МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Войтишин Володимир Володимирович

Кваліфікаційна наукова
праця на правах рукопису

УДК 004.94+316.77:004

ДИСЕРТАЦІЯ

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОЦІНЮВАННЯ ПРОЄКТІВ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

122 – Комп’ютерні науки

Подається на здобуття наукового ступеня доктора філософії

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

_____ / В.В. Войтишин /
(підпис, ініціали та прізвище здобувача)

Науковий керівник:
Теслюк Василь Миколайович
доктор технічних наук, професор

Львів – 2025

АНОТАЦІЯ

Войтишин В.В. Інформаційна технологія оцінювання проєктів з розробки програмного забезпечення. — Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки». — Національний університет «Львівська політехніка», Львів, 2025.

Зміст анотації.

Дисертаційне дослідження присвячене розробленню методів оцінювання проєктів з розробки програмного забезпечення, що застосовні на етапах аналізу та проєктування життєвого циклу проєкту, а також розробленню інформаційної технології, яка забезпечує використання цих методів на практиці. Розроблені методи та інформаційна технологія сприятимуть підвищенню якості оцінок проєктів з розробки програмного забезпечення, дадуть змогу скоротити час, необхідний для підготовки цих оцінок, створять умови для накопичення даних про проєкти та їх оцінки, а також забезпечать опрацювання слабоструктурованого бізнес-процесу оцінювання із застосуванням методів процес-майнингу.

У першому розділі «Аналіз існуючих методів оцінювання проєктів з розробки програмного забезпечення» конкретизовано поняття оцінки та переліку її складових, а також надано визначення точності та надійності оцінки. Зроблений огляд існуючих методів оцінювання базується на їх класифікації на чотири категорії, кожна з яких відповідає етапу їх виникнення та комплементарна ключовим тенденціям у сфері розробки програмного забезпечення: «класичні» методи (50-80-ті роки, процедурне програмування та водоспадна модель життєвого циклу), «класичні» методи другого покоління (кінець 80-х та початок 2000-х років, об'єктно-орієнтоване програмування та спіральна модель життєвого циклу), agile-методи оцінювання (починаючи з кінця 90-х років і до сьогодні) та методи із застосуванням машинного навчання та штучного інтелекту (починаючи з першої половини 2000-х років та до сьогодні). Проведено огляд здобутків українських науковців та практиків у галузі оцінювання проєктів з розробки

програмного забезпечення на фоні стрімкого розвитку IT-індустрії у нашій країні, починаючи з 90-х років. У першому розділі дисертації надано обґрунтування того, що в контексті організації, яка займається розробкою програмного забезпечення на регулярній основі, оцінювання проєктів з розробки програмного забезпечення є слабоструктурованим бізнес-процесом. Зроблено огляд методів процес-майнингу, призначених для побудови схем таких бізнес-процесів. Проведене дослідження виявило ряд недоліків існуючих методів оцінювання, які суттєво ускладнюють, а часто навіть унеможливлюють їх застосування в сучасних комерційних проєктах. Для усунення виявлених недоліків сформульовано наукову задачу розробити методи оцінювання проєктів з розробки програмного забезпечення та інформаційну технологію, що забезпечує застосування цих методів на практиці.

У другому розділі «Розроблення методів оцінювання проєктів з розробки програмного забезпечення» висвітлені основні наукові результати дисертаційної роботи. Зокрема, розгорнуто теоретичні основи розроблених методів, ключовими з яких є оцінювання в контексті життєвого циклу проєкту з розробки програмного забезпечення, ієрархічна структура елементів оцінювання, структура робочого часу інженера-розробника, підхід «нормалізації» до оцінювання трудоємності, система рівнянь балансу робочого часу проєктної команди. В цьому розділі описаний вперше розроблений метод поетапного оцінювання, застосовний на етапах аналізу та проєктування, який передбачає підготовку попередньої, проміжної та детальної оцінок, кожна наступна з яких є точнішою та детальнішою завдяки поглибленню розуміння вимог до проєкту. Як доповнення до методу поетапного оцінювання, вперше розроблено метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту, який базується на розв’язанні задач цілочисельного програмування з наступним застосуванням методу аналізу ієрархій для ранжування отриманих альтернатив. Розроблено метод побудови розкладу реалізації елементів оцінювання, який є подальшим розвитком методу List Scheduling. Цей метод є комплементарним до

методу поетапного оцінювання та рекомендується до застосування на етапі детального оцінювання. Розклад реалізації елементів оцінювання, побудований із застосуванням розробленого методу, базується на балансуванні нормалізованої оцінки розробки змісту проєктних робіт та нормалізованої спроможності розробки проєктної команди.

У третьому розділі «Оцінювання проєктів з розробки програмного забезпечення як слабоструктурований бізнес-процес» оцінювання розглядається з точки зору його практичного застосування в організації, що на постійній основі займається розробкою програмного забезпечення для власних потреб або на замовлення. За таких умов оцінювання проєктів має повторюваний характер, тобто є бізнес-процесом. Обґрунтовано доцільність врахування слабкої структурованості цього бізнес-процесу, яка забезпечує достатній ступінь гнучкості для ефективного застосування інтуїції, знань та досвіду залучених експертів, що, в свою чергу, дає змогу отримувати унікальні конкурентоспроможні результати. Розроблено концептуальну схему бізнес-процесу оцінювання, яка відображає основні активності, рекомендовану послідовність їх виконання, а також ключові відповідальності виконавців. Також розроблено метод побудови актуальної схеми бізнес-процесу оцінювання, який є подальшим розвитком методу процес-майнингу Fuzzy Miner. Розроблений метод відрізняється від існуючого аналогу можливістю враховувати еволюцію схеми слабоструктурованого бізнес-процесу, а також здатністю опрацьовувати потоки даних. У третьому розділі роботи також описана архітектура та реалізація прототипу модуля програмного забезпечення інформаційної технології, що призначений для моніторингу виконання бізнес-процесу оцінювання, застосовуючи підходи до опрацювання потоків даних та розроблений метод процес-майнингу побудови схеми слабоструктурованого бізнес-процесу.

У четвертому розділі «Аналіз вимог, проєктування архітектури та оцінювання інформаційної технології» представлено інформаційну технологію, що реалізує методи, розроблені в рамках цієї дисертаційної роботи. Аналіз вимог та

проєктування архітектури інформаційної технології здійснено разом із оцінюванням її програмного забезпечення, демонструючи тим самим можливості розроблених методів. Зокрема, підготовано попередню, проміжну та детальну оцінки, а також продемонстровано застосуванням методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту і методу побудови розкладу реалізації задач проєкту. Реалізовано прототип інформаційної технології із застосуванням технологій швидкої розробки Microsoft Power Platform. Відштовхуючись від результатів попереднього оцінювання та нефункційних вимог до інформаційної технології, обґрунтовано вибір сценарію реалізації її повнофункційної версії. Враховуючи сучасні потреби бізнесу, а також тенденцій у сфері розробки програмного забезпечення, окреслено пріоритетні напрямки розвитку цієї інформаційної технології.

Основні наукові результати дисертації опубліковано в 16-ти працях, зокрема: три статті – у наукових фахових періодичних виданнях України; три статті – у наукових періодичних виданнях інших держав; десять публікацій – у матеріалах міжнародних науково-технічних конференцій.

Ключові слова: оцінювання проєктів з розробки програмного забезпечення, ресурсно-часове оцінювання, оцінювання трудовитрат та тривалості, підтримка прийняття рішень, слабоструктурований бізнес-процес, інформаційна технологія.

SUMMARY

Voityshyn V.V. Information technology of software development projects estimation. – Qualifying scientific work on manuscript rights. Dissertation for obtaining the scientific degree of Doctor of Philosophy in specialty 122 «Computer Science». – Lviv Polytechnic National University, Lviv, 2025.

Abstract content. The dissertation research is devoted to creation of the estimation methods applicable on the analysis and design phases of the software development life cycle as well as implementation of the information technology aimed at application of the created estimation methods in practice. The estimation methods and the information technology are supposed to improve the quality of the estimates, reduce efforts required for the estimate preparation, accumulate project historical data as well as ensure handling of the software development projects estimation as a semi-structured business process applying process mining methods.

In the first chapter, “Analysis of the Existing Software Development Project Estimation Methods", the term “Software development project estimate” is clarified, and estimate ingredients are defined. Also, the terms estimate accuracy, reliability, precision, and trueness are explained. Analysis of the existing estimation methods is based on the grouping by the four categories corresponding to the time of the method creation and the main trends in the software development industry: the “classical” methods (the 1950s-1980s, procedural programming and the Waterfall Model of the software development life cycle), the “classical” methods of the second generation (from the late1980s till the beginning of the 2000s, object-oriented programming and the Spiral Model of software development life cycle), the agile estimation methods (from the late 1990s till nowadays), the estimation methods based on machine learning and artificial intelligence (from the early 2000s till nowadays). An overview of the contribution of Ukrainian scientists and practitioners in the estimation field is provided. It is proved that, in the context of a company doing software development on a regular basis, the software development project estimation is a semi-structured business process. An overview of process mining methods aimed at building a scheme of such

business processes is provided in Chapter 1. The research carried out has shown downsides of the existing estimation methods that make their practical use difficult or even hardly possible. In order to eliminate the revealed gaps, it is stated a scientific task to advance new estimation methods as well as develop an information technology aiming at implementation of the created methods in practice.

In the second chapter, “Advancing Methods for Estimating Software Development Projects”, the main scientific results are represented. It includes the theoretical basis of the created methods which key parts are the following: estimation as an integral part of the software development life cycle, the estimable items breakdown structure, the structure of software developer working time, the “normalization” approach to the efforts estimation, the system of working time balance equations. In this chapter, the first developed multi-stage estimation method is represented. The method is applicable in the analysis and design stages. It prescribes preparation of the three subsequent estimates: preliminary, intermediate, and precise. The accuracy and level of detail of these estimates increases along with elaboration of project requirements. In addition to the multi-stage estimation method, the method of decision support on a team composition and a project implementation schedule has been first developed. The method is based on solving integer programming problems with the following application of the analytic hierarchy process to rank the alternatives. The method of building estimable items implementation schedule is a further development of the existing List Scheduling method. This method is complementary to the multi-stage estimation method—it is supposed to be used along with the precise estimation. The method builds an implementation schedule that balances the normalized development estimates on the one hand and the normalized development capacity of a project team on the other.

The third chapter, “Software Development Project Estimation as a Semi-structured Business Process”, is devoted to the implementation of estimation as a business process functioning within an organization that does software development regular basis either for own needs or as a vendor for its clients. Under such

circumstances, software development project estimation has a recurring nature, i.e., it is a business process. The semi-structured nature of the estimation business process ensures enough flexibility for applying intuition, knowledge, and experience of the involved experts which, in turn, allows to achieve unique competitive outcomes. A conceptual schema of the estimation business process was developed. The conceptual schema represents main activities and an approximate order of their execution as well as the key responsibilities of the participants. A method of semi-structured business process schema building has been developed. Being a further development of the process mining method called “Fuzzy Miner”, the method differs from its predecessor in the abilities to handle concept drifts and processing of event data streams. An architecture design and a prototype implementation of a software module with the purpose of monitoring actual execution of the estimation business process are described in the third chapter. The module includes event data streams processing and the developed method of building a schema for a semi-structured business process.

In the fourth chapter, “Requirements Analysis, Architecture Desing, and Estimation of the Information Technology”, it is demonstrated application of the developed methods to estimation of the information technology. Preliminary, intermediate, and precise estimates were prepared. Requirements analysis and architecture design were done along with the multi-stage estimation. The developed method of decision support on the project schedule and team composition as well as the method of building estimable items implementation schedule have also been applied. A prototype version of the information technology was implemented with a low-code technology—Microsoft Power Platform. Based on business needs and current software development trends, further evolutionary steps of the information technology have been outlined.

The main scientific results of the dissertation were published in 16 works, in particular: three articles – in scientific specialized periodicals of Ukraine; three articles – in scientific specialized periodicals of other countries; ten publications – in the proceedings of international scientific and technical conferences.

Key words: software development project estimation, software development effort estimation, decision support, semi-structured business process, information technology.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті в наукових фахових виданнях України:

1. A. Batyuk and V. Voityshyn, "Process Mining: Applied Discipline and Software Implementations," *KPI Science News*, no. 5, pp. 22-36, Nov. 2018. doi: 10.20535/1810-0546.2018.5.146178 [1].
2. A. Batyuk and V. Voityshyn, “Distributed software system with web interface for automated business process discovery,” *Bulletin of Lviv Polytechnic National University: Information Systems and Networks*, vol. 5, pp. 70-77, Jun. 2019. doi: 10.23939/sisn2019.01.070 [2].
3. В. Теслюк, А. Батюк, В. Войтишин, “Метод побудови нормалізованого розкладу реалізації задач проєкту з розробки програмного забезпечення для scrum-команди без диференціації спеціалізацій,” *Український журнал інформаційних технологій*, Т. 6, № 2, С. 11–19, 2024, doi: 10.23939/ujit2024.02.011 [3].

Статті у наукових виданнях інших держав:

4. V. Teslyuk, A. Batyuk, and V. Voityshyn, “Method of Software Development Project Duration Estimation for Scrum Teams with Differentiated Specializations,” *Systems*, vol. 123, no. 10, pp. 1–19, Aug. 2022, doi: 10.3390/systems10040123 [4]. (Scopus, Q2)
5. V. Teslyuk, A. Batyuk, and V. Voityshyn, “Preliminary Estimation for Software Development Projects Empowered with a Method of Recommending Optimal Duration and Team Composition,” *Appl. Syst. Innov.*, vol. 7, no. 3, pp. 1-21, Apr. 2024, doi: 10.3390/asi7030034 [5]. (Scopus, Q2)
6. A. Batyuk and V. Voityshyn, "Software Architecture Design of the Information Technology for Real-Time Business Process Monitoring," *ECONTECHMOD*, vol. 7, no. 3, pp. 13-22, 2018. [6].

