", "severity":"Critical", "component":"Access Control", "timestamp":"2026-03-17T09:48:00Z", "related_hypothesis":"HYP-12"}

Комунікаційні інструменти, такі як Confluence і Slack, агрегуються у вигляді узагальнених параметрів без аналізу персональних повідомлень. Їх сенс полягає у фіксації інтенсивності обговорень, кількості питань або рішень щодо конкретних тем. Наприклад, зростання кількості повідомлень навколо одного модуля може свідчити про приховану складність. Подієвий запис може мати наступний вигляд (лістинг 4.5).

Лістинг 4.5 – Приклад подієвого запису для Confluence і Slack

{"source":"communication", "event_type":"discussion_spike", "channel":"#data-security", "messages_count":48, "timestamp":"2026-03-18T15:00:00Z"}

Усі ці події формують базовий шар датасету у форматі SON, тобто Stream-Oriented Narrative. SON означає потокове представлення історії проєкту як впорядкованої у часі послідовності подій, де кожен запис несе семантичний контекст управління. Саме такий формат дозволяє системі управління складними IT-проєктами аналізувати динаміку, виявляти невизначеності та відтворювати логіку прийняття управлінських рішень у часі.

4.2.3 Формування шару латентних даних невизначеностей

Формування синтетичного датасету продовжено з побудовою шару латентних даних невизначеностей, які не спостерігаються напряму в інструментах розробки, але є ключовими для управління складними IT-проєктами. Це не «сирі» події, а інтерпретаційні мітки, які вводяться як навчальні анотації та відображають прихований стан проєкту.

Такі дані не надходять безпосередньо з Jira чи Git, а отримуються шляхом аналітичної обробки, агрегації та переосмислення даних спостережень, сформованих на попередньому кроці. Фактично система або дослідник виконує аналіз патернів: зіставляє затримки задач, часті зміни

вимог, аномальний code churn, зростання дефектів, нестабільність комунікацій і на цій основі робить висновок про наявність певної невизначеності.

Перелік «даних невизначеності» формується не довільно, а як узагальнення ключових джерел складності, які присутні у дослідженнях в складних IT-проєктів і практиці R&D-орієнтованої розробки [9, 10]. Категорії неповних вимог, нової технології, нестабільної команди, архітектурної невизначеності та регуляторних змін покривають різні площини проєкту: зміст, технологію, команду IT-проєкту, системну структуру та зовнішнє середовище.

Цей перелік не є «абсолютно повним» у філософському сенсі, але є достатньо повним у керованому сенсі, оскільки будь-яка інша невизначеність у складному IT-проєкті, як правило, редукується до однієї з цих категорій або їх комбінації [11, 12].

Критерієм повноти тут виступає здатність системи пояснити більшість проблемних спостережень через наявні типи невизначеностей без введення надмірних, слабо відмінних класів. В [13] авторка пропонує такий набір чинників невизначеності: 1) технологічна волатильність: вихід нової критичної версії фреймворку/бібліотеки; 2) безпекова невизначеність: несподіване виявлення нової кібербезпекової загрози (наприклад, Zero-Day вразливості) у використовуваних компонентах; 3) нестабільність вимог: висока динаміка часто призводить до неузгодженості вимог користувачів. Авторка пропонує виконувати ідентифікацію методами PESTLE-аналізу (для зовнішніх факторів) та аналізу стейкхолдерів (для внутрішніх чинників),

Неповні вимоги проявляються, наприклад, у вигляді частих змін user stories, повторного відкриття задач або зростання кількості уточнюючих питань у коментарях. Як це може бути зафіксовано у JSON-мітках подано у лістингу 4.6.

Лістинг 4.6 – Приклад фіксації змін User Stories та повторного відкриття задач

```
{
  "uncertainty_type": "incomplete_requirements",
  "confidence": 0.82,
  "evidence": ["JIRA-134 updated 5 times", "scope_change_frequency_high"]
}
```

Нова технологія означає використання інструментів або фреймворків, з якими команда раніше не працювала, що створює певну категорію невизначеності та відображається у Spike-задачах, нестабільних білдах і високому churn. Приклад анотації подано у лістингу 4.7.

Лістинг 4.7 – Приклад анотації

```
{
  "uncertainty_type": "new_technology",
  "confidence": 0.74,
  "evidence": ["new_stack_introduced", "build_failures_growth"]
}
```

Нестабільна команда фіксується через ротацію учасників, нерівномірну активність у Git і зниження швидкості виконання спринтів, що може бути представлено у лістингу 4.8.

Лістинг 4.8 – Приклад зниження швидкості виконання спринтів

```
{
  "uncertainty_type": "team_instability",
  "confidence": 0.69,
  "evidence": ["developer_turnover", "velocity_drop"]
}
```

Архітектурна невизначеність проявляється у частих рефакторингах, зміні ключових компонентів і зростанні технічного боргу (лістинг 4.9).

Лістинг 4.9 – Приклад зниження швидкості виконання спринтів

```
{
  "uncertainty_type": "architectural_uncertainty",
  "confidence": 0.88,
  "evidence": ["core_module_refactoring", "high_dependency_changes"]
}
```

Регуляторні зміни характерні для доменів з жорсткими вимогами та відображаються через появу нових compliance-задач або зміну політик безпеки, що може бути зафіксовано наступним чином (лістинг 4.10):

Лістинг 4.10 – Приклад прояву регуляторних змін

```
{
  "uncertainty_type": "regulatory_change",
  "confidence": 0.91,
  "evidence": ["new_compliance_task", "policy_update_required"]
}
```

У сукупності ці анотації формують латентний шар датасету, який перетворює потік подій у керовану «історію невизначеностей» і робить можливим навчання та демонстрацію роботи системи управління складними ІТ-проєктами.

4.2.4 Застосування сценарного моделювання для валідації розроблених підходів з управління складними ІТ-проєктами

Сценарне моделювання — це підхід до побудови описових моделей майбутніх або альтернативних станів системи, що дозволяє не лише відтворювати поточні та історичні траєкторії, а й прогнозувати можливі варіанти розвитку складних управлінських ситуацій. Саме воно дає змогу описати простори можливих подій, оцінити вплив невизначеностей і сформувати адекватні управлінські рішення у відповідь на зміни, які можуть виникати у проєктних процесах з високою складністю та ризиками. Сценарне моделювання не націлене на збирання “істинних” даних про реальні ІТ-проєкти. Воно призначене для створення реалістичних та керованих випадків (що називають сценаріями), які імітують поведінку проєкту, включно з подіями, рішеннями та наслідками у часі від управлінських дій.

Літературні джерела 14, 15] підтверджують цінність такого підходу у сфері управління складними системами. Наприклад, в [14] розглянуто техніки та процеси створення сценаріїв для управління комплексними системами на основі великих даних. Автори пропонують структуризацію, формування ключових сценаріїв та їх перевірку для менеджменту складних систем.

Інше дослідження [15] поєднує агентне моделювання (Agent-Based Modeling) із сценарним аналізом для відтворення поведінки корпоративних процесів з високим степенем невизначеності.

Для формування кожного сценарію у синтетичному датасеті необхідно задати початковий стан проєкту. Це може бути набір вихідних параметрів: кількість задач у беклозі, рівень ризиків, структура команди, статуси епіків, поточні метрики якості, технічний борг тощо. Початковий стан формується як фіксована конфігурація таких параметрів, що відображає реалістичну ситуацію на старті циклу моделювання.

Наступним завданням є генерація подій у часі, де описані послідовні зміни стану проєкту. Це можуть бути затримки виконання задач, нові вимоги, зміни в складі команди, знайдені дефекти, результат інтеграції певного функціоналу тощо. Події створюються згідно з моделлю часової динаміки: це можуть бути випадкові процеси (імітовані випадковими величинами), правила залежностей (наприклад, що після дефекту йде спад продуктивності команд), або дані з історій інших проєктів.

Прихована невизначеність вводиться як latent variables, що не спостерігається безпосередньо, але впливає на поведінку системи. Це може бути рівень технічної невизначеності у вимогах, невідомі ризики зовнішніх залежностей або внутрішні інтеракції в команді. Для додавання такої невизначеності потрібно створювати символічні маркери або latent features, які моделюють ці фактори у сценаріях і потім виступають як мітки у навчальному наборі. Внутрішні інтеракції (interaction) в команді — це сукупність неформалізованих, частково прихованих процесів взаємодії між членами команди, які впливають на хід проєкту, але не спостерігаються напряму через стандартні інструменти розробки (Jira, Git, CI/CD). Йдеться не про ролі чи структуру команди як таку, а про динаміку взаємодії між людьми.

Задання “правильної” чи “помилкової” управлінської реакції означає визначити, які дії менеджер може виконати у відповідь на подію, та позначити їх як успішні чи неуспішні з точки зору зміни стану проєкту. Це може бути, наприклад, рішення перенести задачу у беклог, змінити пріоритет, провести Spike, дослідження, навчання, додаткове тестування або змінити команду. Такі

реакції додаються до сценарію як керовані змінні, що впливають на подальшу еволюцію стану.

Наслідок — це стан проєкту після застосованої дії, який відображає нові значення ключових показників (невизначеність, ризики, якість, терміни). Його фіксація дає можливість оцінити ефективність дії у кожному сценарії та використовувати дані для навчання моделей системи управління складними ІТ-проєктами, що розробляється.

Запропонований «сценарний підхід» реалізовано у вигляді алгоритму формування сценарію не як "жорсткий код", а як формалізована процедура: алгоритм сценарної генерації та прогону. В рамках дисертаційній логіці його подано у вигляді формального опису кроків із входами, станами та виходами. Зазначимо, що алгоритм не пишеться під кожну проєктну ситуацію з нуля. В ІС буде існувати один базовий сценарний алгоритм, який завантажується та працює автоматично, але він буде параметризований, тобто змінюються лише його параметри: тип проєкту, типи невизначеностей, початковий стан, правила реакцій, часові горизонти. У термінах інженерії це meta-algorithm, або шаблон сценарного прогону. Таким чином, для різних проєктів і ситуацій не створюються нові алгоритми, а створюються нові сценарні конфігурації, які "підставляються" у той самий механізм.

Сам алгоритм сценарного моделювання не є відповідальністю розробників з команди ІТ-проєкту. Його формує або методологічна група, або Chief Architect / Head of PMO / R&D Lead, часто спільно з досвідченими проєктними менеджерами. На практиці це відбувається як формалізація управлінського досвіду: визначаються типові класи невизначеностей, типові управлінські реакції, очікувані ефекти та лаги у часі. Після цього алгоритм реалізується один раз у системі, а команда проєкту надалі користується ним впродовж життєвого циклу проєкту, який вона розробляє.