Матеріали конференцій:

7. A. Batyuk and V. Voityshyn, "Apache storm based on topology for real-time processing of streaming data from social networks," in *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP'2016)*, Lviv, Ukraine, Aug. 2016, pp. 345-349. doi: 10.1109/DSMP.2016.7583573 [7]. (Scopus)
8. A. Batyuk and V. Voityshyn, "Business Processes Monitoring by Means of Real-Time Visual Dashboards," in *2017 6th International Academic Conference on Information, Communication, Society 2017 (ICS'2017)*, Slavske, Lviv, Ukraine, May 2017, pp. 204-205. [8].
9. A. Batyuk, V. Voityshyn, and V. Verhun, "Software Architecture Design of the Real-Time Processes Monitoring Platform," in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP'2018)*, Lviv, Ukraine, Aug. 2018, pp. 98-101. doi: 10.1109/DSMP.2018.8478589 [9]. (Scopus)
10. A. Batyuk and V. Voityshyn, "Streaming Process Discovery for Lambda Architecture-based Process Monitoring Platform," in *2018 IEEE 13th International Scientific and Technical Conference on Computer Science and Information Technologies (CSIT'2018)*, Lviv, Ukraine, Sep. 2018, pp. 298-301. doi: 10.1109/STC-CSIT.2018.8526592 [10]. (Scopus)
11. A. Batyuk and V. Voityshyn, "Real-Time Process Monitoring Platform: Technical Implementation," in *2018 7th International Academic Conference on Information, Communication, Society 2018 (ICS'2018)*, Chynadiyovo, Lviv, Ukraine, May 2018, pp. 275-276. [11].
12. A. Batyuk and V. Voityshyn, "Real-Time Process Monitoring Platform Based on Streaming Process Discovery Techniques," in *2018 XIV International Scientific Conference on Intellectual Systems of Decision-Making and Problems of Computational Intelligence (ISDMCI'2018)*, Zaliznyi Port, Kherson, Ukraine, May 2018, pp. 7-8. [12].

13. A. Batyuk and V. Voityshyn, "Cloud-based Architecture of the Business Process Monitoring Information Technology," in *2019 8th International Academic Conference on Information, Communication, Society 2019 (ICS'2019)*, Chynadiyovo, Lviv, Ukraine, May 2019, pp. 17-18. [13].
14. A. Batyuk and V. Voityshyn " Streaming Process Discovery Method for Semi-Structured Business Processes," in *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP'2020)*, Lviv, Ukraine, Aug. 2020, pp. 444-448. doi: 10.1109/DSMP47368.2020.9204201 [14]. (Scopus)
15. A. Batyuk and V. Voityshyn, "Process mining-based information technology for operational support of software projects estimation," in *2020 XVI International Scientific Conference on Intellectual Systems of Decision-Making and Problems of Computational Intelligence (ISDMCI'2020)*, Zaliznyi Port, Kherson, Ukraine, 2020, pp. 9-11. [15].
16. V. Teslyuk, A. Batyuk and V. Voityshyn, "Method of Recommending a Scrum Team Composition for Intermediate Estimation of Software Development Projects," in *2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT'2022)*, Nov. 2022, pp. 373-376, doi: 10.1109/CSIT56902.2022.10000432 [16]. (Scopus)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 19

ВСТУП 21

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ОЦІНЮВАННЯ ПРОЄКТІВ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 29

1.1. Оцінка проєкту з розробки програмного забезпечення та її складові .. 29

1.2. Точність та надійність оцінки 31

1.3. Існуючі методи оцінювання проєктів з розробки програмного забезпечення та їх класифікація 33

1.3.1. «Класичні» методи оцінювання 34

1.3.2. «Класичні» методи оцінювання другого покоління 38

1.3.3. Agile-методи оцінювання 41

1.3.4. Методи оцінювання із застосуванням машинного навчання та штучного інтелекту 43

1.4. Огляд здобутків українських науковців та практиків у галузі оцінювання проєктів з розробки програмного забезпечення 46

1.5. Оцінювання проєктів з розробки програмного забезпечення як бізнес-процес 49

1.5.1. Конкретизація поняття «бізнес-процес оцінювання проєктів» .. 49

1.5.2. Оцінювання як слабоструктурований бізнес-процес 52

1.5.3. Застосування процес-майнингу до побудови схем слабоструктурованих бізнес-процесів 55

1.6. Постановка наукових та технічних задач дослідження 58

Висновки до розділу 1 62

РОЗДІЛ 2. РОЗРОБЛЕННЯ МЕТОДІВ ОЦІНЮВАННЯ ПРОЄКТІВ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 63

2.1. Оцінювання в контексті життєвого циклу проєкту з розробки програмного забезпечення 63
2.2. Ієрархічна структура елементів оцінювання 68
2.3. Структура робочого часу інженера-розробника 70
2.4. Оцінювання трудоємності 72
2.5. Графік реалізації проєкту 76
2.6. Проєктна команда 77
2.7. Попереднє оцінювання 80
2.7.1. Попереднє оцінювання та його особливості 80
2.7.2. Система рівнянь балансу робочого часу 81
2.7.3. Оцінювання нормалізованої спроможності розробки 82
2.7.4. Оцінювання тривалості проєкту 83
2.7.5. Оцінювання трудоємності групування за схожістю 83
2.7.6. Коефіцієнт робочого часу розробки 86
2.7.7. Альтернативний підхід до визначення коефіцієнту робочого часу розробки 87
2.7.8. Схема застосування попереднього оцінювання 88
2.7.9. Застереження щодо застосування попереднього оцінювання 91
2.8. Проміжне оцінювання 92
2.8.1. Проміжне оцінювання та його особливості 92
2.8.2. Система рівнянь балансу робочого часу 92
2.8.3. Оцінювання нормалізованої спроможності розробки 95
2.8.4. Оцінювання тривалості проєкту 96
2.8.5. Схема застосування проміжного оцінювання 97
2.8.6. Застереження щодо застосування проміжного оцінювання 99
2.9. Детальне оцінювання 100
2.9.1. Детальне оцінювання та його особливості 100

2.9.2. Схема застосування детального оцінювання 101
2.9.3. Застереження щодо застосування детального оцінювання 102
2.10. Метод побудови розкладу реалізації елементів оцінювання 103
2.10.1. Задача побудови розкладу виконання робіт проєкту 103
2.10.2. Залежності елементів оцінювання 104
2.10.3. Тривалість реалізації елементу оцінювання 106
2.10.4. Розклад реалізації елементів оцінювання 107
2.10.5. Побудова нормалізованого розкладу реалізації елементів оцінювання 109
2.10.6. Застосування методу побудови розкладу реалізації елементів оцінювання 111
2.10.7. Перспективи розвитку методу побудови розкладу реалізації елементів оцінювання 112
2.11. Метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту 113
2.11.1. Визначення цільових функцій 113
2.11.2. Формулювання обмежень 115
2.11.3. Задачі цілочисельного програмування та рекомендації щодо їх розв’язання 116
2.11.4. Ранжування альтернатив 118
2.11.5. Застосування методу підтримки прийняття рішень на різних етапах оцінювання 118
Висновки до розділу 2 119
РОЗДІЛ 3. ОЦІНЮВАННЯ ПРОЄКТІВ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЯК СЛАБОСТРУКТУРОВАНИЙ БІЗНЕС-ПРОЦЕС 121
3.1. Концептуальна схема бізнес-процесу оцінювання 121
3.2. Актуальна схема бізнес-процесу оцінювання 126

3.3. Метод побудови актуальної схеми слабоструктурованого бізнес-процесу оцінювання 127
3.3.1. Постановка задачі 127
3.3.2. Event-дані: основні поняття 128
3.3.3. Оригінальна версія методу Fuzzy Miner 129
3.3.4. Початок та завершення екземплярів бізнес-процесу 130
3.3.5. Еволюція схеми бізнес-процесу 131
3.4. Реалізація методу побудови актуальної схеми слабоструктурованого бізнес-процесу оцінювання 133
3.5. Архітектура модуля RTBPM-E 137
3.5.1. Призначення та функційні можливості модуля RTBPM-E 137
3.5.2. Нефункційні вимоги до модуля RTBPM-E 138
3.5.3. Діаграма компонентів RTBPM-E 140
3.5.4. Технології реалізації RTBPM-E 143
3.5.5. Інтеграція RTBPM-E з інформаційною технологією оцінювання проєктів з розробки програмного забезпечення 145
Висновки до розділу 3 147
РОЗДІЛ 4. АНАЛІЗ ВИМОГ, ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ОЦІНЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ 149
4.1. Завдання інформаційної технології 149
4.2. Сфера застосування інформаційної технології 150
4.3. Користувачі інформаційної технології 150
4.4. Функційні можливості інформаційної технології 150
4.5. Методи, реалізовані в інформаційній технології 154
4.6. Гіпотетичні сценарії реалізації інформаційної технології 154
4.6.1. Сценарій 1: реалізація за підходом «pro-code» 155
4.6.2. Сценарій 2: реалізація за підходом «low-code» 155

4.7. Попереднє оцінювання інформаційної технології 156
4.7.1. Хід попереднього оцінювання 156
4.7.2. Результати попереднього оцінювання 161
4.7.3. Підсумки попереднього оцінювання 161
4.8. Прототип інформаційної технології 163
4.9. Архітектура MVP-версії інформаційної технології 164
4.10. Проміжне оцінювання 167
4.10.1. Хід проміжного оцінювання 167
4.10.2. Підсумки проміжного оцінювання 170
4.11. Детальне оцінювання 171
4.11.1. Хід детального оцінювання 171
4.11.2. Підсумки детального оцінювання 171
4.12. Перспективи розвитку інформаційної технології як програмного продукту 172
Висновки до розділу 4 173
ЗАГАЛЬНІ ВИСНОВКИ ДО РОБОТИ 175
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 178
ДОДАТКИ 197
Додаток 1. Огляд методів оцінювання проєктів з розробки програмного забезпечення 197
Додаток 2. Властивості бізнес-процесів залежно від ступеня структурованості 201
Додаток 3. Етапи оцінювання проєктів з розробки програмного забезпечення 203
Додаток 4. Основні терміни методу поетапного оцінювання проєктів з розробки програмного забезпечення 208

Додаток 5. Атрибути елементів оцінювання 217
Додаток 6. Концептуальна схема слабоструктурованого бізнес-процесу оцінювання проєктів з розробки програмного забезпечення 220
Додаток 7. Характеристики методу побудови схеми слабоструктурованого бізнес-процесу 224
Додаток 8. Структура бази даних реалізації методу побудови схеми слабоструктурованого бізнес-процесу 227
Додаток 9. Попереднє оцінювання програмного забезпечення інформаційної технології 229
Додаток 10. Проміжне оцінювання програмного забезпечення інформаційної технології 247
Додаток 11. Детальне оцінювання програмного забезпечення інформаційної технології 271
Додаток 12. Вихідний код реалізації методу підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди 277
Додаток 13. Акти впровадження результатів дисертаційного дослідження 304

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ЕПЗ – Еквівалент повної зайнятості

ІРСРК – Інженер-розробник середнього рівня компетентності

ІСЕО – Ієрархічна структура елементів оцінювання

ІТ – Інформаційні технології

КЗПА – Коефіцієнт загальних проєктних активностей

КПРК – Коефіцієнт продуктивності за рівнем компетентності

КПСЗ – Коефіцієнт продуктивності за ступенем залученості

КРПРЧ – Коефіцієнт резерву проєктного робочого часу

КРЧР – Коефіцієнт робочого часу розробки

КСАР – Коефіцієнт супровідних активностей розробки

НЕПЗ – Нормалізований еквівалент повної зайнятості

НОР – Нормалізована оцінка розробки

НРЧР – Нормалізований робочий час розробки

НСР – Нормалізована спроможність розробки

ПЗ – Програмне забезпечення

ШІ – Штучний інтелект

AI – Artificial Intelligence

API – Application Programming Interface

BPM – Business Process Management

BPMN – Business Process Modeling and Notation

CMMN – Case Management Model and Notation

COCOMO – Constructive Cost Model

COSMIC – Common Software Measurement International Consortium

CPM – Critical Path Method

CRUD – Create, Read, Update, Delete

ETL – Extract, Transform, Load

FPA – Functional Points Analysis

FTE – Full-Time Equivalent

GERT – Graphical Evaluation and Review Technique

IFPUG – International Function Point Users Group

ISBSG – International Software Benchmarking Standards Group

MVP – Minimum Viable Product

PERT – Project Evaluation and Review Technique

POC – Proof of Concept

RTBPM – Real-Time Business Process Monitoring

RTBPM-E – Real-Time Business Process Monitoring for Estimation

SaaS – System as a Service

SLIM – Software Lifecycle Management

SLOC – Source Lines of Code

SNAP – Software Non-functional Assessment Process

SQL – Structured Query Language

UCP – Use Case Points

UML – Unified Modeling Language

ВСТУП

Актуальність теми. Починаючи з другої половини XX-ого століття, технології розробки програмного забезпечення (ПЗ) динамічно розвиваються, вирішуючи все складніші прикладні задачі. З точки зору бізнесу, одним із ключових аспектів реалізації проєктів з розробки ПЗ є прогнозованість обсягу ресурсів та часу, необхідних для їх успішного виконання. Як показує практика, невірне оцінювання ресурсів та часу часто стає причиною цілого ряду проблем на етапі реалізації і призводить до неуспішного завершення чи навіть провалу проєктів.

Починаючи з 50-х років минулого століття, методи оцінювання проєктів з розробки ПЗ виникали та розвивалися відповідно до еволюції технологій програмування. Такі «класичні» методи оцінювання, як PERT, CPM, метод функційних точок, COCOMO, SLIM були розроблені ще у 50-80 роках. Ці «класичні» методи отримали подальший розвиток у період з другої половини 80-х та до першої половини 2000-х років, давши поштовх для наступної генерації методів оцінювання таких як SNAP, UCP, COCOMO II, FPA, COSMIC тощо. Стрімкий розвиток agile-підходів до розробки, починаючи з другої половини 90-х років, мав наслідком виникнення цілої групи методів оцінювання, найбільш відомими з яких є Planning Poker, Affinity Grouping, T-shirt Sizing. Наступне покоління методів оцінювання почало розвиток із набиранням популярності технологій машинного навчання та ШІ (штучний інтелект). Суть методів цієї групи зводиться до отримання оцінок шляхом застосування методів машинного навчання до накопичених історичних даних. Можна припустити, що наступний виток еволюції методів оцінювання буде базуватися на застосуванні генеративного ШІ (однак на час написання цієї дисертаційної роботи відповідні дослідження перебували ще у зародковому стані).

Ключовою проблемою «класичних» методів оцінювання є їх складність, а також невідповідність сучасним підходами до розробки. Ключовою особливістю agile-методів оцінювання є простота їх застосування та залученість всієї проєктної команди до оцінювання, однак суттєвим недоліком цих методів є те,

що вони призначені для використання на етапі реалізації проєкту і не передбачають можливості застосуванням на етапах аналізу та проєктування (що передують початку реалізації). Головною пересторогою до застосуванням методів, що базуються на технологіях машинного навчання є їх вузька спеціалізованість та неможливість використання в разі відсутності достатнього обсягу історичних даних. Спільним недоліком практично всіх існуючих методів оцінювання є те, що вони застосовні лише на певному етапі життєвого циклу та не забезпечують поетапного оцінювання (із послідовною передачею результатів оцінювання між етапами), комплементарного життєвому циклу розробки ПЗ, починаючи від виникнення ідеї проєкту аж до завершення його реалізації.