Алгоритм сценарного моделювання працюватиме з трьома типами вхідних даних. Перший тип автоматично зібрані дані з проєкту: події Jira, метрики Git, CI/CD, QA, історія змін плану. Це базовий шар, який надходить у

систему без участі людини. Другий — аналітичні або експертні дані, які неможливо надійно витягнути автоматично: оцінка новизни технології, рівень зрілості команди, ступінь архітектурної ясності, регуляторні ризики. Такі дані вводяться вручну через інтерфейс, але не як “текст”, а як структуровані параметри або шкали. Третій тип — зовнішні або напівструктуровані дані, які система може парсити за потреби: документація, вимоги, технічні RFC, результати PoC, звіти Spike-досліджень. Саме поєднання цих трьох джерел дозволяє алгоритму працювати адекватно, а не формально.

Алгоритм сформулювано у вигляді формального опису “Scenario Execution Engine”. Це строгий, відтворюваний опис того, як саме система програє сценарій розвитку проєкту в часі, які дані вона приймає на вході, які стани змінює, за якими правилами генерує події, вводить невизначеності, застосовує управлінські дії та фіксує наслідки. Ця алгоритмічна специфікація зрозуміла і досліднику, і розробнику, і експерту з управління. В науковому сенсі це відповідь на питання: *«За якою процедурою ми моделюємо поведінку складного IT-проєкту?». В* системному сенсі — це “двигун” сценарного моделювання, який забезпечує повторюваність експериментів.

Формальний опис зазвичай включає:

– множину станів проєкту;

– множину подій;

– множину прихованих змінних (невизначеностей);

– правила переходів між станами;

– управлінські дії;

– функцію оцінки результату.

Короткий приклад формального опису у вигляді псевдокоду подано у лістингу 4.11.

Лістинг 4.11 – Алгоритм сценарного моделювання

```
ScenarioExecutionEngine (S0, U, R, T):
  Input:
    S0 – початковий стан проєкту
    U – множина latent-невизначеностей
    R – набір управлінських правил (reactions)
```

```
    T – часовий горизонт сценарію
S ← S0
for t = 1 to T:
    E_t ← GenerateEvents(S, U, t)
    S ← UpdateState(S, E_t)
    if DetectUncertainty(S, U):
        H ← FormHypothesis(S, U)
        A ← SelectAction(H, R)
        S ← ApplyAction(S, A)
    S ← PropagateEffects(S, U)
Outcome ← EvaluateOutcome(S)
return Outcome, Trajectory(S)
```

У цьому описі S — це вектор стану проєкту (терміни, якість, стабільність команди, ризики), U — приховані фактори складності, E(t) — події, які система “спостерігає”, A — управлінська дія (Spike, PoC, рефакторинг плану), а Outcome — зафіксований результат сценарію.

Цей формальний опис — це керований експеримент над проєктом, а не довільна історія. Тобто, Scenario Execution Engine науково коректна процедура для генерації синтетичного датасету і для демонстрації працездатності всієї системи управління складними IT-проєктами.

4.2.5 Розробка генератора синтетичного датасету

Генератор синтетичного датасету для системи управління складними IT-проєктами доцільно формалізувати як гібридну модель, що поєднує rule-based механізми з додатками дослідницьких подій типу R&D [16].

Такий генератор не відтворює випадкові дані, а моделює причинно-наслідкову динаміку проєкту в часі, з урахуванням управлінських рішень і прихованих факторів складності. На формальному рівні генератор визначається як функція *G*:

$$G: (S(t), U(t), A(t)) \rightarrow S_{t+1},$$

де $S(t)$ – вектор спостережуваного стану проєкту в момент часу t;

$U(t)$ – вектор прихованих невизначеностей;

$A(t)$ — управлінські дії або дослідницькі втручання.

Часовий механізм генерації задається дискретною ітераційною моделлю, у якій стан оновлюється за рекурентним правилом

$$S_{t+1} = S(t) + \Delta S(t),$$

де $\Delta S(t)$ – сукупний ефект подій, управлінських дій і латентних факторів.

Наприклад, фактичні витрати часу можуть моделюватися як

$$H_{t+1} = H(t) + \alpha \cdot C(t) + \beta \cdot U(t) + \varepsilon(t),$$

де $H(t)$ — відпрацьовані години;
$C(t)$ — складність поточних задач;
$U(t)$ — рівень технічної або організаційної невизначеності;
α і β — коефіцієнти впливу;
$\varepsilon(t)$ — стохастичне збурення, $\varepsilon(t) \sim N(0,\sigma^2)$.

Стохастичний компонент вводиться для уникнення детермінізму та моделює непередбачувані флуктуації реального проєкту, такі як людський фактор або зовнішні залежності.

Причинно-наслідкові правила формалізуються у вигляді умовних залежностей типу

$$if\ U(t) = \text{“new\_technology”} \rightarrow Var(H(t)) \uparrow,\ DefectRate(t) \uparrow,\ BuildFailure(t) \uparrow,$$

які реалізуються через зміну параметрів розподілів. Наприклад, кількість дефектів $D(t)$ може обчислюватися як

$$D(t) \sim Poisson(\lambda(t)), \quad \lambda(t) = \lambda(0) \cdot (1 + \gamma \cdot U(t)),$$

де $\lambda(0)$ — базова інтенсивність дефектів;

γ — коефіцієнт впливу невизначеності.

Спеціальні дослідницькі події моделюються як керовані події R&D, що тимчасово змінюють динаміку системи. Для Spike-дослідження або PoC вводиться подія $A(t)$ з ефектом

$$U_{t+k} = U(t) \cdot (1 - \delta),$$

де $\delta \in (0,1)$ — коефіцієнт зниження невизначеності після дослідження;

k — затримка ефекту.

Аналогічно, навчання команди може зменшувати дефектність і варіативність продуктивності через корекцію параметрів γ або σ.

Таким чином, формальна модель генератора визначається системою рекурентних рівнянь, умовних правил і стохастичних компонентів, що дозволяє програмно відтворювати різні траєкторії розвитку складного IT-проєкту. Результатом роботи генератора є послідовність станів $S(0)…S(T)$ разом із мітками прихованих невизначеностей і управлінських рішень, що утворює повноцінний синтетичний датасет для навчання й демонстрації працездатності системи управління.

4.3. Оцінка ефективності інформаційної системи управління складним IT-проєктом

4.3.1 Опис працездатності інформаційної системи

Підтвердження працездатності інформаційної системи управління складними IT-проєктами в дисертаційній роботі пропонується представити не у вигляді формального числового порівняння або аналітичного доведення ефективності, а як керований демонстраційний експеримент, оформлений у

вигляді наскрізного процесу для одного складного ІТ-проєкту. Як і в підрозділі 4.2, оцінка ефективності розробленої ІС спирається на застосуванні інструментів глибокого навчання. Такий підхід дозволяє показати систему, яка спирається на розроблені в роботі моделі та методи управління складними ІТ-проєктами, як цілісний управлінський механізм, що функціонує в реалістичному середовищі та працює з типовими для ІТ-проєктів даними і рішеннями.

Основою демонстрації є синтетичний, але керовано-реалістичний датасет, який відтворює історію проєкту в часі, включаючи події, управлінські рішення, реакції команди та наслідки цих рішень, структура якого була розроблена в підрозділі 4.2.

Підтвердження працездатності здійснюється через покроковий опис повного управлінського циклу, починаючи з фіксації вихідного стану проєкту, його початкових пріоритетів, плану та невизначеностей, і завершуючи адаптованим планом управління після застосування системи. У межах цього сценарію демонструється, як система централізовано збирає та консолідує дані з інструментів розробки, як на їх основі ідентифікуються зони невизначеності та приховані проблеми, як формується аналітичне розуміння їхнього впливу на проєкт, і як на цій основі генеруються, оцінюються та реалізуються управлінські гіпотези.

Підтвердженням працездатності є показ того, що система не лише фіксує проблеми, а породжує конкретні управлінські рішення: гіпотези, PoC, Spike-дослідження, рішення щодо навчання команди або зміни процесів і дозволяє відстежити наслідки їх реалізації. Фінальним результатом демонстрації є оновлений план управління зі зміненими пріоритетами та зниженим рівнем невизначеності, що слугує об’єктивним свідченням того, що проєкт перейшов у більш керований та прогнозований стан. Такий формат подачі матеріалу дозволяє наочно і логічно довести практичну працездатність запропонованої та розробленої системи та її доцільність для управління складними ІТ-проєктами.

4.3.2 Початковий план та пріоритети проєкту

Початковий план та пріоритети IT-проєкту формуються (як це було зазначено в 4.2.2) для складного корпоративного IT-проєкту розробки платформи обробки регуляторних і операційних даних для банківської установи, який розглядався в дисертації як наскрізний приклад складного IT-проєкту. Він спрямований на створення єдиного цифрового середовища для збору, нормалізації, зберігання та аналізу регуляторних даних, що надходять з різних внутрішніх і зовнішніх джерел, із забезпеченням відповідності вимогам фінансового нагляду та інформаційної безпеки. На старті проєкт характеризується високим рівнем складності через поєднання нових технологій обробки даних, жорстких регуляторних обмежень і розподіленої міжфункціональної команди.

Проєкт BankDataHub передбачає розробку корпоративної платформи обробки та управління регуляторними даними для банківської установи. Дата старту проєкту встановлена на 01.02.2025, прогнозована тривалість становить 9 місяців, із плановою датою завершення 30.10.2025. Проєкт реалізується із застосуванням фреймворку Scrum з фіксованою тривалістю спринту 2 тижні, що формує орієнтовно 18 спринтів у межах усього життєвого циклу розробки.

У проєкті задіяно 3 кросфункціональні команди, загальною чисельністю 21 учасник. Перша команда зосереджена на ядрових сервісах обробки та трансформації даних і розташована в основному офісі розробника з Data-інженерії. Друга команда відповідає за інтеграцію з банківськими інформаційними системами, безпеку та відповідність регуляторним вимогам і працює у гібридному форматі разом із технічними представниками банку. Третя команда розробляє аналітичні модулі, звітність і аудит та частково розміщена віддалено, що дозволяє залучити фахівців із вузькою кваліфікацією без прив’язки до локації.

Кожна команда має склад, близький до рекомендованого Scrum-діапазону, але більша ніж мінімально-життєздатна команда [17] що забезпечує достатню функціональну автономність та складається з 1 Scrum Master, 1

Product Owner (спільний на проєкт), 4 розробників і 1 QA-інженера. Компетентнісний склад команд включає 6 senior-інженерів (≈8–12 років досвіду), 9 middle-інженерів (≈3–5 років досвіду) та 3 junior-спеціалісти (до 2 років досвіду). Основні ролі зосереджені у сферах backend-розробки, data-інженерії, кібербезпеки та інтеграційних рішень.

Прогнозована вартість проєкту орієнтовно 1,8 млн євро, з яких близько 70 % припадає на витрати на персонал, 20 % — на інфраструктуру та ліцензії, 10 % — на супровід, аудит і регуляторну відповідність. Початковий беклог формується з приблизно 160 user stories, згрупованих у 7 функціональних інкрементів:

- модуль інтеграції з джерелами даних;
- модуль трансформації та нормалізації даних;
- модуль управління метаданими;
- модуль контролю доступу та ролей;
- модуль регуляторної звітності;
- модуль аудиту та журналювання, аналітичний модуль.

Архітектура платформи визначена як мікросервісна, із застосуванням контейнеризації Docker та оркестрації Kubernetes, з асинхронною взаємодією сервісів через Apache Kafka. Основною мовою програмування обрано Java (Spring Boot) для backend-компонентів та Python для сервісів обробки даних. Система буде розгорнута в хмарному середовищі банку на PostgreSQL та object-storage.

Проєкт передбачає інтеграцію з чинними банківськими системами: Core Banking System, CRM-платформою, сховищем даних (Data Warehouse), системами AML/KYC, а також зовнішніми регуляторними шлюзами для передачі звітності. Головні проєктні показники зібрані в таблиці 4.1.

Під пріоритетами в контексті управління складним IT-проєктом будемо розуміти ієрархію управлінських фокусів, яка визначає, на що саме менеджер готовий витрачати обмежені ресурси в ситуації невизначеності. Вони задають логіку прийняття рішень ще до того, як з’являться проблеми.

Формально пріоритети можна розглядати як ваги цілей управління, які впливають на всі подальші рішення.

Таблиця 4.1 – Початковий стан IT-проєкту

Показник	Значення	Коментар
Тип проєкту	Платформа обробки регуляторних даних для банку	Інтеграційна, data-інтенсивна система
Рівень технологічної новизни	Високий	Новий стек обробки даних + регуляторні вимоги
Ступінь визначеності вимог	Низький	Частина вимог декларативні, без технічної деталізації
Регуляторна складність	Висока	GDPR, локальні банківські нормативи, аудит
Склад команди	Розподілена, 3 локації	Обмежений досвід у ключовій технології
Початковий план	Фіксований roadmap на 6 місяців	Без закладених дослідницьких ітерацій
Початкові пріоритети	Функціональність > швидкість	Фокус на delivery, не на зниження невизначеності
Допустимий рівень ризику	Неформалізований	Відсутні явні критерії прийнятного ризику

Наприклад, у початковому стані проєкту можуть бути зафіксовані такі управлінські пріоритети:

– дотримання строків важливіше за оптимальність архітектури;

– відповідність регуляторним вимогам важливіша за швидкість розробки;

– стабільність команди важливіша за агресивне масштабування функціоналу.

Це не задачі, а рамка, у межах якої оцінюються всі управлінські дії. З урахуванням цього, початкові пріоритети проєкту створення BankDataHub направлені на своєчасний запуск базових функцій регуляторної звітності та забезпеченні відповідності вимогам GDPR, ISO/IEC 27001 і місцевих нормативів, при цьому дослідницькі активності, PoC та спеціалізовані навчальні заходи на старті формально не виділені в окремі спринти, що створює потенційні ризики при виникненні технологічної або регуляторної невизначеності.

4.3.3 Централізований збір та консолідація даних проекту

У цьому пункті наведено приклади даних, що надходять до системи для формування централізованої проєктної бази спостережень. Усі приклади це подієві записи із часовою прив'язкою, що відображають динаміку проєкту в березні 2025 року.

Дані з *Jira* демонструють відхилення між плановими та фактичними оцінками робіт. Наприклад, задача *US-101* у модулі *DataIngest* була оцінена у *5 story points*, але фактично потребувала 8, що зафіксовано *03.03.* Інший запис показує, що задача *US-102* у модулі *AccessControl* мала планову тривалість *6 днів,* але виконувалася *14 днів (05.03),* що свідчить про суттєве відхилення від початкової оцінки. Також *10.03* зафіксовано зміну призначення задачі *US-110* зі *Sprint-2 на Sprint-4*, що вказує на зсув строків реалізації функціоналу.

Дані з *Git* відображають інженерну активність і стабільність коду. *05.03 commit-884* у модулі *AccessControl* містив *540 змінених рядків* при плановому очікуванні *120*, що сигналізує про високий *churn* і можливу архітектурну нестабільність. *12.03* для *commit-902* зафіксовано *частоту 19 комітів* за тиждень при *планових 4,* що може свідчити про кризовий стан доопрацювань.

Дані *CI/CD* демонструють технічну нестабільність. *15.03 build-221* у модулі *AccessControl* завершився зі статусом *fail* замість запланованого *success,* що є прямим сигналом технологічного ризику.

Дані *QA* відображають проблеми якості. *18.03 bug-17* у модулі *Reporting* був класифікований як *critical* при плановому рівні *medium,* що вказує на недооцінку складності реалізації.

Усі ці записи консолідуються в єдину проєктну базу спостережень, представлену у таблиці 4.2, та використовуються як вхідні дані для наступного етапу — ідентифікації зон невизначеності та складності.

Усі ці події консолідуються у спільному часовому просторі, зберігаються у проєктній базі спостережень, приклад якої поданий у Додатку В, та стають вхідними даними для наступного кроку — ідентифікації зон невизначеності та складності. Приклад подано у форматі JSON, оскільки він

наочно відображає документно-подієву природу такої бази. Початкові пріоритети IT-проєкту визначають, на які аспекти команда та керівництво орієнтуються в першу чергу під час прийняття управлінських рішень.

Таблиця 4.2 – Фрагмент Проєктної бази спостережень

timestamp	source	entity_id	metric	planned	actual	module
03.03	Jira	US-101	story_points	5	8	DataIngest
05.03	Jira	US-102	duration_days	6	14	AccessControl
05.03	Git	commit-884	churn_lines	120	540	AccessControl
06.03	CI/CD	build-221	build_status	success	fail	AccessControl
08.03	QA	bug-17	severity	medium	critical	Reporting
10.03	Jira	US-110	sprint_assignment	Sprint-2	Sprint-4	DataIngest
12.03	Git	commit-902	frequency_week	4	19	AccessControl
15.03	CI/CD	build-233	test_coverage	82%	61%	AccessControl
18.03	QA	bug-29	defect_count_sprint	5	13	AccessControl

Головним пріоритетом є забезпечення регуляторної відповідності платформи, оскільки будь-які відхилення від вимог наглядових органів можуть призвести до критичних ризиків для банку. Другим за важливістю є стабільність та безпека архітектури обробки даних, включаючи контроль доступу, аудит операцій та захист конфіденційної інформації. Наступним пріоритетом виступає дотримання базових строків реалізації ключових функцій, необхідних для запуску мінімально життєздатної версії платформи. Паралельно фіксується пріоритет ефективного використання ресурсів команди, оскільки частина спеціалістів залучена до інших ініціатив банку.

Таким чином, початковий план і пріоритети відображають бачення проєкту як відносно передбачуваного процесу з чітко визначеними цілями, строками та обмеженнями. Водночас ці пріоритети ще не враховують повною мірою приховані невизначеності, пов’язані з новизною технологій, еволюцією регуляторних вимог і внутрішніми процесами взаємодії команди, що згодом стає підставою для застосування адаптивної системи управління складним IT-проєктом.

4.3.4 Ідентифікація зон невизначеності та складності

Під аномаліями у межах методу розуміються конкретні зафіксовані відхилення показників із проєктної бази спостережень від планових або очікуваних значень. Виявник відхилень працює безпосередньо з полями planned та actual. Наприклад, для задачі US-101 (03.03) планова оцінка становила 5 story points, фактична — 8, що означає перевищення на 60 %. Для задачі US-102 (05.03) тривалість виконання становила 14 днів замість запланованих 6, тобто перевищення більш ніж у два рази. У модулі AccessControl зафіксовано churn_lines 540 при планових 120, а також build_status = fail при плані success. Такі записи формують таблицю 4.3 «Аномалії», де кожен рядок є конкретним сигналом від виявника.

Таблиця 4.3 – Аномалії в стані ІТ-проєкту

ID	Тип сигналу	Об'єкт	Пояснення
A1	Перевищення оцінки	US-101	+60% до планових story points
A2	Затримка	US-102	duration ×2.3
A3	Code churn spike	commit-884	540 рядків при нормі 150
A4	Build instability	build-221	3 fail поспіль
A5	Coverage drop	build-233	-21% покриття
A6	Defect spike	bug-29	13 дефектів замість 5
A7	Commit burst	commit-902	19 комітів за тиждень
A8	Severity jump	bug-17	medium → critical

Невідповідність плану і факту формується як систематизоване зіставлення атрибутів planned та actual. Наприклад, для US-110 (10.03) планом передбачалося виконання у Sprint-2, фактично задача перенесена у Sprint-4. Для bug-17 (08.03) очікувана серйозність medium, фактична — critical. В таблиці 4.4 ці записи вже подані як формалізовані розбіжності, де кожна невідповідність має чіткий числовий або категоріальний характер.

Нестабільні ділянки визначаються агреговано на основі концентрації аномалій у межах одного модуля або періоду часу. У нашому випадку модуль AccessControl має одразу кілька відхилень у проміжку 03.03–12.03:

перевищення тривалості, аномальний churn, збій build та різке зростання frequency_week з 4 до 19.

Таблиця 4.4 – Невідповідність плану і факту

ID	Об’єкт	План	Факт	Відхилення
P1	US-110	Sprint-2	Sprint-4	+2 спринти
P2	US-102	6 днів	14 днів	+133%
P3	Coverage	80%	61%	-19%
P4	Defects	≤5	13	+160%
P5	Release date	30.06	прогноз 15.07	+15 днів
P6	Integration phase	2 спринти	4	×2
P7	Velocity	40 SP	26 SP	-35%
P8	Module stability	стабільний	4 інциденти	нестабільність

В таблиці 4.5 це відображено як агрегований індикатор нестабільності модуля, що базується на кількості зафіксованих відхилень у часовому інтервалі.

Таблиця 4.5 – Нестабільні ділянки (агрегований аналіз)

Module	Кількість аномалій	Build fails	Defects	ΔPlan
AccessControl	6	3	8	високий
DataIngest	2	0	1	середній
Reporting	1	0	2	низький

Система не приймає рішень на цьому етапі, а формує структурований набір сигналів для наступного переходу до аналізу прихованих причин. Висновок, який можна зробити на даному етапі: Модуль AccessControl є зоною підвищеної складності. Показниками цього є: нестабільні білди, різкий churn, зниження test coverage, затримки задач.

4.3.5 Інтерпретація аномалій та формування управлінських гіпотез

На основі таблиць 4.1 – 4.5 система переходить від фіксації симптомів до формування інтерпретованих управлінських проблем. Цей етап також не генерує рішення. Він встановлює причинно-наслідкові зв’язки між подіями

сценарію та ризиками складності для досягнення початкових пріоритетів проєкту. Логіка у вигляді звичайного rule-based inference була сформульована наступним чином:

1. Вхід: таблиця аномалій.
2. Модуль інтерпретації перевіряє правила типу:

якщо:
churn_lines > threshold
I build_status = fail

І є перенесення спринту → проблема = "архітектурна невизначеність"

У коді це виглядатиме як:

```
if churn > 400 and build_failures > 2 and sprint_shift > 1:
   problem = "Architectural uncertainty"
```

Це не нейромережа, це логічний механізм інтерпретації, який у контексті загального сценарного моделювання відповідає фазі «інтерпретації стану системи» (таблиця 4.6).