Важливим фактором ефективності практичного застосування методів оцінювання є їх інтеграція у бізнес-процеси організації, що розробляє ПЗ для власних потреб або на замовлення. Оскільки оцінювання великою мірою залежить від знань та досвіду залучених експертів, а також вимагає певного ступеня слободи та гнучкості, то відповідний бізнес-процес відносять до категорії слабоструктурованих. Бізнес-процес оцінювання вимагає залучення експертів високого рівня кваліфікації та може бути досить трудомістким і навіть тривалим у часі, що може навіть призвести до відчутних витрат бюджету проєкту ще на етапах аналізу та проєктування, а також знижує вірогідність «виграшу» проекту, якщо оцінювання триває довше ніж у конкурентів.

Значний внесок в теорію та практику оцінювання проєктів з розробки ПЗ зробили: B.W. Boehm – розробник методів Wideband Delphi, COCOMO та COCOMO II, автор спіральної моделі життєвого циклу проєкту, один із перших, хто застосував ідею «конусу невизначеності» до проєктів з розробки ПЗ; C.E. Clark – розробник методу PERT; J.E. Kelley and M.R. Walker – автори методу критичного шляху (CPM); J.M. Gorey – запропонував поділ оцінок на чотири типи залежно від мети оцінювання, рівня деталізації та точності; A.J. Albrecht – розробник методу функційних точок, основоположник ідеї вимірювання «розміру» ПЗ у «точкових» одиницях; L.H. Putnam – розробник Putnam-моделі, що була

покладена в основу набору засобів оцінювання SLIM; J.W. Grenning – автор методу Planning Poker; A. Trendowicz – автор методів оцінювання, що базуються на поєднанні аналізу даних та експертних оцінок; М.О. Сидоров – одним із перших серед українських вчених розпочав дослідження методів оцінювання ПЗ, автор навчальних дисциплін, пов'язаних із економікою розробки ПЗ; С.Б. Приходько – автор методів оцінювання, що базуються на нелінійних регресійних моделях; Д.В. Баценко – розробник методу калібрування моделі COCOMO шляхом редукції основного рівняння; І.В. Ізонін – автор методів оцінювання із застосуванням машинного навчання.

Отже, актуальною науковою задачею є розробка методів оцінювання проєктів з розробки ПЗ, які б послідовно охоплювали етапи життєвого циклу проєкту, що передують початку реалізації, а також враховували agile-природу підходів до розробки під час реалізації проєкту. Також актуальною науковою задачею є розробка інформаційної технології оцінювання проєктів з розробки ПЗ, що забезпечує застосування розроблених методів оцінювання на практиці.

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

Дисертація виконувалася відповідно до пріоритетних напрямів науково-дослідних робіт Національного університету «Львівська політехніка» та відповідно до координаційних планів Міністерства освіти й науки України. Зокрема, наукові дослідження виконувалися в рамках держбюджетних наукових тем кафедри автоматизованих систем управління: «Інтелектуальні інформаційні технології багаторівневого управління енергоефективністю регіону» (номер держреєстрації 0117U004450, роки виконання: 2017-2018); «Методи та засоби нейронечіткого управління групою мобільних робототехнічних платформ» (номер держреєстрації: 0123U101688, роки виконання: 2023-2024); «Методи та засоби інтелектуального вимірювання параметрів руху та визначення просторової орієнтації наземних мобільних робототехнічних платформ» (номер держреєстрації: 0124U000822, роки виконання: 2024-2026). Документи, що підтверджують впровадження результатів дисертаційного дослідження надано у додатку 13.

Метою дисертаційної роботи є підвищити точність оцінок проєктів з розробки ПЗ шляхом розроблення нових методів оцінювання та інформаційної технології, що забезпечує застосування цих методів на практиці.

Для розв'язання поставленої наукової задачі необхідно виконати такі завдання:

1. Конкретизувати поняття оцінки проєкту з розробки ПЗ та визначити складові цієї оцінки. Проаналізувати існуючі методи оцінювання проектів з розробки ПЗ та ідентифікувати основні проблеми, а також актуальні напрямки наукових досліджень в цій галузі. Обґрунтувати, що бізнес-процес оцінювання належить до категорії слабоструктурованих бізнес-процесів. Провести огляд методів процес-майнингу, що призначені для побудови схем слабоструктурованих бізнес-процесів.

2. Розробити метод поетапного оцінювання проєктів з розробки ПЗ, застосовний на етапах аналізу та проєктування, який забезпечує поступове підвищення рівня деталізації та точності оцінки, а також враховує agile-природу підходів до розробки на етапі реалізації проєкту.

3. Розробити метод побудови розкладу реалізації задач проєкту, відштовхуючись від оцінки трудоємності цих задач та спроможності розробки проєктної команди.

4. Розробити метод підтримки прийняття рішень, метою якого є надання переліку альтернатив складу команди та графіку реалізації проєкту, за яким експерти з оцінювання отримують змогу обрати найбільш вдалі, з їх точки зору, варіанти.

5. Розробити метод побудови схеми слабоструктурованого бізнес-процесу оцінювання, що базується на застосуванні методів процес-майнингу.

6. Провести аналіз вимог та спроєктувати архітектуру інформаційної оцінювання проектів з розробки ПЗ, що забезпечує практичне застосування розроблених методів оцінювання, враховує слабку структурованість відповідного бізнес-процесу, а також реалізує базу даних проєктів та їх оцінок. Проде-

монструвати можливості розроблених методів, застосувавши їх до оцінювання ПЗ цієї інформаційної технології.

Об'єкт дослідження – оцінювання проєктів з розробки програмного забезпечення.

Предмет дослідження – методи і засоби інформаційної технології оцінювання проєктів з розробки програмного забезпечення.

Методи дослідження: математичні методи дослідження операцій (зокрема, методи розв'язання задач цілочисельного програмування та метод аналізу ієрархій), методи теорії розкладів, методи процес-майнингу, методи проєктування архітектури ІТ-систем, методи об'єктно-орієнтованого аналізу та проєктування.

Наукова новизна одержаних результатів:

вперше розроблено:

– метод поетапного оцінювання проєктів з розробки ПЗ, що передбачає послідовну підготовку попередньої, проміжної та детальної оцінок, поступово підвищуючи рівень деталізації та точність оцінки по мірі поглиблення розуміння вимог до проєкту, а також враховуючи agile-природу підходів, використовуваних на етапі реалізації проєкту;

– метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту, що базується на розв'язанні задач цілочисельного програмування та ранжуванні альтернатив зі застосуванням методу аналізу ієрархій, який у поєднанні з методом поетапного оцінювання дає змогу підвищити ефективність роботи експертів над оцінкою;

отримали подальший розвиток:

– метод List Scheduling, використаний для побудови графіку реалізації елементів оцінювання, враховуючи умову балансу між нормалізованою оцінкою розробки змісту проєктних робіт та нормалізованою спроможністю розробки проєктної команди; сформульовано критерій якості графіку реалізації елементів оцінювання, який базується на мінімізації нормалізованого часу простою проєктної команди;

– метод Fuzzy Miner, до основної здатності якого будувати схеми слабоструктурованих бізнес-процесів додано також можливості опрацювання потоків даних та врахування еволюції схеми бізнес-процесу, що дає змогу підтримувати схему бізнес-процесу в актуальному стані за умов постійного надходження даних про його реальний перебіг.

Практичне значення одержаних результатів.

Метод поетапного оцінювання та метод побудови розкладу реалізації елементів оцінювання готові до практичного застосування. Завдяки своїй простоті, ці методи дозволяють навіть найпростішу програмну реалізацію у табличному редакторі. Для кожного із трьох етапів оцінювання: попереднього, проміжного, детального – розроблено схеми застосування (із відповідними візуалізаціями у вигляді діаграм активностей UML), а також визначено умови, за яких застосування відповідного етапу оцінювання не рекомендовано.

Метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту готовий до практичного застосування у поєднанні із методом поетапного оцінювання. Надано рекомендації щодо його програмної реалізації (зокрема, запропоновано бібліотеки для розв’язування задачі цілочисельного програмування). Програмна реалізація цього методу у вигляді Python-скрипта, готова до практичного використання, представлена у додатку 12.

Розроблена архітектура та підготована оцінка інформаційної технології можуть бути взяті за основу для реалізації програмного продукту метою якого є автоматизація бізнес-процесу оцінювання проєктів з розробки ПЗ (в рамках цієї дисертаційної роботи розроблено прототип відповідного програмного забезпечення).

Результати дисертаційних досліджень впроваджені в навчальний процес кафедри автоматизованих систем управління Національного університету «Львівська політехніка» при викладанні наступних дисциплін: «Командна робота та презентаційні навички» та «Математичні методи дослідження операцій» (підтверджено актом впровадження у додатку 13).

Особистий внесок здобувача.

Усі наукові результати дисертаційної роботи отримані автором самостійно. Особисто здобувачеві належать такі наукові результати: проведено аналіз існуючих методів оцінювання проєктів з розробки ПЗ [4]; розроблено метод поетапного оцінювання, застосовний під час аналізу та проєктування, що передбачає підготовку попередньої, проміжної та детальної оцінок [4], [5], [16]; розроблено метод побудови розкладу реалізації елементів оцінювання, що базується на балансуванні нормалізованої оцінки змісту проєктних робіт та нормалізованої спроможності розробки команди [3]; розроблено метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту [5], [16]; розроблено метод побудови схеми слабоструктурованого бізнес-процесу оцінювання, що забезпечує обробку потоку даних та враховує еволюцію схеми бізнес-процесу [1], [2], [6], [9], [10], [14].

Апробація результатів дисертації.

Основні наукові та практичні результати роботи доповідались та обговорювались на міжнародних наукових та науково-технічних конференціях: 1-ій міжнародній конференції «IEEE Data Stream Mining & Processing 2016» (DSMP’2016), 6-ій міжнародній науковій конференції «Інформація, комунікація, суспільство 2017» (ICS’2017), 2-ій міжнародній конференції «IEEE Data Stream Mining & Processing 2018» (DSMP’2018), 13-ій міжнародній науково-технічній конференції «IEEE Computer Science and Information Technologies 2018» (CSIT’2018), 7-ій міжнародній науковій конференції «Інформація, комунікація, суспільство 2018» (ICS’2018), 14-ій міжнародній науковій конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту 2018» (ISDMCI’2018), 8-ій міжнародній науковій конференції «Інформація, комунікація, суспільство 2019» (ICS’2019), 3-ій міжнародній конференції «IEEE Data Stream Mining & Processing 2020» (DSMP’2020), 16-ій міжнародній науковій конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту 2020» (ISDMCI’2020), 17-ій міжнародній науково-технічній

конференції «IEEE Computer Science and Information Technologies 2022» (CSIT’2022).

Матеріали дисертації неодноразово доповідалися та обговорювалися на наукових семінарах кафедри автоматизованих систем управління Національного університету «Львівська політехніка».

Структура та обсяг дисертації.

Дисертаційна робота складається із вступу, чотирьох розділів, висновків, списку літератури з 154 найменувань та 13 додатків. Загальний обсяг дисертації становить 307 сторінок, з них 177 сторінок основного тексту, який містить 15 рисунків та 14 таблиць.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ОЦІНЮВАННЯ ПРОЄКТІВ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Оцінка проєкту з розробки програмного забезпечення та її складові

Зазвичай, проєкти з розробки ПЗ включають значну частку невизначеності, що зумовлено інноваційною природою цих проєктів та динамічним бізнес-середовищем (через що, зокрема, вимоги до них постійно зазнають змін). Не зважаючи на значний ступінь ризику, пов'язаного з проєктами з розробки ПЗ, для бізнесу критично важливою залишається прогнозованість на кожному з етапів життєвого циклу таких проєктів. З метою забезпечення цієї прогнозованості і проводиться оцінювання. В україномовній науковій літературі вживаються такі терміни як «ресурсно-часове оцінювання» [17] або «оцінювання трудовитрат та тривалості» [18]. Англомовним еквівалентом терміну «оцінювання» є «estimation», відповідно «оцінка» – «estimate»; часто, особливо у наукових публікаціях, можна зустріти більш повні варіанти – «software development project estimation» (SDPE) або «software development effort estimation» (SDEE).

В рамках цієї дисертаційної роботи під *оцінкою проєкту з розробки ПЗ розумітимемо прогноз ресурсів та часу, необхідних для його реалізації*. В цій дисертаційній роботі опрацьовуються наступні складові оцінки:

1. *Зміст проєктних робіт* є деталізацією того, що повинно бути виконано під час реалізації проєкту. Як правило, зміст проєктних робіт представляється у вигляді ієрархічної структури робіт (англ. «work breakdown structure», WBS), так званого проєктного «беклогу», специфікації вимог тощо. Розроблений в рамках дисертаційної роботи підхід до представлення змісту проєктних робіт описано у підрозділі 2.2.

2. *Оцінка трудоємності проєктних робіт* є так званим «розміром» змісту проєктних робіт, що не залежить від складу команди та графіку реалізації проєкту. На практиці такий «розмір» визначають, використовуючи кількість

логічних ліній вихідного коду (SLOC-метрика, англ. «Source Lines of Code»), функційні точки (або інші «точкові» одиниці), часові одиниці (наприклад, людино-години). Розроблений підхід «нормалізації» до оцінювання трудоємності представлено у підрозділі 2.4.

3. *Склад проєктної команди*, як правило, має значний вплив на собівартість та графік реалізації проєкту. Теоретичні основи, необхідні для роботи із складом команди, представлені у підрозділі 2.6.

4. *Графік реалізації проєкту* є поділом реалізації проєкту на часові проміжки, наприклад, фази, кожна з яких, в свою чергу, ділиться на спринти (якщо реалізація здійснюється із використанням agile-підходів). Тривалість проєкту є сумою тривалостей цих часових проміжків. Більше інформації про графік реалізації проєкту представлено у підрозділі 2.5.

5. *Розклад реалізації задач проєкту* є часовим розподілом реалізації задач проєкту в межах його графіку. У підрозділі 2.10 подано розроблений метод побудови розкладу реалізації задач проєкту.

До оцінки проєкту з розробки ПЗ також належать складові, які не входять до завдань цієї дисертаційної роботи:

6. *Оцінка ризиків*, що включає ідентифікацію ризиків, визначення ступеня їх серйозності та ймовірності виникнення, прогнозування їх впливу на перебіг реалізації проєкту.

7. *Оцінка інших ресурсів* включає прогнозування вартості ресурсів, необхідних для реалізації проєкту та подальшої виробничої експлуатації розробленого ПЗ. Сюди, для прикладу, входять ліцензії на програмне забезпечення, підписки на «хмарні» сервіси, необхідне апаратне забезпечення тощо.

8. *Оцінка бюджету* є однією із найбільш важливих складових оцінки проєкту. Основними вхідними даними для розрахунку бюджету є оцінка трудоємності проєктних робіт (пункт 2), склад команди (пункт 3), графік реалізації проєкту (пункт 4). Зазвичай, у випадку великих проєктів для підготовки бюджету залучаються експерти-фінансисти.

9. *Зворотний зв'язок щодо оцінки* може бути отриманий як до початку реалізації проєкту, так і підчас та після її завершення. Наприклад, зворотний зв'язок від проєктної команди під час реалізації проєкту важливий з точки зору моніторингу відхилень затрачених ресурсів та часу від початкової оцінки. Важливість зворотного зв'язку після завершення реалізації проєкту полягає, зокрема, в тому що це дає змогу визначити точність та слабкі сторони оцінки, наданої ще до початку реалізації.

Вище представлено перелік основних найбільш вживаних складових оцінки проєкту. Однак, варто підкреслити, що для конкретних практичних задач складові оцінки, ймовірно, відрізнятимуться від цього переліку: певні елементи можуть видалятися, а також можуть додаватися й інші елементи (наприклад, оцінка очікуваних вигод від реалізації проєкту, рентабельність інвестицій тощо) [19].

Варто підкреслити, що оцінювання є актуальною задачею на кожному з етапів життєвого циклу проєкту. Залежно від цілей оцінювання та етапу життєвого циклу складові оцінки можуть мати різний формат та ступінь деталізації. Більше деталей щодо різновидів оцінок в контексті життєвого циклу проєкту представлено у підрозділі 2.1 та додатку З.

1.2. Точність та надійність оцінки

Як показано у підрозділі 1.1, оцінка проєкту з розробки ПЗ включає цілий набір складових, окремі з яких є числовими значеннями, наприклад: оцінка трудоємності проєктних робіт або тривалість реалізації проєкту. У цьому підрозділі дамо визначенням поняттям точності та надійності, застосовні до складових оцінки, які мають числові значення.

Точністю оцінки (англ. «accuracy») є міра її близькості до відповідного актуального значення, при цьому останнє визначається після реалізації проєкту (або його частини) [19], [20], [21]. Очевидним недоліком такого визначення

точності оцінки є те, що її значення не можна отримати на етапі підготовки оцінки (тобто до завершення реалізації).

Надійністю оцінки (англ. «reliability») називають міру близькості значень оцінок, отриманих різними методами або від різних експертів [19], [21]. Іншими словами, надійність оцінки характеризує її «повторюваність» (англ. «repeat-ability»). У англомовній літературі міру близькості різних значень оцінки для одного і того ж об'єкту ще називають «precision» (що дослівно перекладається як «точність», не плутати із «точністю» в значенні «accuracy»). Користуючись методами математичної статистики, значення надійності оцінки може бути задано як дисперсія або середнє квадратичне відхилення. Як видно із визначення, надійність оцінки (на відміну від точності) можна отримати безпосередньо під час оцінювання.

Точність та надійність оцінки об'єднуються у понятті *правдивості оцінки* (англ. «trueness»), що є мірою близькості середнього значення оцінок, отриманих різними методами або від різних експертів, та відповідного актуального значення [21].

Оскільки оцінка за своїм визначенням є прогнозом, то її значення зручно задавати не одним числом, а у вигляді інтервалу від мінімального до максимального значення. При цьому на етапі оцінювання вважається, що актуальне значення потрапить у заданий інтервал з ймовірністю близькою до 1. Чим вища «впевненість» (англ. «confidence») у точності оцінки, тим менша різниця між максимальним та мінімальним значеннями цього інтервалу. Іншими словами, інтервал значень оцінки може слугувати своєрідним виразом «оцінки» її точності. Прикладом підходу інтервального оцінювання може слугувати 3-точкова оцінка методу PERT [22], [23], де межі інтервалу визначаються як оптимістичне та песимістичне значення оцінки.

1.3. Існуючі методи оцінювання проєктів з розробки програмного забезпечення та їх класифікація

Виникнення та еволюція методів оцінювання відбувалася паралельно з розвитком технологій розробки ПЗ, починаючи з другої половини XX-ого століття. Існуючі методи оцінювання класифікуватимемо за категоріями, що відповідають етапам їх еволюції.

1. Так звані *«класичні» методи* були розроблені у 1950-1980-х роках XX-ого століття. Методи цієї категорії набули широкого поширення і активно використовуються навіть зараз. Одним із найбільш відомих представників цієї категорії є COCOMO [24]. До цих методів, зокрема, належать PERT [22], [23] та CPM [25], які початково розроблялися для інших галузей і досить швидко були адаптовані до проєктів з розробки ПЗ. Важливою особливістю цих методів є те, що їх ідеї мали значний вплив на розвиток методів оцінювання наступних поколінь: наприклад, застосування функційних точок [26], [27] до вимірювання трудоємності отримало подальший розвиток у ряді методів, зокрема, COCOMO II [28], UCP [29] та agile-методах [30], [31].

2. *«Класичні» методи другого покоління* були розроблені у другій половині 80-х та першій половині 2000-х років. Ці методи були подальшим розвитком, зокрема, таких «класичних» методів першого покоління як COCOMO [24] та методу функційних точок [26], [27].

3. *Agile-методи оцінювання* розпочали свій розвиток з кінця 90-х років з поширенням agile-підходів до реалізації проєктів з розробки ПЗ. Ключовими відмінностями цих методів від «класичних» є простота їх застосування та те, що вони призначені для використання командою на етапі реалізації проєкту.

4. *Методи оцінювання із застосуванням машинного навчання та штучного інтелекту (ШІ)* почали набувати поширення з початку 2000-х років. Їх головною особливістю є необхідність використання історичних даних, до яких застосовуються ті чи інші методи машинного навчання.

З іншими підходами до класифікації методів оцінювання можна ознайомитися у працях [32], [33], [34], [35], [36], [37].

Аналіз існуючих методів оцінювання проводитимемо, опираючись на наступні критерії:

1) етапи життєвого циклу проєкту, на яких застосовується метод;

2) одиниці вимірювання трудоємності;

3) використання історичних даних;

4) залежність від експертних оцінок;

5) необхідний рівень компетентності експертів, які використовують метод;

6) можливість застосування до сучасних agile-процесів розробки.

Проаналізовані існуючі методи оцінювання та їх ключові характеристики представлені у додатку 1.

1.3.1. «Класичні» методи оцінювання

«Класичними» називатимемо методи оцінювання, що виникли у 50-80 рр. XX-ого століття. Поштовхом до появи та еволюції цих методів стало зародження та подальший стрімкий розвиток технологій розробки ПЗ. Значимість «класичних» методів полягає, зокрема, в тому, що їхні ідеї лягли в основу багатьох наступних методів оцінювання і знаходять своє застосування на практиці навіть зараз.

Одним із найперших та найбільш відомих методів планування проєктів став PERT (англ. «Program/Project Evaluation and Review Technique»), який був розроблений військово-морськими силами США у 1958 р. [22], [23] і згодом адаптований іншими індустріями, зокрема, розробкою ПЗ. Метод поєднує так звану 3-точкову експертну оцінку (що включає оптимістичне, песимістичне і найбільш ймовірне значення) та статистичні методи, що дають змогу отримати фінальне значення оцінки. PERT також включає інструментарій моніторингу процесу виконання проєкту, даючи змогу корегувати план відповідно до актуального стану реалізації проєкту. Не зважаючи на те, що PERT був розроблений близько 70-ти років тому, його 3-точкова експертна оцінка широко застосо-

вується й сьогодні [38]. Перевагами цього методу є також відсутність необхідності опиратися на історичні дані та простота практичного використання. Суттєвим недоліком оригінальної версії PERT при його застосуванні до сучасних проєктів з розробки ПЗ є те, що він не дає змоги враховувати agile-природу процесів розробки.

Іншим відомим методом, що часто застосовується у поєднанні з PERT, є метод критичного шляху (англ. «critical path method», CPM) [25]. Як і PERT, CPM був розроблений у кінці 50-х рр. Його головне завдання полягає в оцінюванні тривалості проєкту як тривалості «найдовшого» ланцюжка проєктних задач (який і називають критичним шляхом). Аналогічно до PERT, CPM вимагає досить детальної декомпозиції змісту проєктних робіт, а також визначення залежностей між задачами. Останній факт унеможливлює його застосування на ранніх етапах аналізу та проєктування, але робить його корисним під час надання оцінок на завершальних етапах аналізу та проєктування (перед початком реалізації проєкту).

В цьому контексті варто згадати також метод GERT (англ. «Graphical Evaluation and Review Technique») [39], розроблений у середині 60-х рр., який базується на більш складному мережному аналізі у порівнянні з PERT та CPM. Цей метод враховує ймовірнісну природу мережної діаграми, що відображає залежності між проєктними задачами. На практиці GERT застосовують до планування проєктів із складними залежностями між задачами.

Опублікована у 1978 р. модель Ларі Путнама (англ. «Larry Putnam») [40], [41] стала одним із перших підходів до оцінювання проєктів, що базуються на математичній моделі. Ця математична модель, застосовуючи розподіл Рейлі, встановлює залежності між «розміром» ПЗ (вираженим як кількість ліній коду), трудоємністю (вираженою у людино-годинах) та тривалістю проєкту. Відповідний метод оцінювання проєктів отримав назву SLIM (англ. «Software Lifecycle Management»). Вже у наступному, 1979 р., компанія QSM (англ. «Quantitative Software Management») випустила першу версію комерційного програмного за-

безпечення, що базувалося на методі SLIM, а також включало Монте-Карло симуляції для аналізу ризиків та методи лінійного програмування для планування ресурсів. На сьогодні, після 45-ти років роботи на ринку, компанія QSM пропонує один із найбільш потужних комплексів комерційних програмних продуктів для оцінювання проєктів з розробки ПЗ, що, зокрема, включає: базу даних із більш як 14 тисяч виконаних проєктів, широкий спектр інструментів оцінювання проєктів, інструменти контролю виконання проєктів із можливостями моделювання перебігу їх подальшої реалізації, засоби для роботи з портфоліо проєктів (https://www.qsm.com/tools, дата звернення: 06.03.2025).

Метод функційних точок (англ. «Function Points»), розроблений Аланом Альбрехтом (англ. «Allan Albrecht»), був вперше опублікований у 1979 р. [26]. Ключова ідея методу полягає у вираженні функційного розміру (англ. «functional size measurement», FSM) ПЗ (тобто кількості бізнес-функцій) у так званих «функційних точках», які згідно [26] визначаються як сума зважених значень функцій даних та функції транзакцій, помножена на коефіцієнт складності. У [27] на основі аналізу 24-ох проєктів, що виконувалися компанією IBM, показана кореляція між функційними точками та кількістю логічних ліній коду (SLOC-метрикою), а також трудоємністю, вираженою у людино-годинах. Як наслідок, був запропонований двокроковий підхід до оцінювання: спершу на основі кількості функційних точок визначається значення SLOC-метрики, а тоді із значення SLOC-метрики отримується трудоємність у людино-годинах. Не зважаючи на те, що метод функційних точок у своєму початковому вигляді досить складно застосовувати до оцінювання сучасних проєктів, ідея функційних точок отримала подальший розвиток у багатьох інших методах оцінювання, зокрема, COSMIC, COCOMO II, Planning Poker, тощо. У 1986 р. була створена неприбуткова організація International Function Point Users Group, IFPUG (https://ifpug.org/, дата звернення: 01.03.2025), метою якої є розробка та просування методів визначення «розміру» ПЗ, застосовуючи «точкові» одиниці. В рамках цієї

дисертаційної роботи ідея «точкових» одиниць знайшла застосування на етапі попереднього оцінювання (див. пункт 2.7.5).

Одним із найбільш відомих методів оцінювання був та залишається на сьогодні COCOMO (англ. «Constructive Cost Model») [24], [42], який був розроблений Баррі Боемом (англ. «Barry Boehm») упродовж 70-х рр. та вперше опублікований на початку 80-х рр. Цей метод давав змогу оцінювати трудоємність, розмір проєктної команди та тривалість проєкту, відштовхуючись від кількості ліній коду (SLOC-метрика). Беззаперечно сильною стороною COCOMO є те, що він базується на математичній моделі, яка була побудована за результатами аналізу 63-ох проєктів з розробки ПЗ. Метод пропонує три моделі залежно від рівня деталізації та вимог щодо точності оцінки: базову, проміжну та детальну. Метод також враховує рівень складності проєкту та рівень компетентності проєктної команди. Використання кількості ліній програмного коду для вираження «розміру» ПЗ, очевидно, залежить від мови та технологій програмування та вимагає наявності історичних даних. Не зважаючи на те, що використання оцінки кількості ліній коду було і залишається дискусійним питанням, цей підхід отримав широке практичне застосування та подальший у розвиток у наступній версії цього методу – COCOMO II [28].

Серед «класичних» методів оцінювання варто також відзначити Wideband Delphi [24], [37], який був розроблений у 1970-х рр. як адаптація до сфери розробки ПЗ існуючого на той час Delphi-методу. Суть цього методу полягає в залученні команди експертів, кожен з учасників якої готує свою оцінку, після чого всі експерти збираються та обговорюють отримані оцінки. Такий ітеративний процес оцінювання повторюється доти, доки не буде досягнуто консенсусу між експертами. Однією із особливостей цього методу є відсутність необхідності у високій деталізації задач. Wideband Delphi отримав подальший розвиток у agile-методах оцінювання, зокрема, його ідея лягла в основу широко відомого методу Planning Poker [43], який призначений для використання agile-командами (див. пункт 1.3.3).

1.3.2. «Класичні» методи оцінювання другого покоління

Розробка та активне використання «класичних» методів оцінювання другого покоління припадає на період з другої половини 80-х та до початку 2000-х рр. Ці методи є подальшим розвитком методів попередньої генерації у відповідності до потреб розробки ПЗ того часу, базуючись на досвіді оцінювання проєктів, набутому упродовж попередніх десятиліть. Варто відзначити, що ця група методів оцінювання значною мірою орієнтується на водоспадну та спіральну моделі життєвого циклу проєкту, які були домінуючими до широкого поширення agile-підходів.