Таблиця 4.6 – Інтерпретовані проблеми

№	Сукупність аномалій	Модуль / зона	Інтерпретована проблема	Потенційний вплив
1	churn=540 + build fail + перенесення US-102	AccessControl	Архітектурна невизначеність інтеграції	Ризик зриву релізу R1
2	19 commits/week при плані 4 + frequent rework	AccessControl	Відсутність стабільного технічного рішення	Зростання трудомісткості
3	1 critical defect + reopen задач	Reporting	Неповні або нечіткі регуляторні вимоги	Ризик невідповідності
4	Перенесення Sprint-2 → Sprint-4	DataIngest	Недооцінка складності інтеграції	Порушення графіку
5	Зниження test stability	AccessControl	Низька зрілість модульного тестування	Ризик регресій

Результатом цього кроку є перелік проблем, пов’язаних із початковими пріоритетами проєкту (регуляторна відповідність, архітектурна стабільність, строки).

Оскільки виявлені проблеми не мають наперед заданого алгоритму усунення, то наступним кроком, система формує дослідницькі та управлінські

гіпотези, за допомогою Мапи *“Problem → Hypothesis Template”* логіки (лістингу 4.12):

Лістинг 4.12 – Формування управлінських гіпотез

```
problem_hypothesis_map = {
    "Architectural uncertainty": "Run Spike to validate integration pattern",
    "Regulatory ambiguity": "Conduct compliance workshop",
    "Test instability": "Introduce automated regression framework"
}
```

Тобто система працює як: *Problem detected → lookup → generate hypothesis record*.

Це відповідає логіці складних IT-проєктів, де рішення не завжди детерміноване. В межах сценарного моделювання цей крок відповідає фазі «генерації альтернативних управлінських реакцій». Сформовані гіпотези подані в таблиці 4.7.

Таблиця 4.7 – Управлінські гіпотези

№	Проблема	Гіпотеза	Тип дії	Очікуваний ефект
1	Архітектурна невизначеність	Провести Spike з альтернативною схемою авторизації	Spike	Зниження churn ≥ 30%
2	Нестабільність інтеграції	Реалізувати PoC окремого сервісу адаптера	PoC	Стабілізація build
3	Нечіткі регуляторні вимоги	Провести спільний workshop з комплаєнс-відділом	Організаційна дія	Зменшення reopen
4	Недооцінка складності	Ревізія backlog + повторна оцінка story points	Процесна зміна	Реалістичний roadmap
5	Низька тестова зрілість	Навчання команди unit-testing	Training	Підвищення test coverage

Результатом цього етапу є реалізація сценарного моделювання до рівня «гіпотези», тобто система виконала сценарій: *дані → аномалії → проблеми → гіпотези*, без залучення інтуїтивного управління.

4.3.6 Оцінка та пріоритизація управлінських гіпотез.

Після формування управлінських гіпотез система переходить до наступного кроку «Оцінка та пріоритизація гіпотез». Це аналітичний етап управлінського вибору. Попередні кроки відповідали на питання «що відбувається?» та «чому це проблема?», а тепер система відповідає на питання «що саме робити?». Оцінка дозволяє визначити, чи доцільно реалізовувати гіпотезу, який очікуваний ефект вона дасть, наскільки критичною є проблема, чи дозволяють ресурси виконати втручання та який варіант забезпечить максимальний управлінський результат.

Етап логічно поділено на дві фази. Фаза 1 — аналітична оцінка. У цій фазі кожна гіпотеза отримує формалізовану кількісну оцінку. Фаза 2 — вибір гіпотез, де відбувається ранжування та прийняття рішення Execute / Postpone / Reject.

На цьому етапі сценарне моделювання через оцінювання потенційного впливу гіпотез на параметри системи, які були визначені раніше (динаміка дефектів, нестабільність модулів, регуляторний ризик), визначає наступний вибір. Реалізація сценаріїв відбудеться на наступному кроці, а тут формується основа для цього вибору.

Формалізована модель оцінки сформована як багатокритеріальна зважена сума:

$$Score_i = w_1 \cdot Impact_i + w_2 \cdot RiskReduction_i - w_3 \cdot Cost_i - w_4 \cdot Time_i$$

де $Impact_i$ — очікуваний позитивний вплив на систему;

$RiskReduction_i$ — зменшення прогнозованого ризику;

$Cost_i$ — витрати ресурсів;

$Time_i$ — тривалість реалізації;

$w_1 \ldots w_4$ — вагові коефіцієнти управлінської політики.

Перед розрахунком усі критерії нормалізуються до інтервалу [0;1] методом Min-Max scaling. Реалізація в Python з бібліотеками pandas і numpy подана в Додатку Б.

Результати оцінки подані в таблиці 4.8 яка містить поля: Hypothesis_ID, Problem_ID, Impact, Risk_Reduction, Cost, Time, Score. Вона демонструє числовий результат аналітичної фази:

P1 — Нестабільність модуля AccessControl (високий churn, build failures);

P2 — Невизначеність вимог у DataIngest (перенесення спринтів, зростання story points);

P3 — Критичні дефекти у Reporting.

Для них були сформовані гіпотези:

H1 — Провести Spike по архітектурі AccessControl;

H2 — Провести PoC інтеграції DataIngest;

H3 — Провести навчання команди з регуляторної логіки Reporting

Таблиця 4.8 – Оцінка гіпотез

Hypothesis_ID	Problem_ID	Impact	Risk_Reduction	Cost	Time	Score
H1	P1	0.9	0.8	0.6	0.5	0.47
H2	P2	0.8	0.7	0.7	0.6	0.32
H3	P3	0.6	0.5	0.3	0.4	0.38

Використано модель: $Score_i = 0.4 \cdot Impact + 0.3 \cdot RiskReduction - 0.2 \cdot Cost - 0.1 \cdot Time,$ Усі значення вже нормалізовані до [0;1].

З таблиці видно, що H1 має високий вплив і суттєве зниження ризику, хоча потребує помірних витрат. H2 має сильний вплив, але високу вартість і тривалість, а от H3 має менший стратегічний вплив, але дешева та швидка. Далі гіпотези подані за рангом (таблиця 4.9). Decision визначається пороговим правилом або обмеженням ресурсу. Це і є результат пріоритизації.

Таблиця 4.9 – Ранжовані гіпотези

Rank	Hypothesis_ID	Score	Decision
1	H1	0.47	Execute
2	H3	0.38	Execute
3	H2	0.32	Postpone

Було застосовано порогове правило:

$$\begin{cases} Score \geq 0.35 \rightarrow Execute \\ 0.25 - 0.35 \rightarrow Postpone \\ < 0.25 \rightarrow Reject \end{cases}$$

в результаті H1 та H3 потрапляють до реалізації у поточному циклі, а H2 відкладається через ресурсні обмеження.

4.3.7 План реалізації та оцінка ефективності управлінських гіпотез

На етапі реалізації гіпотез, пріоритизовані гіпотези передаються для перевірки через альтернативні мікро-гілки сценарію. Непріоритетні гіпотези не зникають, а залишаються в backlog адаптивного управління.

При формуванні плану було враховано тип втручання в процеси управління проєктом. H1 це Spike по архітектурі AccessControl, це дослідницька технічна активність, її мета — зменшити технологічну невизначеність і стабілізувати модуль. H3 — Training по регуляторній логіці Reporting. Це організаційне втручання, яке зменшує когнітивну невизначеність команди. Тобто, H1 — технічна (архітектурна), H3 — знаннєва (доменно-регуляторна).

Було запропоновано три можливі сценарії. 1.Незалежна паралельна реалізація. Якщо ресурси різні (архітектори ≠ QA/PO), то H1 і H3 можуть виконуватись в одному спринті паралельно, бо не конкурують за однакових спеціалістів. 2. Послідовна реалізація. Якщо Training залежить від результатів Spike (наприклад, потрібно спочатку уточнити архітектуру, а потім навчати), тоді, гіпотези реалізуються послідоно. 3. Частково паралельна реалізація. Spike стартує на початку спринту, а навчання проводиться у другій половині спринту після первинних висновків. План управління поданий в таблиці 4.10.

Таблиця 4.10 – План реалізації гіпотез

Hypothesis_id	Type	Sprint	Responsible	Dependency	Execution_Order	Status
H1	Spike	5	Solution Architect	–	1	Planned
H3	Training	6	QA Lead + Product Owner	H1	2	Planned
H2	PoC	6	Backend Team Lead	–	1	Postponed

Таблиця 4.10 пов’язана структурно і логічно з усіма попередніми таблицями. Так ідентифікатор гіпотез Hypothesis_ID відповідає таблицям 4.8 – 4.9, що забезпечує трасованість між проблемою, оцінкою та реалізацією. Стовпець Type визначає тип управлінського втручання: Spike це дослідницька технічна перевірка; PoC — підтвердження життєздатності рішення; Training — організаційний або доменний розвиток команди. Стовпець Sprint це номер запланованого спринту реалізації, з урахуванням ресурсних та логічних обмежень проєкту. Responsible — роль або конкретна відповідальна особа/підкоманда для реалізації гіпотези. Dependency вказує, чи залежить виконання гіпотези від результатів іншої. Execution_Order це порядок реалізації в межах управлінського циклу, що дає можливість формально описати причинно-наслідкову послідовність. Status це стан виконання: Planned / In Progress / Completed / Postponed.

Було запропоновано таку логіку реалізації гіпотез. H1 виконується наприклад, у Sprint 5, як Spike для зниження архітектурної невизначеності AccessControl. Після отримання результатів (оновлена схема інтеграції, стабілізація churn, зменшення build failures) у Sprint 6 запускається H3 як внутрішнє навчання команди з уточненої регуляторної логіки та архітектурних змін. H2, попри позитивну оцінку, відкладено через обмеження ресурсів у поточному циклі, що демонструє управлінський вибір, а не автоматичне виконання всіх згенерованих гіпотез.

Доказ ефективності розробленого методу управління складними IT-проєктами оснований на процесно-логічному підході та порівняльному аналізі варіантів управління без використання фіктивних кількісних показників. Розрахунок ефективності подано в Додатку В. В цьому пункті подано лише логічні висновки, що спираються на отримані результати.

В дисертаційному дослідженні для цього було застосовано якісну матрицю ризиків (таблиця 4.11) з якісною шкалою оцінювання: Н — низький рівень, С — середній рівень, В — високий рівень, для порівняння базового

сценарію без управління та сценарію із впровадженням розроблених моделей та методів управління складними ІТ-проєктами.

В базовому сценарії (сценарії А) ризик критичної архітектурної помилки оцінюється як високий, а адаптивність системи — як низька через відсутність раннього виявлення технологічної невизначеності.

Таблиця 4.11 – Якісна матриця ризиків для сценаріїв управління

Критерій ризику	**Сценарій А (без управління)**	**Сценарій В (Spike + Training)**
Архітектурна невизначеність (AccessControl)	В	Н
Ризик регуляторної помилки (Reporting)	В	С
Когнітивна невизначеність команди	В	Н
Ризик критичного дефекту у продакшені	В	С
Адаптивність до зміни вимог	Н	В
Стійкість до технологічних змін	Н	В

Застосування розробленого методу знижує ризик до низького або середнього рівня завдяки виконанню Spike (Н1), який зменшує архітектурну невизначеність, та Training (Н3), що знижує когнітивні ризики команди. Формалізований план реалізації гіпотез із визначенням типу втручання, залежностей, відповідальних ролей і порядковості виконання забезпечує трасованість та причинно-наслідкову логіку управління.