Наступним кроком розвитку методу функційних точок Алана Альбрехта [26], [27] став метод FPA (англ. «Function Point Analysis»), розроблений організацією IFPUG [44]. На сьогодні метод FPA представлений як стандарт ISO/IEC 20926:2009. Варто підкреслити, що метод FPA, головним чином, враховує функційні вимоги до проєкту, однак на практиці нерідко трапляються ситуації, коли нефункційні вимоги значно впливають на обсяги ресурсів та часу, необхідних для реалізації проєкту. Для таких випадків організацією IFPUG розроблено метод SNAP (англ. «Software Non-functional Assessment Process») [45], рекомендації щодо застосування якого представлено у стандарті ISO/IEC/IEEE 32430:2025. Згідно з баченням IFPUG, застосування методів FPA та SNAP у поєднанні дає змогу найбільш повно оцінити проєкт, враховуючи як функційні, так і нефункційні вимоги.

В цьому контексті не можна обійти увагою ще один метод оцінювання, який також є подальшим розвитком методу функційних точок – COSMIC (англ. «Common Software Measurement International Consortium»), що був розроблений у кінці 90-х рр. (https://cosmic-sizing.org/, дата звернення: 06.03.2025). На сьогодні рекомендації щодо застосування COSMIC представлені у вигляді стандарту ISO/IEC 19761:2011. Цей метод застосовний до оцінювання функційного розміру як бізнес-застосувань, так і ПЗ, що функціонує в режимі реального часу.

Незаперечною перевагою COSMIC є можливість його використання на проєктах, які реалізуються згідно з agile-підходами.

Одним із стимулів до подальшого розвитку методу функційних точок стало поширення об'єктно-орієнтованого аналізу та проєктування і пов'язане з цим виникнення мови візуального моделювання UML (англ. «Unified Modeling Language»). Метод UCP (англ. «Use Case Points») [29], [34], [46] передбачає визначення «розміру» ПЗ, застосовуючи так звані «use-case»-точки, які розраховуються на основі UML-діаграм прецедентів (англ. «UML use case diagrams»). Очевидно, що сильною стороною UCP є аналіз змісту проєктних робіт із застосуванням відносно простої візуалізації – UML-діаграм прецедентів. Що, в свою чергу, відкриває можливості застосування цього методу на ранніх етапах аналізу вимог. Однак, варто відзначити, що формули розрахунку «use-case»-точок, запропоновані в оригінальній версії методу [29], потребують певного переосмислення перед застосуванням до оцінювання сучасних проєктів.

Спільною рисою кожного із вище згаданих методів: FPA, SNAP, COSMIC, UCP – є необхідність конвертування «розміру» ПЗ, вираженого у «точкових» одиницях, у трудоємність, вимірювану у людино-годинах. Одним із найбільш поширених підходів до вирішення цієї задачі є використання історичних даних. Наприклад, для COSMIC можна застосувати базу даних ISBSG (англ. «International Software Benchmarking Standards Group»), що дає змогу отримати трудоємність для однієї функційної точки на основі накопичених даних про проєкти, «схожі» до того, який оцінюється [47].

Опублікований на початку 2000-х рр. метод COCOMO II [28] є подальшим розвитком його попередньої версії – COCOMO 81 [24]. Під час розробки COCOMO II було проаналізовано 163 проєкти з розробки ПЗ. На відміну від попередньої версії, COCOMO II включає три підходи до оцінювання залежно від етапу життєвого циклу проєкту:

1) на етапі *прототипування* використовується так званий «application composition»-підхід, який базується на застосуванні «application»-точок (що є по-

дальшим розвитком ідеї функційних точок [26], [27] та похідних від неї об'єктних точок [48]);

2) *рання фаза проєктування* (англ. «early design phase») передбачає застосування підходу, що включає меншу кількість параметрів (в порівнянні з третім етапом) і забезпечує дещо нижчу точність оцінки; визначення «розміру» ПЗ здійснюється з використанням кількості ліній коду (SLOC-метрика) або функційних точок (англ. повна назва – «unadjusted function points», UFP);

3) на *етапі завершення проєктування* (англ. «post architecture design phase»), що передує початку етапу реалізації проєкту, застосовний такий самий підхід до оцінювання, як і на другому етапі, але з більшою кількістю параметрів, що дає змогу підготувати оцінку з вищим ступенем точності; як і на другому етапі, для визначення «розміру» ПЗ використовуються SLOC-метрика або функційні точки.

COCOMO II є одним із тих методів оцінювання, що отримали найбільш повне наукове обґрунтування. Не зважаючи на те, що COCOMO II був розроблений майже 30 років тому, він не втрачає актуальності й сьогодні. Слабкою стороною цього методу є те, що він був створений до виникнення та широкого поширення agile-процесів розробки, що, як наслідок, вимагає його адаптації до сучасних проєктів, більшість з яких реалізовуються згідно з agile-підходами. Перешкодами до практичного застосування COCOMO II також є його складність (що, зокрема, вимагає спеціального навчання) та необхідність у історичних даних для калібрування значень параметрів.

Однією із найбільш відомих і найстаріших реалізацій COCOMO є комерційний програмний продукт SystemStar, що підтримує як COCOMO II [28], так і COCOMO 81 [24], [42], а також метод COSYSMO (англ. «Constructive Systems Engineering Cost Model») [49], котрий належить до сімейства методів COCOMO (https://www.softstarsystems.com/, дата звернення: 26.02.2025).

1.3.3. Agile-методи оцінювання

Agile-підходи до розробки ПЗ почали набувати популярності з другої половини 90-х років як альтернатива домінуючим на той час процесам, що базувалися на водоспадній та спіральній моделях життєвого циклу проєкту. Головним завдання agile-підходів було зменшення регламентованості процесів розробки та підвищення гнучкості у плануванні, що мало підвищити швидкість розробки і знизити ймовірність перевищення бюджету та часу реалізації проєктів. Одним з перших agile-підходів було так зване «екстремальне програмування», детально описане К. Беком у його відомій праці [50], вперше опублікованій у 1999 р. Офіційною датою початку використання agile вважається 11-13 лютого 2001 р., коли група із 17 експертів-практиків у галузі розробки ПЗ сформулювала широко відомий agile-маніфест (англ. «The Agile Manifesto»), який декларував 4 цінності та 12 принципів [51].

Серед методів оцінювання, застосовних до agile-підходів розробки, варто виділити наступні (нижче будуть використовуватися англомовні назви методів, за якими вони відомі у науковій літературі та серед розробників-практиків):

1. *Planning Poker* [30], [43], [52], [53] передбачає процедуру, коли кожен з учасників команди пропонує свою оцінку для певної задачі так, щоб інші учасники її не бачили, після чого оцінки всіх учасників одночасно відкриваються. Якщо оцінки не співпадають, то розпочинається обговорення – фінальна оцінка задачі визначається шляхом досягнення консенсусу між всіма учасниками. Planning Poker є подальшим розвитком методу Wideband Delphi [37], [42] у застосуванні до agile-підходів.

2. Згідно методу *T-shirt Sizing* [52] учасники команди визначають відносну складність або трудоємність задач за розмірами одягу, наприклад: XS, S, M, L, XL, 2XL, 3XL.

3. Метод *Affinity Grouping* [53] передбачає визначення шкали оцінювання (це можуть бути розміри одягу, числа Фібоначчі тощо) і наступну розстановку задач на цій шкалі. Важливо, що оцінка задачі не обов’язково повинна відпові-

дати цілим значенням шкали, а може займати проміжне значення. Особливістю цього методу є те, що учасники команди здійснюють щось на зразок «кластеризації» задач, об'єднуючи їх за схожістю (наприклад, відносною складністю, трудоємністю, ступенем невизначеності). Варто відзначити, що в цій дисертаційній роботі ідея методу Affinity Grouping отримала застосування у визначенні трудоємності для попереднього оцінювання (див. пункт 2.7.5).

4. Метод *Bucket System* [52], [53] передбачає визначення множини «кошиків», що відповідають, наприклад, послідовності чисел Фібоначчі. Тоді вибираються кілька задач, які «розміщуються» у «кошиках» 8, 5 або 3. Всі наступні задачі «розміщуються» у «кошиках» відносно перших задач, іншими словами, відбувається оцінювання за відносною складністю або трудоємністю. Відмінність від Affinity Grouping полягає в тому, що немає можливості «розмістити» задачі у проміжній позиції між «кошиками». Перевагою цього методу є швидкість оцінювання великої кількості задач.

5. Метод *Random Distribution* передбачає розроблення відносно простої шкали і наступне випадкове розміщення задач на цій шкалі. Після цього кожен з учасників команди робить корекцію позиції однієї із задач. Процес продовжується до тих пір, поки всі учасники команди не погодяться із позиціонуванням задач на шкалі.

6. Метод *Dot Voting* [52] передбачає надання кожному з учасників команди обмеженої кількості «голосів». Учасники «віддають» свої голоси за задачі, які вони вважають найбільш складними чи такими, що вимагають найбільше часу на реалізацію. Як наслідок, відносна трудоємність задачі є тим більшою, чим більшу кількість «голосів» вона отримала.

7. Цей метод оцінювання передбачає сортування задач за трьома категоріями: *Large, Small, Uncertain* [52]. З однієї сторони це дає змогу досить швидко провести оцінювання, а з іншої – ідентифікувати задачі, для оцінювання яких недостатньо інформації.

Спільною рисою вище згаданих agile-методів оцінювання є те, що вони розраховані на використання командою на етапі реалізації проєкту. Беззаперечно, їх сильною стороною є забезпечення залученості всієї проєктної команди до оцінювання. Традиційно в якості одиниці вимірювання трудоємності agile-методи оцінювання використовують так звані «сторі-поінти» (англ. «story points») [52], [54], які є похідними від функційних точок [26], [27].

Серед недоліків agile-методів слід відзначити наступні:

1) конвертування «сторі-поінтів» у часові одиниці, як правило, вимагає наявності певних історичних даних, наприклад, кількість «сторі-поінтів», які команда реалізовує за один спринт;

2) складність їх застосування до початку реалізації проєкту, коли проєктна команда ще не сформована;

3) орієнтованість на короткотермінове планування, наприклад, на спринт чи кілька спринтів;

4) точність та надійність, які б давали змогу використовувати оцінку для взяття зобов'язань щодо реалізації проєкту, наприклад, під час укладення договору з клієнтом.

Відзначимо, що ці недоліки частково вирішуються у методі CAEA (англ. «Constructive Agile Estimation Algorithm») [30], [55], який поділяє оцінювання на дві частини: раннє та ітеративне.

Як показує практика, сучасні проєкти з розробки ПЗ поєднують кращі практики з agile-підходів, а також водоспадної та спіральної моделей життєвого циклу. Метод поетапного оцінювання, розроблений в цій дисертаційній роботі, враховує цю особливість сучасних проєктів з розробки ПЗ (див. розділ 2).

1.3.4. Методи оцінювання із застосуванням машинного навчання та штучного інтелекту

Розвитку методів оцінювання з цієї категорії посприяло, починаючи з 2000-х років, поширення застосування методів «дата-саєнс» (англ. «data science»), машинного навчання (англ. «machine learning»), штучного інтелекту,

ШІ (англ. «artificial intelligence», AI) до розв'язання бізнес-задач. Як можна бачити із [56], оцінювання проєктів є одним із найбільш актуальних напрямків застосування машинного навчання та ШІ у галузі розробки ПЗ. Методи оцінювання з цієї категорії можна диференціювати на такі підкатегорії:

1. *Методи оцінювання за аналогією* [57], [58], [59], [60] передбачають отримання оцінки, відштовхуючись від даних схожих проєктів. Такі методи можуть базуватися як на формальних підходах, що використовують метрики[1] схожості між проєктами [61], [62], [63], так і на експертних оцінках. В обох цих випадках необхідна наявність історичних даних аналогічних проєктів. Суттєвим недоліком методів з цієї категорії є недостатність аналізу змісту проєктних робіт, головна перевага – швидкість оцінювання. Тому методи оцінювання за аналогією доцільно застосовувати на найбільш ранніх етапах аналізу та проєктування.

2. *Методи оцінювання, що базуються на застосуванні моделей, «навчених» на наборах даних історичних проєктів*. Упродовж крайніх 10-15 років було опубліковано досить велику кількість наукових праць, що висвітлюють застосування широкого спектру методів машинного навчання (наприклад, дерев рішень, Баєсових мереж, штучних нейронних мереж, регресійного аналізу, генетичних алгоритмів, нечіткої логіки тощо) до існуючих публічно доступних наборів даних історичних проєктів з розробки ПЗ [64], [65]. Приклади результатів таких досліджень, зокрема, представлені у працях [62], [66], [67], [68], [69], [70], [71], [72].

3. Ще одну групу складають *методи оцінювання, покращені за рахунок застосування машинного навчання на ШІ*. До цієї груми належать методи, що базуються як на «класичних» методах першого та другого поколінь, так і на agile-методах. Покращення можуть полягати у зменшенні частки застосування експертних оцінок, калібруванні значень параметрів, підвищенні точності тощо. Приведемо лише декілька прикладів наукових праць з цієї категорії: застосу-

[1] Тут термін «метрика» використовується в розумінні метричних просторів, якими оперує математична наука.

вання нечіткої логіки [73] та Баєсових мереж [74] до методу PERT, автоматизація методу CPM [75], покращення COCOMO шляхом застосування штучних нейронних мереж [76] та генетичних алгоритмів [77], підвищення точності UCP шляхом застосування дерев рішень [78].

4. Передбачається, що наступним поштовхом до розвитку методів оцінювання стане *генеративний ШІ* (англ. «generative artificial intelligence», GenAI або GAI). Від класичного машинного навчання генеративний ШІ відрізняється здатністю генерувати новий контент (текст, зображення, відео, звук, програмний код тощо). Динамічний розвиток відповідних технологій розпочався з 2020-х років, зокрема, революційним став вихід чат ChatGPT 3-ої та 4-ої версій відповідно у 2022 р. та 2023 р., розробленого компанією OpenAI. Не зважаючи на критику сучасного стану технологій генеративного ШІ, домінуючою є точка зору, що ця технологія зробить революцію у галузі розробки ПЗ вже у найближчому десятилітті [79]. Можна передбачити, що на методи оцінювання це матиме двоякий вплив: з однієї сторони очікується розробка методів оцінювання із застосуванням генеративного ШІ, а з іншої передбачається модифікація вже існуючих методів оцінювання з врахуванням того, яких змін під впливом генеративного ШІ зазнає сама розробка ПЗ, зокрема, написання програмного коду.

Спільною рисою методів оцінювання, що базуються на використанні машинного навчання та ШІ є необхідність у історичних даних. Тому одним зі завдань інформаційної технології, розробленої в рамках цієї дисертаційної роботи, є забезпечення накопичення історичних даних про проєкти та їх оцінки. Накопичення достатньої кількості таких даних відкриє можливості для покращення реалізованих методів оцінювання за рахунок застосування машинного навчання та ШІ.

1.4. Огляд здобутків українських науковців та практиків у галузі оцінювання проєктів з розробки програмного забезпечення

Передумовами стрімкого розвитку ІТ-індустрії в Україні, що розпочався з другої половини 90-х років, стали, з однієї сторони, потужний науковий та інженерний потенціали нашої країни [80], а з іншої – кон'юнктура світового ринку, коли собівартість розробки ПЗ в Україні була значно нижчою у порівнянні із країнами Заходу, зокрема, США. Суперпозиція цих факторів дала поштовх утворенню сприятливого середовища для виникнення та швидкого зростання компаній, що спеціалізуються на розробці комерційного ПЗ для закордонних замовників.

Особливістю української ІТ-сфери до середини 2010-х рр. було переважання «аутсорсингових» (англ. «outsourcing») проєктів, коли власне написання вихідного коду здійснювалося українськими розробниками, а бізнес-аналіз та проєктування архітектури проводилися на стороні замовників [80]. Публікації українських науковців цього періоду, здебільшого, присвячені як огляду можливостей, так і подальшому розвитку таких відомих методів оцінювання як COCOMO II та SLIM [81], [82], [83], [84], [85].

На думку автора, з другої половини 2010-х рр. значно посилилася тенденції переходу від «аутсорсингових» послуг до розробки проєктів «під ключ». В цей час в поле зору українських науковців почали потрапляти й інші методи оцінювання.

Цікавим, з практичної точки зору, є метод формування команди експертів з оцінювання [17], що враховує персональні схильності експертів (схильність до ризику відносно часу або ресурсів) з наступною згорткою експертних оцінок із застосуванням 4-ох видів степеневих середніх: середнього гармонійного, середнього геометричного, середнього арифметичного, середнього квадратичного.

Не обійшли уваги українських науковців і методи, що базуються на ідеї функційних точок. Зокрема, у [86] представлена модифікація методу FPA з підвищеною точністю оцінювання, що враховує повторне використання функцій

одного і того ж самого проєкту. Науковий і практичний інтерес становить праця [47], присвячена застосуванню методу COSMIC [87] до оцінювання програмного забезпечення національних реєстрів України, використовуючи наступний підхід: визначити функційний розмір ПЗ, керуючись специфікацією вимог, архітектурою та нормативними актами; тоді отримати трудоємність змісту проєктних робіт, скориставшись базою даних проєктів з розробки ПЗ ISBSG (https://www.isbsg.org/, дата звернення: 06.03.2025); і, знаючи трудоємність, оцінити вартість розробки, відштовхуючись від середньої заробітної плати IT-фахівців на ринку України.

У праці [88] запропоновано модифікацію методу PERT, де замість бета-розподілу використовується розподіл Рейля. У цій статті, зокрема, показано, що оцінка за найбільш ймовірним та мінімальним часом співпадає із класичним методом PERT, в той час як оцінка за максимальним часом суттєво відрізняється. Іншими словами, удосконалена версія методу є більш песимістичною, що, в свою чергу, дає змогу отримати більш реалістичну оцінку для сучасних проєктів, в яких часто трапляються затримки під час реалізації.

У статті [89] запропоновано вирішення актуальної практичної задачі оптимізації планування проєктів, що реалізовуються кількома командами. Запропонована математична модель мінімізує тривалість реалізації проєкту, враховуючи залежності між задачами та базуючись на поєднанні методу критичного шляху (CPM) [25] та методів лінійного програмування.

Тенденція дослідження можливостей застосування методів машинного навчання до оцінювання проєктів з розробки ПЗ знайшла також відображення у працях українських науковців, зокрема, варто відзначити публікації, присвячені розробці методів, які базуються на регресійних моделях [90], [91], [92], [93], деревах рішень та генетичних алгоритмах [71], [72]. При цьому потрібно підкреслити, що у сенсі застосування до оцінювання реальних виробничих проєктів потенціал методів, що базуються на машинному навчанні та ШІ, розкритий ще далеко не повністю і представляє значний як науковий, так і практичний інтерес.

На сьогодні компетентності оцінювання проєктів включені в навчальні програми підготовки ІТ-фахівців. Відповідні теми, зокрема, входять до навчальних курсів, пов’язаних з проєктним менеджментом, командною роботою, технологіями розробки ПЗ. Варто відзначити, що в ряді університетів у програму підготовки для спеціальності 121 «Інженерія програмного забезпечення» включено дисципліну «Економіка програмного забезпечення», одним із основних завдань якої є набуття студентами компетентностей з оцінювання економічних показників проєктів з розробки ПЗ, зокрема, застосовуючи такі методи оцінювання як PERT, CPM, FPA, COCOMO [94], [95].

Упродовж багатьох років роботи на ринку кожна з українських ІТ-компаній напрацювала свої власні підходи до оцінювання проєктів, що, зазвичай, базуються на вже існуючих методах оцінювання, адаптованих до специфіки діяльності конкретної компанії. З досвіду автора, ці підходи, здебільшого, базуються на 3-точковій оцінці PERT та методі критичного шляху (CPM). Широкого поширення серед українських розробників набули agile-методи оцінювання (в основному завдяки своїй простоті та забезпеченню залученості всієї команди). При цьому варто відзначити, що методи із сімейства COCOMO чи такі, що базуються на функційних точках, знаходять своє застосування на практиці значно рідше. Це, головним чином, пов’язано із їх складністю та відсутністю необхідних даних для калібрування значень параметрів та конвертування функційних точок чи кількості логічних ліній коду у людино-години.

Зростаючий рівень складності виконуваних проєктів та збільшення конкурентного тиску на світовому ринку вимагають від українських розробників постійного вдосконалення практичних підходів до оцінювання, що, в свою чергу, актуалізує відповідну наукову задачу із розвитку вже існуючих та розроблення нових методів оцінювання проєктів з розробки ПЗ.

1.5. Оцінювання проєктів з розробки програмного забезпечення як бізнес-процес

1.5.1. Конкретизація поняття «бізнес-процес оцінювання проєктів»

Організація реалізовує свої цілі, виконуючи певні послідовності операцій, які прийнято називати бізнес-процесами. На сьогодні не існує єдиного загально визнаного визначення цього поняття. Щоб забезпечити однозначне розуміння терміну «бізнес-процес» в контексті дисертаційної роботи, зробимо стислий огляд тлумачень цього поняття, які використовуються в таких галузях як економічна наука, бізнес-процес менеджмент (англ. «business process management», BPM) та процес-майнинг (англ. «process mining»).

В економічний науці часто використовують тлумачення, дане М. Хаммер та Дж. Чампі [96], згідно якого бізнес-процес – це сукупність різних активностей (англ. «activities»), що приймає на вхід визначений тип ресурсів, а на виході представляє результат, який має цінність для користувача або замовника. Інше поширене трактування належить Т. Давенпорт [97], в якому бізнес-процес є структурованою вимірюваною послідовністю кроків, що призначена для створення певного результату для клієнта чи ринку. Л.І. Чорнобай та О.І. Дума [98] пропонують розуміти бізнес-процес як «систему безперервних, взаємопов'язаних, відповідним чином упорядкованих і керованих дій (процедур, операцій, виконуваних функцій), яка, в свою чергу, є елементом механізму формування доданої вартості (споживчої цінності) через перетворення організаційних ресурсів, зосереджених на досягненні однієї комплексної цілі, спрямованих на забезпечення продуктивності та ефективності організації в цілому і забезпеченні донесення доданої вартості (споживчої цінності) до цільового ринку через бізнес-модель підприємства».

У специфікації BPMN (англ. «Business Process Modeling and Notation») [99] бізнес-процес визначається як множина бізнес-операцій, які виконуються в межах організації і спрямовані на досягнення її цілей. При цьому бізнес-процес характеризується правилами переходу від однієї операції до іншої і використовує

інформацію та ресурси. В контексті бізнес-процес менеджменту М. Веске [100] дає визначення бізнес-процесу як множини активностей, що скоординовано виконуються в організаційному та технічному середовищі, і в сукупності реалізують бізнес-цілі. М. Веске підкреслює, що бізнес-процес виконується в межах однієї організації і при цьому може взаємодіяти з бізнес-процесами, що функціонують в інших організаціях.

А. Бураттін [101] розглядає бізнес-процес як деяку повторювану стандартну процедуру з наступними характеристиками:

1) процедура складається з множини задач, для яких визначений порядок виконання, але він може в певній мірі змінюватися для різних повторень процесу;

2) кожна із задач реалізовується одним або більше виконавцями, які можуть бути людьми або машинами;

3) виконання окремої задачі створює певний результат, який може використовуватися на наступних кроках процесу.

В задачах процес-майнингу бізнес-процес, як правило, розглядаються з точки зору так званих event-даних (англ. «event data»). Так, В. ван-дер-Аалст [102] робить наступні припущення щодо event-даних:

1) процес складається з кейсів (англ. «case»);

2) кейс складається з подій (англ. «event»), конкретна подія належить лише до одного кейсу;

3) події в межах кейсу є впорядкованими;

4) події можуть мати атрибути.

З точки зору математики, бізнес-процес може розглядатися як орієнтовний граф [103], вершини якого складають множину всіх можливих активностей, що виконуються в рамках бізнес-процесу, а множина дуг визначає можливі переходи від однієї активності до іншої. Слід зауважити, що визначення бізнес-процесу із [103] є зручною формалізацією схеми бізнес-процесу, що може бути використана для його візуалізації.

Підсумовуючи вище сказане, в контексті цієї дисертаційної роботи під терміном «бізнес-процес» розумітимемо вид діяльності організації, що має наступні характеристики:

1) функціонує в межах лише однієї організації;

2) приймає на вхід ресурси та інформацію і забезпечує отримання бажаного результату на виході;

3) складається з множити задач, для яких визначений приблизний порядок виконання, що може в певній мірі змінюватися для різних повторень процесу;

4) виконавцями задач можуть виступати як працівники організації, так і програмні компоненти (наприклад, веб-сервіси);

5) в результаті виконання бізнес-процесу залишається цифровий «відбиток» (англ. «digital footprint») в IT-системах (наприклад, базі даних, текстових файлах тощо).

Отже, *бізнес-процес оцінювання проєктів з розробки ПЗ* (англ. «business process of software development projects estimation») – це бізнес-процес, який:

1) функціонує в межах організації, що займається виконанням проєктів з розробки ПЗ (така організація може розробляти ПЗ як на замовлення, так і для власних потреб);

2) приймає на вхід вимоги до проєкту та забезпечує отримання оцінки проєкту з розробки ПЗ (у розумінні, сформульованому у підрозділі 1.1);

3) визначає перелік задач, які необхідно виконати, залишаючи про цьому певний ступінь гнучкості, який дає змогу експертам, залученим до оцінювання, застосовувати творчі підходи та опиратися на свої унікальні знання та досвід;

4) координує взаємодію експертів та програмних компонент (наприклад, інструментів підтримки прийняття рішень, баз даних історичних проєктів тощо);

5) в результаті виконання цього бізнес-процесу залишається цифровий «відбиток» у IT-системах організації як про сам його перебіг, так і про результати оцінювання (наприклад, у Excel-документах, презентаціях, базах даних тощо).

З метою спрощення поряд із повним формулюванням терміну також будемо використовувати його скорочений варіант – «бізнес-процес оцінювання» (англ. «estimation business process»).

1.5.2. Оцінювання як слабоструктурований бізнес-процес

Як було зазначено у попередньому пункті, однією із характеристик бізнес-процесу оцінювання те, що, з однієї сторони, він визначає певну послідовність кроків, а, з іншої, залишається досить гнучким, забезпечуючи для експертів «простір» для застосування нестандартних творчих підходів, відштовхуючись від їх знань, досвіду та інтуїції. Часто такі бізнес-процеси називають слабоструктурованими (англ. «semi-structured»).

На сьогодні в літературі немає єдиного визначення поняття слабоструктурованого бізнес-процесу. В контексті дисертаційної роботи ми розглядатимемо слабоструктуровані бізнес-процеси як проміжну ланку між структурованими та неструктурованими. Для визначення характеристик, за якими диференціюють бізнес-процеси цих трьох категорій, розглянемо існуючі підходи до класифікації бізнес-процесів за рівнем їх структурованості.

В. ван-дер-Аалст [102] розрізняє дві основні категорії бізнес-процесів: структуровані (в термінології автора «лазанья»-процеси) та неструктуровані (так звані «спагеті»-процеси). З точки зору В. ван-дер-Аалста, такий поділ, перш за все, зумовлений можливістю застосування тих чи інших методів процес-майнингу. У структурованих бізнес-процесах всі кроки є повторюваними і мають чіткі вхідні та вихідні параметри. Такі процеси дозволяють автоматизацію, оскільки вони практично не залежать від рішень, що базуються на знаннях, досвіді чи інтуїції виконавців. Згідно з критерієм, представленим у [102], бізнес-процес вважається структурованим, якщо для нього можна побудувати схему, погоджену з учасниками процесу, згідно з якою більш як 80% подій виникають, як заплановано. Неструктуровані бізнес-процеси (або «спагеті»-процеси) знаходяться на протилежному кінці спектру. Їх визначною відмінністю є висока залежність від залучених у їх виконання учасників. Іншою характеристикою таких

процесів є велика кількість кроків із складною системою переходів та розгалужень. В свою чергу, слабоструктуровані бізнес-процеси займають проміжне місце між «лазанья»- та «спагеті»-процесами.

П. Хармон [104] виділяє такі три категорії бізнес-процесів, опираючись на ступінь фіксованості їх кроків та рівень компетентності учасників: прості, більш складні та дуже складні. Бізнес-процеси з першої категорії характеризуються чітко визначеною послідовністю кроків, відносно невеликою кількістю правил та точок розгалуження, а від їх учасників не вимагається високий рівень компетентності чи знань предметної області. Дуже складні процеси розглядаються як повна протилежність процесам із першої категорії: послідовність їх кроків може суттєво змінюватися, рішення приймаються, опираючись на досвід, інтуїцію та глибокі експертні знання їх учасників. Процеси 2-ої категорії розглядаються як перехідна ланка між 1-ою та 3-ою категоріями. Їх послідовність кроків визначена (але не є строго фіксованою), при цьому закладається можливість виникнення відхилень та нестандартних ситуацій. Учасники таких бізнес-процесів опираються на свої знання предметної області, хоча не вимагається, щоб їх компетентність була найвищого рівня.

Досить схожа класифікація бізнес-процесів пропонується С. Кемслі [105], який виділяє наступні чотири категорії бізнес-процесів: структуровані, структуровані з виключеннями, неструктуровані із структурованими фрагментами та неструктуровані. Згідно з С. Кемслі [105] на рівень структурованості бізнес-процесу, зокрема, випливають рівень знань та спосіб взаємодії їх учасників. Таким чином, структуровані бізнес-процеси є рутинними та не потребують високого рівня знань та досвіду; взаємодія між учасниками таких бізнес-процесів є, як правило, ієрархічною. В свою чергу, неструктуровані бізнес-процеси вимагають креативності та високого рівня компетентності. Взаємодія між учасниками в цих бізнес-процесах має характер співпраці і не вимагає (або навіть не дозволяє) жорсткого ієрархічного контролю.