Аналіз чутливості (Додаток Д, таблиця Д.4) показує, що навіть при зміні технологічних умов або ресурсних обмежень сценарій із пріоритизацією гіпотез зберігає стійкість (таблиця Д.5), тоді як підхід без застосування розробленого методу управління, демонструє втрату керованості при першій суттєвій зміні середовища. Таким чином, ефективність методу доводиться через його робастність (таблиця Д.3), зниження ризику критичних помилок та підвищення адаптивності управлінської системи.

Аналіз чутливості методу управління складними ІТ-проєктами є завершальним етапом дослідження, оскільки саме на цьому етапі

перевіряється стійкість отриманих результатів до варіацій вхідних параметрів та зовнішніх умов.

Отриманий результат свідчить, що розроблений метод зменшує верхню межу інтегрального ризику складності навіть за найгірших допустимих варіацій параметрів, тобто демонструє структурну та функціональну робастність. Адекватність аналізу забезпечується коректним кодуванням порядкових даних, збереженням причинно-наслідкових залежностей гіпотез та використанням мінімаксного критерію без припущень щодо ймовірнісних розподілів. Таким чином, аналіз чутливості підтверджує, що метод управління складними IT-проєктами не лише зменшує чи ліквідовує невизначеність, але й те, що його результати залишаються стійкими та адекватними при зміні зовнішніх умов проєкту. Тобто, отримані результати гарантуватимуть ефективність проєктів для розробки і управління якими будуть застосовані розроблені в дисертації наукові та практичні результати. На рисунку 4.4 подано екранну форму дашборду, яку буде використовувати проєктний менеджер при управлінні складним IT-проєктом.

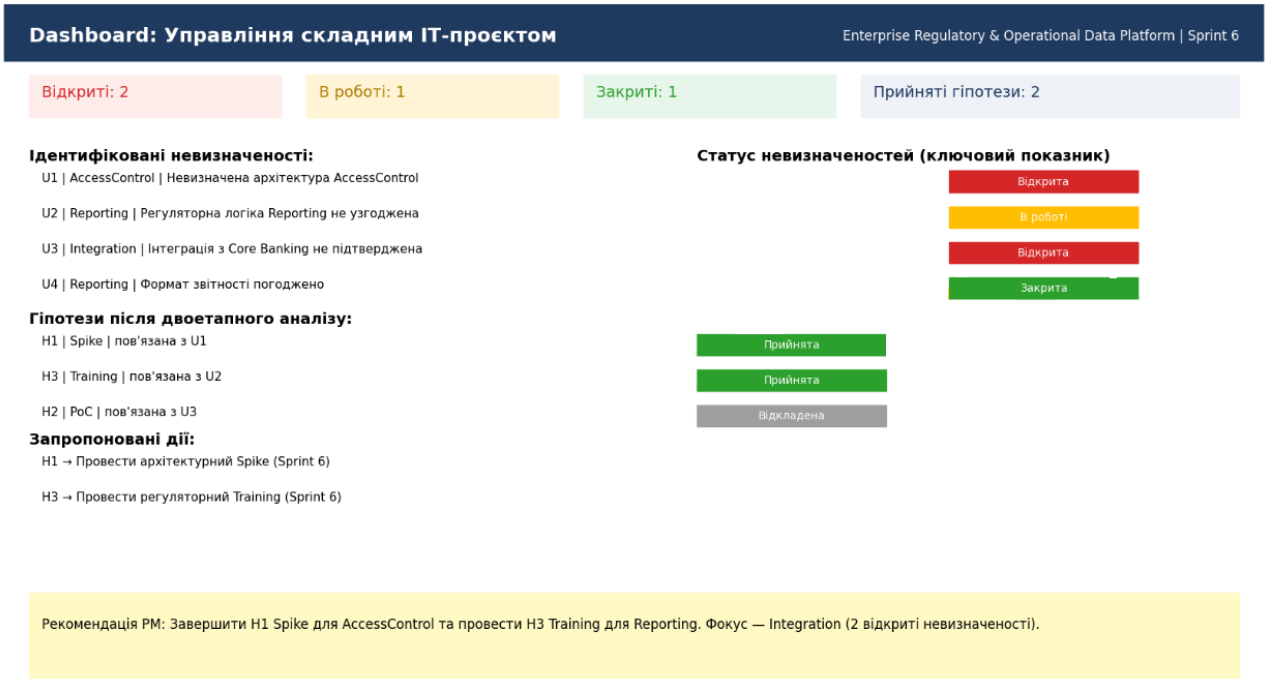


Рисунок 4.4 – Дашборд ІС системи управління складними IT-проєктами. Джерело: розроблено автором

Структура дашборду є адаптивною, та такою, що настроюється під властивості конкретного проєкту та побажання користувача.

Поданий на рисунку 4.4 дашборд містить (для прикладу): 1) назву ІТ-проєкту; 2) індикатор прогресу невизначеностей; 3) перелік ідентифікованих невизначеностей; 4) статус невизначеностей; 5) перелік відібраних гіпотез; 6) запропоновані дії до відібраних гіпотез; 7) рекомендацію інтелектуальної системи для проєктного менеджера.

Висновки до розділу 4

1. В розділі 4 виконано реалізацію системи управління складними ІТ-проєктами як цілісного інструменту підтримки адаптивного управління в умовах невизначеності, технологічної новизни та багатофакторної складності. Отримані результати підтвердили, що запропонована інформаційна система не обмежується функціями моніторингу стану проєкту, а формує інтелектуальний контур управління, у межах якого дані, аналітична інтерпретація, управлінські гіпотези, їх перевірка та рефакторинг плану утворюють логічно завершений цикл прийняття проектних рішень. На відміну від традиційних систем управління ІТ-проєктами, орієнтованих переважно на контроль строків, завдань і ресурсів, реалізована система працює з невизначеністю як з окремим об’єктом управлінського впливу, що суттєво розширює її функціональні можливості.

2. В процесі розробки було обґрунтовано мету та задачі системи, визначено її межі, акторів та варіанти використання. Було формалізовано вимоги до системи як до середовища підтримки проєктного менеджера, аналітика, технічних ролей та інтегрованих зовнішніх платформ. Побудована діаграма варіантів використання забезпечила узгодженість між функціональними можливостями системи та управлінськими сценаріями складного ІТ-проєкту. Діаграми класів забезпечила перехід від концептуального бачення до формального опису структури системи.

Визначені моделі даних, сервіси та контролери реалізують принцип модульності, єдиної відповідальності та масштабованості.

3. Логічне уявлення модулів інформаційної системи відтворює розроблений у попередніх розділах метод управління складними IT-проєктами та забезпечує її програмну реалізацію. У системі формалізовано перехід від збору та консолідації подій проєкту до виявлення зон невизначеності, генерації альтернативних управлінських гіпотез, їх оцінки, реалізації, аналізу наслідків і адаптації плану. Таким чином, реалізація інформаційної системи підтвердила працездатність запропонованого методу.

4. Запропонований синтетичний датасет дозволив перевірити працездатність системи в умовах керовано-реалістичного сценарію. Синтетичний характер датасету не знижує його наукової цінності, оскільки він відтворює не випадковий набір подій, а керовану історію розвитку складного IT-проєкту з подіями, аномаліями, прихованими невизначеностями, управлінськими реакціями та наслідками цих реакцій. Завдяки цьому було показано, що система здатна централізовано об’єднувати дані з Jira, Git, CI/CD, QA та комунікаційних інструментів, інтерпретувати їх як сигнали проблемного стану та трансформувати у формалізовані управлінські артефакти. Запропонований сценарний механізм і генератор синтетичного датасету підтвердили, що система може бути використана не лише для демонстрації працездатності, але й як інструмент подальших експериментів з управлінськими політиками.

5. У межах демонстраційного експерименту на прикладі проєкту BankDataHub було показано, що система коректно проходить повний управлінський цикл: від фіксації початкового стану, централізованого збору спостережень і виявлення невизначеностей до інтерпретації причин, формування гіпотез, їх оцінки, ранжування та включення у план реалізації. Побудовані таблиці невизначеностей, невідповідностей плану і факту, нестабільних ділянок, інтерпретованих проблем і гіпотез забезпечили трасованість між симптомами, причинами та запропонованими

управлінськими діями. Це свідчить, що реалізована система підтримує не інтуїтивне, а логічно обґрунтоване управління складним ІТ-проєктом.

5. Обґрунтовано доцільність вибору управлінських втручань через механізм оцінки та пріоритизації гіпотез. Модель багатокритеріального вибору забезпечило можливість переходу від множини альтернативних дій до обмеженого набору рішень, які дійсно доцільно реалізовувати в поточному етапі життєвого циклу проєкту. Подальший план реалізації гіпотез і пов’язаний з ним механізм рефакторингу плану управління продемонстрували, що система здатна змінювати хід проєкту не декларативно, а через конкретні керовані втручання, пов’язані з ресурсами, ролями, залежностями та спринтами.

6. Реалізація системи управління складними ІТ-проєктами підтвердила практичну здійсненність і внутрішню узгодженість усіх розроблених у дисертації моделей і методів. Система забезпечує поєднання збору даних, інтелектуального аналізу, сценарного моделювання, підтримки рішень та адаптації плану управління в єдиній ІС. Це дозволяє розглядати її як науково обґрунтовану та прикладно значущу основу для підвищення керованості, адаптивності та прогнозованості управління складними ІТ-проєктами через зниження невизначеності проєктного середовища.

7. Результати досліджень четвертого розділу опубліковані в роботах [1, 6].

Список використаних літературних джерел у розділі 4

1. Путій І. (2025). Розробка інтелектуальної системи управління складними ІТ-проєктами. Штучний інтелект в інклюзивному розвитку. AIID-2025: Тези доповідей І Міжнародної науково-практичної конференції. Одеса. ІШІР, 123 – 127. https://www.doi.org/10.5281/zenodo.19053150

2. Путій І. Д., Тесленко П. О. (2025). «Канвас» методу формування гіпотез для складного ІТ-проєкту. Управління розвитком складних систем, 64, 120–127. DOI: 10.32347/2412-9933.2025.64.

3. Єфремов, М. Ф., Єфремов, Ю. М., & Єфремов, В. М. (2016). Проектування програмного забезпечення з використанням UML. https://eztuir.ztu.edu.ua/bitstream/handle/123456789/3296/12.pdf?sequence=1

4. Путій І. Д., Тесленко П. О. (2025). Концептуальна модель складного ІТ-проєкту. Таврійський науковий вісник. Серія: Технічні науки, (2), 148-154. DOI: https://doi.org/10.32782/tnv-tech.2025.2.16

5. Полотай О. І. (2011). Використання діаграми класів UML для запровадження освітніх ІТ-проектів у ВНЗ. Торгівля, комерція, підприємництво, (13), 111-115.