Як видно із підходів до класифікації бізнес-процесів за критерієм структурованості, чим менш структурований процес, тим більше його перебіг залежить від рівня компетентності (знань, навиків, досвіду, інтуїції) його виконавців. Тому неструктуровані, а також слабоструктуровані бізнес-процеси, часто відносять до так званих «knowledge-intensive» (англ.) бізнес-процесів [106], [107], [108]. Характерною рисою таких бізнес-процесів є висока ймовірність зміни послідовності кроків під час виконання, що залежить від рішень його учасників. Співпраця між учасниками визначається як ключова складова перебігу такого бізнес-процесу.

Приклади структурованих бізнес-процесів можна знайти у виробництві, фінансах, логістиці. Одним із факторів, що впливають на рівень їх структурованості є частота повторюваності – чим частіше повторюється бізнес-процес, тим краще він структурований. Неструктуровані бізнес-процеси зустрічаються, наприклад, в розробці нових продуктів. Це, зокрема, можна пояснити тим, що розробка нового (а тим більше унікального) продукту вимагає високого рівня креативності та інтенсивної командної взаємодії його учасників. Часто неструктурованими є бізнес-процеси, пов’язані з маркетингом, оскільки з метою позиціонування на конкурентному ринку доводиться вдаватися до нестандартних чи інноваційних підходів. Бізнес-процеси в таких сферах як продажі високотехнологічних рішень, розробка ПЗ, як правило, є слабоструктурованими. Однією із причин, яка суттєво ускладнює трансформацію цих бізнес-процесів в структуровані, є необхідність опиратися на знання, досвід та навички їх учасників. Порівняння бізнес-процесів за ступенем їх структурованості представлено у додатку 2.

Бізнес-процес оцінювання проєктів з розробки ПЗ віднесемо до категорії слабоструктурованих з наступних причин:

1) послідовність його основних кроків є визначеною, але при цьому не є чітко фіксованою, що забезпечує певний ступінь гнучкості;

2) від учасників бізнес-процесу вимагається досить високий рівень компетентності та досвіду;

3) певні ланки бізнес-процесу (наприклад, оцінювання трудоємності) повністю опираються на експертну оцінку учасників;

4) стиль взаємодії між учасниками будується на основі співпраці (а не ієрархічності);

5) бізнес-процес оцінювання постійно еволюціонує, опираючись на раніше отриманий досвід.

1.5.3. Застосування процес-майнингу до побудови схем слабоструктурованих бізнес-процесів

Однією із основних дисциплін, в рамках якої здійснюється вивчення та автоматизація слабоструктурованих бізнес-процесів є процес-майнинг (англ. «process mining»). Процес-майнинг виступає своєрідним «мостом», що поєднує кілька галузей такі як: бізнес-процес менеджмент (англ. «business process management», BPM), машинне навчання (англ. «machine learning»), інтелектуальний аналіз даних (англ. «business intelligence», BI). В той час як методи бізнес-процес менеджменту показують хорошу ефективність для структурованих бізнес-процесів, методи процес-майнингу застосовні до слабоструктурованих чи навіть неструктурованих бізнес-процесів.

Процес-майнинг як прикладна дисципліна виник у 90-х роках. Значний вклад у його розвиток було зроблено у Технічному університеті Ейндговена (Нідерланди) під керівництвом професора В. ван-дер-Аалста. Процес-майнинг маніфест [109] визначає наступні три основні категорії задачі: процес-діскавері (англ. «process discovery»), перевірка відповідності (англ. «conformance checking») та покращення (англ. «enhancement»). Метою задач з першої категорії є побудова так званих «реальних» схем бізнес-процесів, що базуються на даних, зібраних під час виконання цих бізнес-процесів. Задачі другої категорії фокусуються на виявленні відмінностей між «реальними» (згенерованими на основі даних) та «ідеальними» (розроблених аналітиками) схемами бізнес-процесів. В

свою чергу, задачі з третьої категорії мають за мету надання рекомендацій щодо покращення бізнес-процесів. Більше інформації про задачі процес-майнингу та його прикладне застосування можна знайти у [1], [102], [110], [111], [112].

Для слабоструктурованого бізнес-процесу оцінювання проєктів з розробки ПЗ, перш за все, є актуальною перша задача – процес-діскавері, що передбачає автоматичну побудову схеми цього бізнес-процесу із цифрового відбитку, залишеного в результаті виконання цього бізнес-процесу.

З метою конкретизації поняття «схема бізнес-процесу»[1] скористаємося визначенням із [113]: *схема бізнес-процесу* (англ. «business process model») складається із множини кроків (або активностей) бізнес-процесу та правил переходів між ними; *екземпляр бізнес-процесу* (англ. «business process instance») є множиною виконаних кроків бізнес-процесу; тоді схема бізнес-процесу виступає узагальненням для множини екземплярів бізнес-процесу. Прикладом схеми бізнес-процесу може слугувати BPMN-діаграма, UML-діаграма, мережа Петрі тощо.

Одним із найбільш ранніх методів процес-діскавері був так званий α-алгоритм [102], [114], розроблений на початку 2000-х років. Цей метод забезпечував побудову схеми бізнес-процесу у вигляді мережі Петрі. Не зважаючи на те, що α-алгоритм показав досить хороші результати при побудові схем структурованих бізнес-процесів, його застосування до слабоструктурованих чи неструктурованих бізнес-процесів призводило до генерації схем, які є досить важкими для сприйняття, в першу чергу через те, що вони відображають навіть найдрібніші деталі перебігу бізнес-процесу. Іншим недоліком α-алгоритму, що суттєво обмежує його практичне застосування, було використання мереж Петрі (перш за все, через складність їх розуміння користувачами, які не є експертами у процес-майнингу).

Heuristic Miner [115], [116] є іншим відомим методом процес-майнингу. Цей метод усовує деякі недоліки α-алгоритму, зокрема, відфільтровує так званий

[1] Відзначимо, що аналогом терміну «схема бізнес-процесу» в англомовній науковій літературі є «business process model». При перекладі англомовного варіанту на українську мову слово «model» було замінено на «схема», щоб уникнути неоднозначності, пов'язаної із науковим трактуванням поняття «модель».

«шум» в event-даних, а також виключає із схеми бізнес-процесу кроки та переходи між ними, які найменш часто зустрічаються. Для візуалізації схеми бізнес-процесу Heuristic Miner використовує казуальні мережі (англ. «casual nets»), які є простішими для розуміння та сприйняття у порівнянні з мережами Петрі. Ці переваги Heuristic Miner дають змогу отримувати схеми слабоструктурованого бізнес-процесу, які є значно інформативнішими та легшими для сприйняття користувачами у порівняння із схемами, отриманими з допомогою α-алгоритму.

Недоліки α-алгоритму, що проявляються при побудові схем слабоструктурованих та неструктурованих бізнес-процесів, вирішуються також методом Fuzzy Miner [117], [118]. Ідея цього методу полягає в тому, що більш «важливі» аспекти бізнес-процесу відображаються на схемі, а менш важливі приховуються. Такий ефект досягається застосуванням набору характеристик[1], що використовуються для визначення «важливості» кроків бізнес-процесу та переходів між ними. В результаті рівень деталізації схеми бізнес-процесу можна корегувати, змінюючи значення його параметрів. В цьому сенсі схема бізнес-процесу, побудована із застосуванням Fuzzy Miner, нагадує географічну мапу, рівень деталізації якої зменшується при віддаленні і збільшується при наближенні. Така гнучкість у візуалізації схем бізнес-процесів є перевагою Fuzzy Miner перед Heuristic Miner. Про ефективність методу Fuzzy Miner у застосуванні до практичних задач свідчить і той факт, що цей метод був адаптований такими відомими програмними продуктами процес-майнингу як Disko (https://fluxicon.com/disco/, дата звернення: 22.05.2024) [119] та Celonis (https://www.celonis.com/, дата звернення: 22.05.2024) [120]. При цьому варто підкреслити, що оригінальна версія Fuzzy Miner [117], [118] не призначена для опрацювання потоків даних, що може спричинити певні обмеження в разі застосування цього методу до наборів event-даних великого обсягу, до яких постійно надходять нові дані.

[1] В англомовній літературі у зв'язку із Fuzzy Miner вживається термін «metric», що перекладається як «метрика». Однак, з метою уникнення колізії з трактуванням терміну «метрика», що визначається у математичній науці як «відстань» між елементами метричних просторів, у цій дисертаційні роботі у контексті Fuzzy Miner замість «метрика» використовується «характеристика».

Отже, з існуючих методів процес-майнинг Fuzzy Miner найкраще підходить для побудови схеми бізнес-процесу оцінювання. Такий вибір методу зумовлений гнучкістю, яку забезпечує Fuzzy Miner, при візуалізації схем слабоструктурованих бізнес-процесів. При цьому варто зазначити, що оригінальна версія Fuzzy Miner потребує модифікацій, зокрема, забезпечення підтримки потоків даних (див. підрозділи 3.3 та 3.4).

1.6. Постановка наукових та технічних задач дослідження

За результатами аналізу існуючих методів оцінювання та, враховуючи слабоструктуровану природу бізнес-процесу оцінювання, а також, відштовхуючись від практичного виробничого досвіду автора в галузі розробки ПЗ, ідентифіковано наступні проблеми:

1. *Невідповідність існуючих методів оцінювання сучасним підходам до реалізації проєктів з розробки ПЗ.* Значна частина методів оцінювання була розроблена упродовж другої половини XX-ого століття (це стосується, в першу чергу, «класичних» методів) і, як наслідок, вже не повністю відповідає сучасним вимогам. Зокрема, така невідповідність проявляється в тому, що переважна більшість проєктів зараз реалізується із застосуванням agile-підходів, що не враховується цими методами.

2. *Рівень складності існуючих методів оцінювання.* Для прикладу, головною пересторогою до практичного застосування таких відомих методів як COCOMO II, FPA, COSMIC є їх складність, що вимагає спеціальної підготовки експертів, які їх використовують.

3. *Використання SLOC-метрики для визначення «розміру» ПЗ.* Використання SLOC-метрики розпочалося ще у «класичних» методах оцінювання (COCOMO, SLIM, методі функційних точок) більш як 50 років тому. З того часу сфера розробки ПЗ неодноразово зазнавала значних змін, і вимірювання розміру ПЗ з використанням SLOC-метрики багатократно піддавалося критиці. Тому

практичне застосування методу оцінювання, що передбачає використання SLOC-метрики, викликає скептицизм у фахівців з оцінювання та розробки ПЗ.

4. *Недоступність історичних даних.* Ряд «класичних» методів, зокрема: COCOMO та COCOMO II, FPA, COSMIC – вимагають наявності історичних даних для калібрування значень їх параметрів. Недоступність даних історичних проєктів також є пересторогою до використання методів, що базуються на застосуванні машинного навчання та ШІ. З однієї сторони, існує ряд джерел даних як комерційних (наприклад, SLIM), так і з відкритим доступом (наприклад, ISBSG). Однак, залишаються невирішеними питання щодо якості цих даних та їх відповідності реаліям конкретної організації. З іншої сторони, для організації, що займається розробкою ПЗ на постійній основі, важливо мати власну базу даних виконаних проєктів, дані з якої використовуватимуться для оцінювання наступних проєктів (часто, така база даних в організації відсутня тільки тому, що оцінювання здійснюється у Excel-документах).

5. *Послідовне охоплення етапів життєвого циклу розробки ПЗ.* Переважна більшість методів оцінювання (виняток становить COCOMO II) застосовна лише на певному етапі життєвого циклу проєкту. Яскравим прикладом цього є agile-методи оцінювання, які розроблені для застосування на етапі реалізації проєктною командою; такі методи, як правило, не передбачають можливості застосування до початку етапу реалізації, коли проєктна команда ще не сформована. Іншим аспектом цієї проблеми є передача оцінки від етапів аналізу та проєктування до етапу реалізації проєкту: несумісність методів оцінювання, які використовуються на цих етапах життєвого циклу, призводить до складнощів з порівнянням оцінки, наданої експертами до початку реалізації, з оцінкою від проєктної команди, зробленої вже на етапі реалізації.

6. *«Одноразова» оцінка*, яка готується лише один раз на завершальних етапах аналізу та проєктування, безпосередньо перед початком реалізації. На перший погляд, може здатися, що такий підхід дає змогу зекономити час та ресурси, що затрачаються на оцінювання. Але, як показує практика, застосування

«одноразових» оцінок, особливо у випадку великих проєктів, має ряд негативних наслідків:

а) недостатність інформації щодо оцінки на ранніх етапах аналізу та проєктування;

б) низька якість та точність оцінки (що, значною мірою, зумовлено часовими обмеженнями на її підготовку);

в) високий рівень стресу для команди експертів, що займаються підготовкою оцінки (особливо, це стосується аналізу змісту проєктних робіт та оцінювання його трудоємності).

7. *Недостатність зворотного зв'язку* від проєктної команди, яка здійснює реалізацію проєкту, щодо точності та якості оцінки, яка була підготована до початку реалізації командою експертів. Такий зворотний зв'язок, зокрема, є корисним для експертів, що займатимуться підготовкою оцінок майбутніх проєктів.

8. *Недоліки програмного забезпечення*, що використовується для автоматизації оцінювання та підготовки окремих складових оцінки. Найбільш поширеним інструментом для підготовки оцінок є Microsoft Office 365, зокрема, Excel, суттєвим недоліком якого є неможливість реалізації складних математичних методів, зокрема, методів математичного програмування та підтримки прийняття рішень. Застосування Excel також суттєво ускладнює формування бази даних історичних проєктів. Недоліком такого відомого програмного продукту як Microsoft Project є, зокрема, складність його пристосування до планування проєктів, що реалізуються за agile-підходами.

9. *Підхід до оцінювання як до структурованого бізнес-процесу* призводить, як правило, до того, що побудовані схеми цього бізнес-процесу не враховують багатьох його практичних аспектів, і тому його реальний перебіг може суттєво відрізнятися від таких схем.

Таким чином, для усунення виявлених проблем пропонується розробити методи оцінювання проєктів з розробки ПЗ та відповідну інформаційну технологію для забезпечення практичного застосування розроблених методів.