6. Ілля Путій, Павло Тесленко. (2025). Інформаційна система управління складними ІТ-проєктами. Труди ОНПУ, №2, 256–278

7. Ілля Путій, Павло Тесленко. (2025). Рефакторинг-метод управління складними ІТ-проєктами. Project, Program, Portfolio Management. P3M-2025: Тези доповідей VIII Міжнародної науково-практичної конференції. Одеса. ІШІР, 190 – 193. https://www.doi.org/10.5281/zenodo.18428325

8. Бєлан, В., & Думин, І. (2022, November). Синтетичні дані і їхнє застосування в області нейронних мереж. In The 9th International scientific and practical conference “Study of world opinion regarding the development of science”(November 22-25, 2022) Prague, Czech Republic. International Science Group. 2022. 734 p. (p. 635).

9. Путій І.Д., Бондар О.А. (2024). Складність ІТ-проєктів з науково-дослідною та дослідно-конструкторською розробкою. Міжнародна науково-практична конференція «Інформаційні системи в управлінні проєктами та програмами», Коблево, ХНУРЕ, 197 –200.

10. Joseph Nazeer, Marnewick. (2021). Measuring Information Systems Project Complexity: A Structural Equation Modelling Approach. Complexity, 1. URL : https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/5907971. DOI :https://doi.org/10.1155/2021/5907971.

11. Janczarek, P., & Sosnowski, J. (2015). Managing Complex Software Projects. Information Systems Management, 4(3), 171-182.

12. Marle F. (2020). An Assistance to Project Risk Management Based on Complex Systems Theory and Agile Project Management. Complex., 2020, 3739129:1-3739129:20. https://doi.org/10.1155/2020/3739129.

13. Коптєва Г.М. (2025). Побудова структурно-логічної моделі управління невизначеністю в інформаційно-технологічних проєктах. Економіка: реалії часу. Науковий журнал. 6 (82). 100-109. URL : https://economics. net.ua/files/archive/2025/No6/100.pdf. DOI: 10.15276/ETR.06.2025.10. DOI: 10.5281/zenodo.18064574.

14. Wang Haiyan, Sheng Zhaohan.Big Data-Driven Modeling of Complex System Management Scenarios: Technologies and ProcessesChinese Journal of Management Science [J], 2025, 33(1): 22-33 doi:10.16381/j.cnki.issn1003-207x.2024.2083

15. Stanford, C. A., Luk, W., Weston, S., Guo, C., Babaie-Harmon, J., & Vytelingum, K. Agent-based Modeling for Scenario Analysis in Management Consulting. Technical paper. WILMOTT magazine. https://www.doc.ic.ac.uk/~cg1710/pub/2024/wilmott24cs.pdf?utm_source

16. Galli B. (2019). The Application of Systems Engineering to Project Management: A Review of Their Relationship. Int. J. Syst. Dyn. Appl., 9, 81-106. https://doi.org/10.4018/ijsda.2020010105.

17. Близнюкова І.О., Тесленко П.О., Малахова Д.О. (2022). Особливості формування команди управління ІТ-проєктом. Вісник Національного технічного університету "ХПІ". Сер. : Стратегічне управління, управління портфелями, програмами та проектами : зб. наук. пр. Харків : НТУ "ХПІ",– № 2 (6). 14-20.

ВИСНОВКИ

У дисертаційній роботі вирішено складну науково-практичну задачу підвищення ефективності управління складними ІТ-проєктами в умовах невизначеності шляхом розроблення та вдосконалення моделей, методів і інформаційних засобів зменшення впливу невизначеності проєктного середовища та забезпечення можливості прийняття обґрунтованих проєктних рішень. На відміну від існуючих підходів, орієнтованих переважно на окремі аспекти управління проєктами, у роботі реалізовано комплексний підхід до дослідження невизначеності, який охоплює процеси її виявлення, кількісного оцінювання, прогнозування, моніторингу та врахування під час підтримки прийняття рішень у складних ІТ-проєктах.

У науковому аспекті отримані результати дозволили розвинути теоретичні положення управління складними ІТ-проєктами в умовах невизначеності та сформувати цілісний інформаційно-аналітичний підхід до підтримки проєктних рішень. Розроблені моделі та методи забезпечили формалізацію процесів оцінювання невизначеності, побудову механізмів аналізу проєктного середовища та створення інструментів інтелектуальної підтримки управлінських рішень. Запропоновані рішення дозволили інтегрувати моделі оцінювання ризиків, прогнозування та аналізу проєктних даних у єдину інформаційну технологію управління складними ІТ-проєктами. Отримані результати розширюють науково-методичний апарат управління проєктами та створюють передумови для подальшого розвитку інтелектуальних технологій підтримки прийняття рішень у динамічному та слабо прогнозованому середовищі.

У практичному аспекті результати роботи були реалізовані у вигляді комплексу моделей, методів та інформаційних засобів, що утворюють інформаційну технологію управління складними ІТ-проєктами в умовах невизначеності. Запропоновані рішення забезпечують зменшення впливу невизначеності на процеси планування, контролю та прийняття проєктних

рішень, підвищують обґрунтованість вибору управлінських дій та сприяють покращенню якості управління проєктами. Практична цінність одержаних результатів полягає у можливості їх застосування під час реалізації складних ІТ-проєктів, що функціонують у динамічному середовищі та характеризуються високим рівнем невизначеності, значною кількістю взаємозалежних факторів і необхідністю оперативного реагування на зміни зовнішніх та внутрішніх умов.

Розроблені моделі, методи та інформаційна система не є ізольованими рішеннями, а утворюють взаємопов'язану інформаційну технологію, в межах якої реалізовано наскрізний цикл підтримки прийняття проєктних рішень. Її особливістю є поєднання засобів аналізу, прогнозування та інтелектуальної обробки даних, що забезпечує адаптивність до змін проєктного середовища та можливість урахування невизначеності під час управління складними ІТ-проєктами. Це дозволяє підвищити ефективність використання ресурсів, покращити точність прогнозування та зменшити ймовірність прийняття необґрунтованих управлінських рішень.

Адекватність розроблених моделей, методів та інформаційної системи підтверджується результатами обчислювальних експериментів, моделювання, тестування та практичної апробації, які засвідчили їх працездатність, узгодженість із поставленими завданнями та здатність забезпечувати досягнення очікуваних результатів. Отримані показники ефективності свідчать про доцільність використання запропонованої інформаційної технології для підтримки процесів управління складними ІТ-проєктами в умовах невизначеності.

Таким чином, сукупність отриманих наукових і практичних результатів дозволяє стверджувати, що поставлена у роботі мета, яка полягала у підвищенні ефективності управління складними ІТ-проєктами через розробку та вдосконалення моделей, методів та інформаційних засобів зменшення невизначеності проєктного середовища задля забезпечення можливості прийняття обґрунтованих проєктних рішень, досягнута в повному обсязі.

Розроблена в дисертаційній роботі інформаційна система управління складними ІТ-проєктами в умовах невизначеності апробована в ТОВ «СФ Інвест», науково-практичні результати використані у навчальному процесі на кафедрі штучного інтелекту та аналізу даних Національного університету «Одеська політехніка» в освітній програмі магістерської підготовки «Управління ІТ-проєктами» та в освітній програмі бакалаврської підготовки «Інтеленктуальний аналіз даних», також результати використані в НДР кафедри ШІАД: № 236-177 «Інтелектуальні технології управління ІТ-проєктами в умовах невизначеності» (номер державної реєстрації 0123U102593) та НДР № 237-177 «Програмні системи машинного навчання та їх застосування в галузі інтелектуального аналізу даних», (номер держреєстрації 01230102594).

ДОДАТОК А. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
«СФ ІНВЕСТ»
65042, м. Одеса, вул. Балтська дорога, 6; ЄДРПОУ 44482912

«ЗАТВЕРДЖУЮ»
Директор ТОВ «СФ ІНВЕСТ»
«10» березня 2026 року

Вих. № 14 від 10.03.26

АКТ
використання результатів дисертаційної роботи
Путія Іллі Дмитровича на тему:
«Інформаційна технологія управління складними ІТ-проєктами в умовах невизначеності»

Цим актом підтверджується, що результати дисертаційної роботи Путія І.Д. були використані ТОВ "СФ Інвест" при реалізації ІТ-проєктів компанії, що виконувались протягом 2025 року. Була задіяна у бізнесові процеси компанії інформаційна система управління складними ІТ-проєктами для зниження невизначеності

Використання розробленого дисертантом програмного продукту, на основі розроблених методів та моделей дозволило забезпечити успішне завершення складних ІТ-проєктів за рахунок зниження невизначеності проєктного середовища та забезпечило зменшення кількості перевищених дедлайнів у порівнянні з подібними ІТ-проєктами, що виконувалися раніше без її застосування.

Директор ТОВ «СФ ІНВЕСТ» Євгеній ЧЕХОВСЬКИЙ

УКРАЇНА
ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
«СФ Інвест»
44482912
м. Одеса Одеська обл.

Україна
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

КОД ЄДРПОУ 43861328

пр.Шевченка, 1, м.Одеса, 65044
Україна
тел. +38 048 7223474
факс. +38 048 7221992
E-mail: mail@op.edu.ua

11.06.2026 №1007/181-07
на №

ДОВІДКА

про використання результатів дисертаційної роботи
Путія Іллі Дмитровича
«Інформаційна технологія управління складними ІТ-проєктами в умовах невизначеності»
у науково-дослідницькій діяльності
Національного університету «Одеська політехніка»

Довідку видано для підтвердження того, що у науково-дослідній діяльності Національного університету «Одеська політехніка» використовуються такі наукові результати, отримані в дисертаційному дослідженні Путія Іллі Дмитровича:

– інформаційна модель складного ІТ-проєкту з науково-дослідницькою та дослідно-конструкторською складовою враховуючи доменну структуру предметної галузі;

– концептуальна модель управління складним ІТ-проєктом в умовах невизначеності з науково-дослідницькою та дослідно-конструкторською складовою з адоптацією процедури рефакторингу до управління проєктами, що зробить управління проєктом прогнозованим, гнучким і науково обґрунтованим;

– модель управління гіпотезами складних ІТ-проєктів;

– метод управління складними ІТ-проєктами в умовах невизначеності;

– інформаційна система управління складними ІТ-проєктами в умовах невизначеності.

Зазначені результати використовуються під час виконання НДР № 236-177 «Інтелектуальні технології управління ІТ-проєктами в умовах невизначеності» (номер державної реєстрації 0123U102593) та НДР № 237-177 «Програмні системи машинного навчання та їх застосування в галузі інтелектуального аналізу даних», (номер держреєстрації 01230102594), а також під час підготовки наукових публікацій, методичних матеріалів і дослідних прототипів.

Використання зазначених результатів підвищує методичну забезпеченість та наукову складову при підготовці здобувачів вищої освіти.

Проректор з наукової та науково-педагогічної роботи, д.т.н., професор Дмитро ДМИТРИШИН

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ • НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА» • 43861328

Україна
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

КОД ЄДРПОУ 43861328

пр.Шевченка, 1, м.Одеса, 65044
Україна
тел. +38 048 7223474
факс. +38 048 7221992
E-mail: mail@op.edu.ua

10.06.2026 № 1003/181-07
на №

ДОВІДКА

про використання результатів дисертаційного дослідження
Путія Іллі Дмитровича
«Інформаційна технологія управління складними ІТ-проєктами в умовах невизначеності»
у навчальному процесі
Національного університету «Одеська політехніка»

Довідку видано для підтвердження використання наукових результатів дисертаційного дослідженні Путія Іллі Дмитровича «Інформаційна технологія управління складними ІТ-проєктами в умовах невизначеності» в навчальному процесі кафедри штучного інтелекту та аналізу даних Інституту штучного інтелекту та робототехніки за спеціальністю 122 «Комп'ютерні науки» для підготовки здобувачів:

– першого (бакалаврського) рівня вищої освіти за освітньою програмою «Інтелектуальний аналіз даних» дисципліна «Управління ІТ-проєктами» – у межах тем, присвячених управлінню складними ІТ-проєктами ;

– другого (магістерського) рівня вищої освіти за освітньою програмою «Управління ІТ-проєктами» дисципліна «Методологія управління ІТ-проєктами» – результати дисертації застосовуються для ознайомлення здобувачів із принципами управління складними проєктами та системами.

Використання наукових результатів актуалізує освітні компоненти та посилює практичну значущість навчального процесу через формування у здобувачів компетентностей щодо управління гіпотезами, розробки інформаційних систем управління складними ІТ-проєктами в умовах невизначеності.

Перший проректор, проректор з науково-педагогічної та виховної роботи, д.т.н., професор — Сергій НЕСТЕРЕНКО

Голова методичної ради ШШР, к.т.н., доцент — Павло СТУПЕНЬ

ДОДАТОК Б СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Путій І. Д., Тесленко П. О. (2024). Аналіз сучасних підходів до управління складними ІТ-проєктами. Управління розвитком складних систем, 59, 81 – 89.

DOI: 10.32347/2412-9933.2024.59.81-88

2. Путій, І. Д., & Тесленко, П. О. (2025). Концептуальна модель складного ІТ-проєкту. Таврійський науковий вісник. Серія: Технічні науки, (2), 148-154.

DOI: https://doi.org/10.32782/tnv-tech.2025.2.16

3. Путій І. Д., Тесленко П. О. (2025). «Канвас» методу формування гіпотез для складного ІТ-проєкту. Управління розвитком складних систем, 64, 120–127.

DOI: 10.32347/2412-9933.2025.64.

4. I. Putii, P. Teslenko. (2025) Complex IT project management information system. Proceedings of Odessa Polytechnic University, ISSN 2223-3814 (online) Issue 2(72), 119 – 127.

DOI: 10.15276/opu.2.72.2025.13

5. Путій І.Д., Бондар О.О., Скопін Р.П. (2023). Особливості проєктів створення радіотехнічних продуктів. Збірка праць МНПК «Інтелектуальні інформаційні системи в управлінні проєктами та програмами», Коблево, 12–15 вересня 2023 р. — Харків: ХНУРЕ, 2023, С. 167-169.

6. Путій, А. Косенко, О.Бондар. (2023). Управління проєктами створення радіотехнічних пристроїв як складними системами // Project, Program, Portfolio Management. Р3М-2023: Тези доп. VIII Міжнар. науково-практ. конф., м. Одеса, 1–2 груд. 2023 р. Одеса, 2023. – С. 201-204

7. Путій І.Д., Бондар О.А., Мартиненко С.С. (2024). Особливості управління дослідженнями в складних ІТ проєктах. Кібербезпека, робототехніка, автоматика, комп'ютерна інженерія. КРАКІН'24. Матеріали 59-

ї науково-технічної конференції студентів та молодих вчених ІШІР (13-15 травня 2024 р.). – Одеса: Національний університет «Одеська політехніка», С. 5 – 7.

8. Путій І.Д., Бондар О.А. (2024). Складність IT-проєктів з науково-дослідною та дослідно-конструкторською розробкою. Міжнародна науково-практична конференція «Інформаційні системи в управлінні проєктами та програмами», Коблево, ХНУРЕ, 197 –200.

9. Путій І.Д, Бобровський О. (2024). Інформаційна модель складного IT-проєкту з дослідницькою складовою. Project, Program, Portfolio Management. P3M-2024: Тези доповідей IX Міжнародної науково-практичної конференції:[у 2т.]. Відп. за вип. П.О. Тесленко. Том 1. Одеса: ІШІР, 2024. С. 316-319.

https://www.doi.org/10.5281/zenodo.151651746.

10. Ілля Путій. (2025). Розробка інтелектуальної системи управління складними IT-проєктами. Штучний інтелект в інклюзивному розвитку. AIID-2025: Тези доповідей I Міжнародної науково-практичної конференції. Одеса. ІШІР, 123 – 127.

https://www.doi.org/10.5281/zenodo.19053150

11. Ілля Путій, Павло Тесленко. (2025). Рефакторинг-метод управління складними IT-проєктами. Project, Program, Portfolio Management. P3M-2025: Тези доповідей VIII Міжнародної науково-практичної конференції. Одеса. ІШІР, 190 – 193.

https://www.doi.org/10.5281/zenodo.18428325

ДОДАТОК В. ПРОЄКТНА БАЗА СПОСТЕРЕЖЕНЬ


```
[
 {
  "event_id": "EVT-001",
  "timestamp": "2024-03-01T09:10:00Z",
  "source": "Jira",
  "entity_id": "US-101",
  "event_type": "story_estimation_recorded",
  "module": "DataIngest",
  "attributes": {
   "metric": "story_points",
   "planned": 5,
   "actual": 8,
   "sprint": "Sprint-2",
   "priority": "High"
  }
 },
 {
  "event_id": "EVT-002",
  "timestamp": "2024-03-03T18:40:00Z",
  "source": "Jira",
  "entity_id": "US-102",
  "event_type": "duration_recorded",
  "module": "AccessControl",
  "attributes": {
   "metric": "duration_days",
   "planned": 6,
   "actual": 14,
   "sprint": "Sprint-2",
   "priority": "High"
  }
 },
 {
  "event_id": "EVT-003",
  "timestamp": "2024-03-05T13:22:00Z",
  "source": "Git",
  "entity_id": "commit-884",
  "event_type": "code_churn_recorded",
  "module": "AccessControl",
  "attributes": {
   "metric": "churn_lines",
   "planned": 120,
   "actual": 540,
   "author": "dev_backend_2"
  }
 },
 {
  "event_id": "EVT-004",
  "timestamp": "2024-03-06T15:12:00Z",
  "source": "CI/CD",
  "entity_id": "build-221",
  "event_type": "build_status_recorded",
  "module": "AccessControl",
  "attributes": {
```

```
    "metric": "build_status",
    "planned": "success",
    "actual": "fail",
    "pipeline": "main"
   }
  },
  {
   "event_id": "EVT-005",
   "timestamp": "2024-03-08T11:05:00Z",
   "source": "QA",
   "entity_id": "bug-17",
   "event_type": "defect_severity_recorded",
   "module": "Reporting",
   "attributes": {
    "metric": "severity",
    "planned": "medium",
    "actual": "critical",
    "test_phase": "system_testing"
   }
  },
  {
   "event_id": "EVT-006",
   "timestamp": "2024-03-10T10:30:00Z",
   "source": "Jira",
   "entity_id": "US-110",
   "event_type": "sprint_reassignment",
   "module": "DataIngest",
   "attributes": {
    "metric": "sprint_assignment",
    "planned": "Sprint-2",
    "actual": "Sprint-4",
    "reason": "Unresolved dependency"
   }
  },
  {
   "event_id": "EVT-007",
   "timestamp": "2024-03-12T16:50:00Z",
   "source": "Git",
   "entity_id": "commit-902",
   "event_type": "commit_frequency_recorded",
   "module": "AccessControl",
   "attributes": {
    "metric": "frequency_week",
    "planned": 4,
    "actual": 19,
    "author_group": "backend_team"
   }
  }
]
```

ДОДАТОК Г. КОД ФОРМАЛІЗОВАНОЇ ОЦІНКИ

```
import pandas as pd
import numpy as np

data = pd.DataFrame({
    "Hypothesis_ID": ["H1","H2","H3"],
    "Impact": [0.8,0.6,0.9],
    "Risk_Reduction": [0.7,0.5,0.4],
    "Cost": [0.6,0.3,0.8],
    "Time": [0.5,0.4,0.7]
})

weights = {"Impact":0.4,"Risk_Reduction":0.3,"Cost":0.2,"Time":0.1}

for col in ["Impact","Risk_Reduction","Cost","Time"]:
    data[col] = (data[col]-data[col].min())/(data[col].max()-data[col].min())

data["Score"] = (
    weights["Impact"]*data["Impact"] +
    weights["Risk_Reduction"]*data["Risk_Reduction"] -
    weights["Cost"]*data["Cost"] -
    weights["Time"]*data["Time"]
)

data = data.sort_values("Score",ascending=False)
data.to_csv("evaluation_results.csv",index=False)
```

ДОДАТОК Д. РОЗРАХУНОК ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНИХ РІШЕНЬ

1) Кодування ordinal-оцінок і уніфікація напрямку “чим більше, тим гірше”

Нехай маємо множину критеріїв $i = 1..m$і ordinal-оцінку $x_i \in$ {H,C,B}. Кодування:

$$\text{H} = 1, \text{C} = 2, \text{B} = 3.$$

Для інтегрального підсумовування всі критерії приводимо до єдиного змісту “більше = гірше” через індикатор ризику r_i.

Якщо критерій є **ризиковим** (негативним) — архітектурна невизначеність, регуляторний ризик, когнітивна невизначеність, ризик критичного дефекту:

$$r_i = x_i.$$

Якщо критерій є **позитивним** (benefit) — адаптивність, стійкість (чим більше, тим краще), робимо інверсію:

$$r_i = 4 - x_i.$$

Тоді для всіх i: менше r_iозначає менший інтегральний ризик.
Ми використовуємо 6 критеріїв ($m = 6$):
C_1архітектурна невизначеність (ризик),
C_2регуляторний ризик (ризик),
C_3когнітивна невизначеність (ризик),
C_4ризик критичного дефекту (ризик),
C_5адаптивність (benefit),
C_6стійкість до змін (benefit).

Чисельні дані :
Сценарій A: $x^A = (3,3,3,3,1,1)$.
Сценарій B: $x^B = (1,2,1,2,3,3)$.

Перетворення в r: Для A: перші 4 — без змін; C_5, C_6: $4 - 1 = 3$.

$$r^A = (3,3,3,3,3,3).$$

Для B: перші 4 — без змін; C_5, C_6: $4 - 3 = 1$.

$$r^B = (1,2,1,2,1,1).$$

2) Розрахунок інтегрального ризику в базовому стані

Інтегральний ризик політики πу стані s(або “базовому” s_0):

$$R(\pi, s) = \sum_{i=1}^{m} w_i \, r_i(\pi, s) \, , \sum_{i=1}^{m} w_i = 1, \, w_i \geq 0.$$

Для першого проходу беремо рівні ваги (це стандартний нейтральний вибір, якщо ваги ще не калібровані експертно):

$$w_i = \frac{1}{6}.$$

Розрахунок для A (база s_0)

$$R(A, s_0) = \frac{1}{6}(3 + 3 + 3 + 3 + 3 + 3) = \frac{18}{6} = 3.00.$$

Розрахунок для B (база s_0)

$$R(B, s_0) = \frac{1}{6}(1 + 2 + 1 + 2 + 1 + 1) = \frac{8}{6} = 1.333.$$

3) Робастний (мінімакс) розрахунок $R_{\max}$

Робастний показник:

$$R_{\max}(\pi) = \max_{s \in S} R(\pi, s),$$

де S— множина стрес-сценаріїв зовнішнього середовища.

Ми використовуємо 3 стрес-сценарії (як уже було погоджено раніше в розрахунках):

s_1— різка зміна регуляторики,

s_2— технологічна зміна в AccessControl,

s_3— ресурсне обмеження спринту.

Чисельні дані для стресів (узгоджені раніше)

Для A приймаємо, що система і так у “стелі” ризику, тому:

$$r^{A,s_1} = r^{A,s_2} = r^{A,s_3} = (3,3,3,3,3,3) \Rightarrow R(A, s) = 3.00.$$

Для B (погіршення лише частини критеріїв):

- s_1: C_2: $2 \to 3$, C_4: $2 \to 3$

$$r^{B,s_1} = (1,3,1,3,1,1), R(B, s_1) = \frac{10}{6} = 1.667.$$

- s_2: C_1: $1 \to 2$, C_4: $2 \to 3$

$$r^{B,s_2} = (2,2,1,3,1,1), R(B, s_2) = \frac{10}{6} = 1.667.$$

- s_3: C_3: $1 \to 2$, C_4: $2 \to 3$

$$r^{B,s_3} = (1,2,2,3,1,1), R(B,s_3) = \frac{10}{6} = 1.667.$$

Мінімакс-підсумок

$$R_{\max}(A) = \max(3.00,3.00,3.00) = 3.00,$$
$$R_{\max}(B) = \max(1.667,1.667,1.667) = 1.667.$$

Робастний виграш:

$$\Delta_{\text{robust}} = R_{\max}(A) - R_{\max}(B) = 3.00 - 1.667 = 1.333.$$

4) Розрахунок чутливості до зовнішніх впливів (дельти відносно бази)

Для дискретної ordinal-моделі коректно використовувати прирости:

$$\Delta R(\pi, s) = R(\pi, s) - R(\pi, s_0).$$

Для A:

$$\Delta R(A, s_1) = 3.00 - 3.00 = 0, \Delta R(A, s_2) = 0, \Delta R(A, s_3) = 0.$$

Для B:

$$\Delta R(B, s_1) = 1.667 - 1.333 = 0.334,$$
$$\Delta R(B, s_2) = 0.334, \Delta R(B, s_3) = 0.334.$$

Додатково (щоб показати “чутливість профілю”), можна рахувати векторну “амплітуду” зміни критеріїв у стресі:

$$\Delta\mathbf{r}(\pi, s) = \mathbf{r}(\pi, s) - \mathbf{r}(\pi, s_0), \| \Delta\mathbf{r} \|_1 = \sum_i | \Delta r_i |.$$

Для B:

- s_1: $\Delta\mathbf{r} = (0, +1, 0, +1, 0, 0) \Rightarrow \| \Delta\mathbf{r} \|_1 = 2$
- s_2: $(+1, 0, 0, +1, 0, 0) \Rightarrow 2$
- s_3: $(0, 0, +1, +1, 0, 0) \Rightarrow 2$

Для A завжди нуль.

5) Робастність висновку до невідомих ваг критеріїв

Мета: довести, що результат “B краще за A” не є артефактом вибору рівних ваг.

Різниця інтегральних ризиків у базі:

$$R(A, s_0) - R(B, s_0) = \sum_{i=1}^{6} w_i \, (r_i^A - r_i^B).$$

Маємо:

$$r^A - r^B = (3-1,\ 3-2,\ 3-1,\ 3-2,\ 3-1,\ 3-1) = (2,1,2,1,2,2).$$

Отже:

$$R(A, s_0) - R(B, s_0) = 2w_1 + w_2 + 2w_3 + w_4 + 2w_5 + 2w_6.$$

Оскільки $w_i \geq 0$та $\sum w_i = 1$, мінімально можливе значення цієї різниці досягається, коли вся вага припадає на критерії з найменшим коефіцієнтом (тут це коефіцієнт 1 у w_2або w_4). Тоді:

$$R(A, s_0) - R(B, s_0) \geq 1 \cdot \sum_{i=1}^{6} w_i = 1.$$

Тобто навіть у найгіршому для нас випадку ваг (але з не-негативними вагами) маємо гарантію:

$$R(A, s_0) \; - \; R(B, s_0) \; \geq \; 1,$$

а отже домінування Bнад Aє стійким до вибору ваг. Аналогічно цей аргумент переноситься і на стрес-стани, якщо вектор різниць $r^{A,s} - r^{B,s}$залишається невід'ємним покомпонентно (в наших стресах це виконується).

6) Редукція невизначеності від управлінських втручань (H1 Spike, H3 Training)

Оскільки H1 і H3 — це цілеспрямовані інтервенції, їх ефект доцільно показувати як зменшення відповідних компонент ризикового профілю. Для цього вводимо "крок редукції" по критерію:

$$\Delta r_i = r_i^{\text{до}} - r_i^{\text{після}}.$$

У нашій логіці "до" відповідає сценарію A (без запропонованого методу), "після" — сценарію B (із запропонованим методом). Тоді загальна редукція:

$$\Delta R = R(A, s_0) - R(B, s_0).$$

А розклад ефекту за критеріями:

$$\Delta R = \sum_{i=1}^{6} w_i \Delta r_i, \Delta r_i = r_i^A - r_i^B.$$

Ми вже маємо $\Delta r = (2,1,2,1,2,2)$.

Щоб "прив'язати" це до гіпотез, фіксуємо відповідність (вона випливає з вашого опису типів інтервенцій):

H1 Spike по AccessControl зменшує насамперед C_1(архітектурна невизначеність) і частково C_4(ризик критичного дефекту через стабілізацію модуля).

H3 Training по Reporting зменшує C_3(когнітивна невизначеність) і C_2(регуляторний ризик через коректне трактування доменно-регуляторної логіки).

Тоді "внесок" гіпотез у редукцію (на рівні критеріїв) можна представити як суму їхніх Δr по релевантних компонентах:

$$\Delta r(H1) = \Delta r_1 + \Delta r_4 = 2 + 1 = 3,$$
$$\Delta r(H3) = \Delta r_2 + \Delta r_3 = 1 + 2 = 3.$$

За рівних ваг це переводиться в інтегральні внески:

$$\Delta R(H1) = \frac{1}{6}(2 + 1) = \frac{3}{6} = 0.5,$$
$$\Delta R(H3) = \frac{1}{6}(1 + 2) = \frac{3}{6} = 0.5.$$

Залишкова частина редукції (адаптивність і стійкість) відображає системний ефект методу як цілісної політики:

$$\Delta R(\text{system}) = \frac{1}{6}(\Delta r_5 + \Delta r_6) = \frac{1}{6}(2 + 2) = \frac{4}{6} = 0.667.$$

Перевірка суми:

$$0.5 + 0.5 + 0.667 = 1.667 \approx 3.00 - 1.333 = 1.667$$

Це різниця базових ризиків; робастний виграш за worst-case ми рахували окремо як 1.333.

Далі, результати розрахунків подана в таблицях де згруповані дані з підрозділу 4.3.

Таблиця Д.1 – Вхідні ordinal-оцінки x_iта уніфіковані ризики r_i(база)

Критерій	**Тип**	**A: x_i**	**A: r_i**	**B: x_i**	**B: r_i**
C_1Архітектурна невизначеність	ризик	3	3	1	1
C_2Регуляторний ризик	ризик	3	3	2	2
C_3Когнітивна невизначеність	ризик	3	3	1	1
C_4Ризик критичного дефекту	ризик	3	3	2	2
C_5Адаптивність	benefit	1	3 (=4−1)	3	1 (=4−3)
C_6Стійкість до змін	benefit	1	3 (=4−1)	3	1 (=4−3)

Таблиця фіксує, які саме значення підставляються у формулу R, з трасованістю від ordinal до "ризикової" шкали.

Таблиця Д.2 – Інтегральний ризик у базовому стані (рівні ваги $w_i = 1/6$)

Політика	$\sum r_i$	$R(\pi, s_0) = \frac{1}{6}\sum r_i$
A	18	3.000
B	8	1.333

Базовий інтегральний ризик у B нижчий на $3.000 - 1.333 = 1.667$.

Таблиця Д.3 – Робастний (мінімакс) результат по стрес-сценаріях

Сценарій середовища	$\boldsymbol{R(A,s)}$	$\boldsymbol{R(B,s)}$
s_0базовий	3.000	1.333
s_1регуляторні зміни	3.000	1.667
s_2технологічні зміни	3.000	1.667
s_3ресурсні обмеження	3.000	1.667
$R_{\max}$	3.000	1.667

Worst-case для B все одно нижчий; робастний виграш $\Delta_{\text{robust}} = 1.333$.

Таблиця Д.4 – Чутливість $\Delta R(\pi, s) = R(\pi, s) - R(\pi, s_0)$

Сценарій	$\boldsymbol{\Delta R(A,s)}$	$\boldsymbol{\Delta R(B,s)}$
s_1	0.000	0.334
s_2	0.000	0.334
s_3	0.000	0.334

B змінюється під стресом (що нормально), але залишається далеко від рівня A, саме це і показує робастність.

\

Таблиця Д.5 – Стійкість висновку до ваг (аналітичний результат)

Вираз	**Результат**
$R(A) - R(B) = 2w_1 + w_2 + 2w_3 + w_4 + 2w_5 + 2w_6$	—
За умов $w_i \geq 0, \sum w_i = 1$	$R(A) - R(B) \geq 1$

Домінування B не “ламається”, навіть якщо ваги критеріїв змінюються (в межах невід’ємності та нормування).

Таблиця Д.6 – Редукція невизначеності/ризику за критеріями та внесок гіпотез

Критерій	$\boldsymbol{\Delta r_i = r_i^A - r_i^B}$	**Прив’язка до інтервенції**
C_1	2	H1 Spike (ключовий)
C_2	1	H3 Training (частково)
C_3	2	H3 Training (ключовий)
C_4	1	H1 Spike (частково)
C_5	2	системний ефект методу
C_6	2	системний ефект методу

H1 і H3 дають вимірюваний внесок у зниження невизначеності по релевантних критеріях, а “адаптивність/стійкість” відображають узагальнений ефект всієї управлінської політики.