Для розв’язання поставленої наукової задачі необхідно виконати такі завдання:

1. Конкретизувати поняття оцінки проєкту з розробки ПЗ та визначити складові цієї оцінки. Проаналізувати існуючі методи оцінювання проектів з розробки ПЗ та ідентифікувати основні проблеми, а також актуальні напрямки наукових досліджень в цій галузі. Обґрунтувати, що бізнес-процес оцінювання належить до категорії слабоструктурованих бізнес-процесів. Провести огляд методів процес-майнингу, що призначені для побудови схем слабоструктурованих бізнес-процесів. (Варто підкреслити, що це завдання виконано в розділі 1.)

2. Розробити метод поетапного оцінювання проєктів з розробки ПЗ, застосовний на етапах аналізу та проєктування, який забезпечує поступове підвищення рівня деталізації та точності оцінки, а також враховує agile-природу підходів до розробки на етапі реалізації проєкту.

3. Розробити метод побудови розкладу реалізації задач проєкту, відштовхуючись від оцінки трудоємності цих задач та спроможності розробки проєктної команди.

4. Розробити метод підтримки прийняття рішень, метою якого є надання переліку альтернатив складу проєктної команди та графіку реалізації проєкту, за яким експерти отримують змогу обрати найбільш вдалі, на їх думку, варіанти.

5. Розробити метод побудови схеми слабоструктурованого бізнес-процесу оцінювання, що базується на застосуванні методів процес-майнингу.

6. Провести аналіз вимог та спроєктувати архітектуру інформаційної технології оцінювання проектів з розробки ПЗ, що забезпечує практичне застосування розроблених методів оцінювання, враховує слабку структурованість цього бізнес-процесу та реалізує базу даних проєктів та їх оцінок. Продемонструвати можливості розроблених методів, застосувавши їх до оцінювання ПЗ цієї інформаційної технології.

Варто зауважити, що наступні задачі не входять до завдань дисертаційної роботи:

1) розробка методів оцінювання для етапу реалізації проєкту;

2) розробка методів розрахунку бюджету проєкту;

3) розробка методів опрацювання проєктних ризиків;

4) розробка методів визначення точності, надійності та якості оцінки.

Висновки до розділу 1

Конкретизовано зміст поняття «оцінювання проєктів з розробки ПЗ» та визначено складові такої оцінки. Проведено аналіз існуючих методів оцінювання та запропоновано їх класифікацію за етапами еволюції, починаючи із найбільш ранніх методів, розроблених у 50-х роках минулого століття, та завершуючи найновішими методами, що базуються на застосуванні машинного навчання. Зроблено висновок, що існуючі методи оцінювання не задовольняють сучасним вимогам, зокрема, немає методів, які б забезпечували послідовне охоплення всіх етапів життєвого циклу проєкту від аналізу та проєктування до завершення реалізації, і водночас враховували agile-природу підходів до реалізації проєктів.

Конкретизовано зміст поняття «бізнес-процес оцінювання проєктів з розробки ПЗ», опираючись, зокрема, на визначення, що використовуються в економічний науці, бізнес-процес менеджменті та процес-майнингу. Обґрунтовано, що бізнес-процес оцінювання належить до категорії слабоструктурованих бізнес-процесів, що, зокрема, впливає на спектр методів та засобів, застосовних до роботи з ним. Проведено огляд методів процес-майнингу, що призначені для побудови схеми слабоструктурованих бізнес-процесів. Прийнято рішення здійснити подальший розвиток існуючого методу Fuzzy Miner для побудови схеми бізнес-процесу оцінювання.

За результатами проведеного аналізу ідентифіковано проблеми, що потребують вирішення, сформульовано мету та зроблено постановку завдань дисертаційної роботи; окремо підкреслено, що не входить до завдань цієї дисертаційної роботи.

РОЗДІЛ 2. РОЗРОБЛЕННЯ МЕТОДІВ ОЦІНЮВАННЯ ПРОЄКТІВ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Оцінювання в контексті життєвого циклу проєкту з розробки програмного забезпечення

З точки зору бізнесу, прогнозованість є одним із ключових аспектів розробки ПЗ. Іншими словами, на кожному з етапів життєвого циклу важливо розуміти обсяг ресурсів та часу, необхідних для реалізації проєкту. Будучи невід'ємною складовою життєвого циклу розробки ПЗ, оцінювання на кожному з його етапів має певні цілі, потребує спеціалізованих методів та забезпечує отримання різних результатів.

Ідея покрокового оцінювання в залежності від цілей та етапу життєвого циклу проєкту не є новою. Зокрема, в публікації [121] висловлюється ідея диференціації оцінок на чотири типи, основною відмінністю між якими є точність[1]. Так, перший (і найменш точний) із цих типів передбачає варіацію значень оцінки в межах від -50% до +100%, а найбільш точний, четвертий, допускає варіацію в межах ±5%. Іншим відомим виразом такої ідеї є концепція конусу невизначеності (англ. «the cone of uncertainty»), запропонована С. Маконнеллом [122], [123]. Згідно із конусом невизначеності рівень невідомого є найвищим на початку проєкту і поступово знижується з проходженням етапів його життєвого циклу. Відповідно й оцінка (яка є нічим іншим як прогнозом ресурсів та часу, необхідних для реалізації проєкту, див. підрозділ 1.1) також поступово стає більш точною із зменшенням рівня невідомого. Керуючись схожим принципом, PMBOK 6th [124] розрізняє три типи оцінок: наближена оцінка (англ. «rough order of magnitude») з варіацією значень від -25% до +75%, оцінка бюджету (англ. «budget estimate») з точністю від -10% до +25% та так звана «остаточна» оцінка

[1] Під точністю (англ. «accuracy») оцінки слід розуміти її близькість до відповідного актуального значення, отриманого після завершення реалізації. При цьому на етапі оцінювання ступінь точності оцінки виражається у вигляді інтервалу – чим більше впевненості у точності оцінки, тим меншою є різниця між максимальним та мінімальним значеннями цього інтервалу. Див. підрозділ 1.2.

(англ. «definitive estimate») з варіацією від -5% до +10%. Прикладом того, як залежно від етапу життєвого циклу проєкту застосовують різні методи оцінювання, є COCOMO II [28].

Розроблений в дисертаційні роботі метод поетапного оцінювання передбачає, що кожен з етапів оцінювання відповідає етапу життєвого циклу проєкту з розробки ПЗ (рис. 2.1). Етапи, визначені цим методом, називатимемо життєвим циклом оцінювання (англ. «Estimation Life Cycle», ELC).

Так, *початкове оцінювання* (англ. «introductory estimation») застосовується під час первинного знайомства з ідеєю проєкту. Метою початкового оцінювання є отримання приблизного розуміння «порядку» обсягу ресурсів (трудоємності, розміру команди) та часу (тобто тривалості), необхідних для реалізації проєкту. Зазвичай, на цьому етапі швидкість отримання оцінки є важливішою за її точність. За своєю суттю початкове оцінювання близьке до так званого «оцінювання-вгадування» (англ. «guesstimation») [125], коли оцінка надається за умов недостатності інформації. Для раціонального підкріплення такої оцінки часто застосовують метод оцінювання за аналогією із вже реалізованими проєктами [57], [58], [59], [63]. Враховуючи високий рівень невизначеності (зокрема, відсутність архітектури рішення та аналізу змісту проєктних робіт), на цьому етапі варто оцінювати кілька альтернативних сценаріїв реалізації проєкту (наприклад, із застосуванням різних технологій програмування).

По мірі поглиблення розуміння ідеї проєкту наступним кроком є *попереднє оцінювання* (англ. «preliminary estimation») [5]. Як і у випадку початкового оцінювання, цей етап не передбачає надання оцінки з високою точністю. Однак, оцінювання відбувається за наявності змісту проєктних робіт, представленого у вигляді ієрархічної структури елементів оцінювання (див. підрозділ 2.2), а також початкового розуміння архітектури рішення. Попереднє оцінювання також передбачає опрацювання альтернативних сценаріїв, однак кількість таких сценаріїв доцільно зменшити у порівнянні з попереднім етапом.

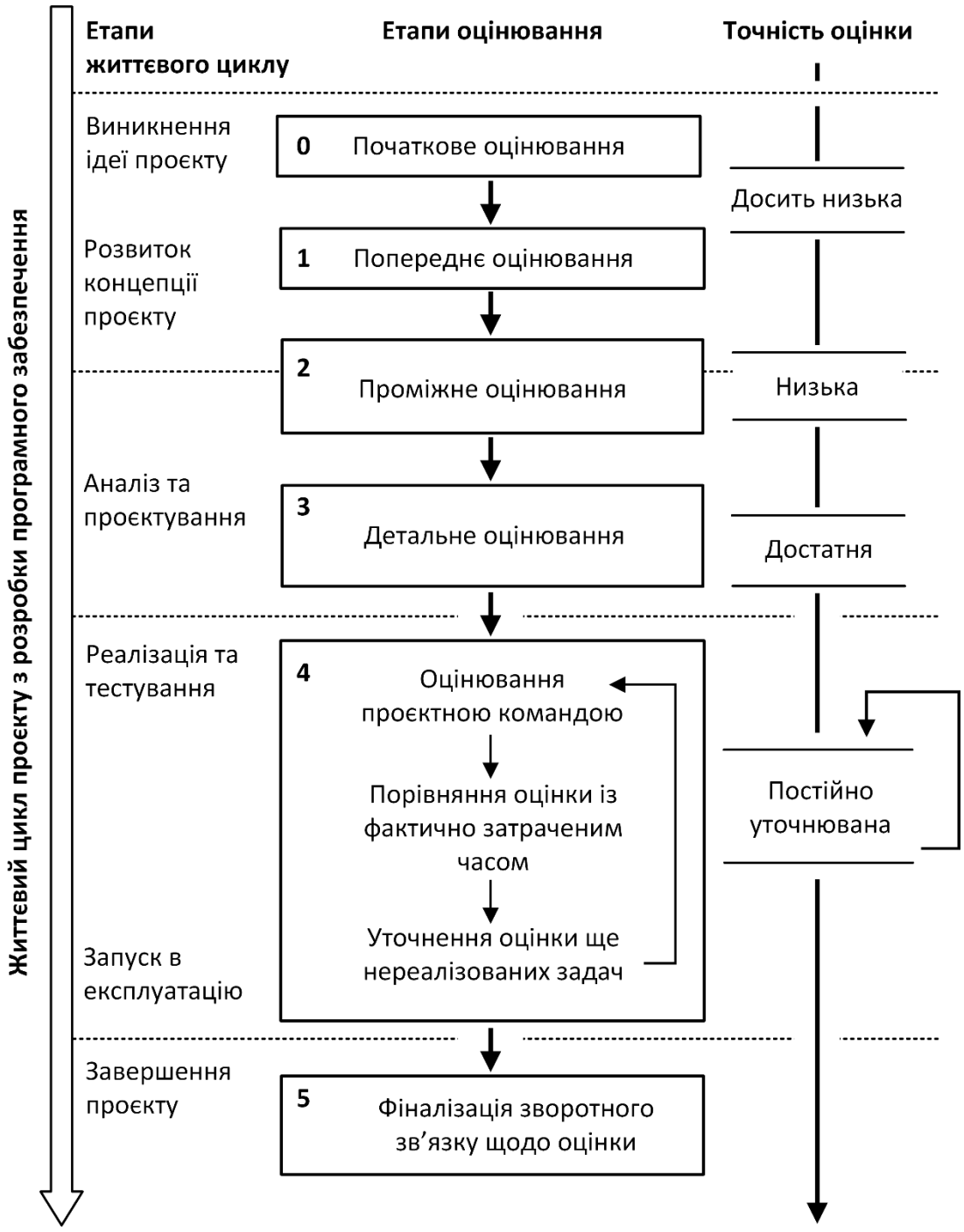


Рис. 2.1. Етапи оцінювання в контексті життєвого циклу проєкту з розробки програмного забезпечення

Проміжне оцінювання (англ. «intermediate estimation») [4], [16] застосовується на початку аналізу та проєктування. Основною метою цього типу оцінювання є надання оцінки точнішої за ту, яка отримана на попередніх етапах. Під час проміжного оцінювання варто сфокусуватися на одному сценарії реалізації проєкту, відкинувши альтернативні сценарії, які аналізувалися перед цим. Вища

точність досягається, зокрема, за рахунок поглиблення розуміння змісту проєктних робіт, деталізації архітектури рішення, а також уточнення складу проєктної команди. Спільною рисою початкового, попереднього та проміжного оцінювань є те, що їх результати не характеризуються достатньою точністю, а отже, не рекомендовані до використання для взяття зобов'язань щодо реалізації проєкту (наприклад, для підписання контракту з клієнтом).

Головним завданням наступного етапу, *детального оцінювання* (англ. «precise estimation»), є отримання оцінки, точність якої дає змогу використати її за основу для взяття зобов'язань з реалізації проєкту. Цей етап оцінювання супроводжує завершення фази аналізу та проєктування. Достатня точність оцінки забезпечується деталізацією змісту проєктних робіт, побудовою розкладу реалізації проєктних задач (наприклад, у вигляді діаграми Ганта чи мережної діаграми), а також аналізом проєктних ризиків і залежностей.

Як можна бачити з рис. 2.1, оцінювання не завершується із початком реалізації проєкту. Безпосередньо на початку етапу реалізації, коли вже сформовано основну частину проєктної команди, команді необхідно провести деталізацію змісту проєктних робіт до рівня задач реалізації (у agile-підходах це так звані «user stories») та провести власне оцінювання цих задач з наступним порівнянням з оцінкою, наданою експертами до початку реалізації.

Уточнення оцінки ще не реалізованих задач, а також порівняння фактично затраченого часу з відповідними оцінками рекомендується проводити регулярно під час реалізації проєкту. Одним із завершальних кроків є фіналізація збору зворотного зв'язку щодо оцінки. Такий зворотний зв'язок є цінним для експертів, що оцінюватимуть майбутні проєкти.

Запропоновані етапи оцінювання є послідовними: результати попереднього етапу слугують вхідними даними для наступного. Проведення такого поетапного оцінювання, зокрема, дає змогу отримати оцінку вищої якості у порівнянні з випадком, коли оцінювання виконується як одноразова задача безпосередньо перед початком реалізації проєкту. Такий підхід поступового підви-

щення точності оцінки, зокрема, дає змогу знизити рівень стресу для експертів, що здійснюють оцінювання.

Варто підкреслити, що, так як оцінювання на перших двох етапах: початковому та попередньому – відбувається за умов високого ступеня невизначеності при відсутності детальної інформації про проєкт, то доцільним є застосування так званого підходу «згори – вниз» (англ. «top-down estimation»), згідно з яким опрацьовують «високорівневі» проєктні задачі без їх деталізації. По мірі поглиблення розуміння вимог до проєкту на етапі проміжного оцінювання стає можливим поєднати підхід «згори – вниз» із діаметрально протилежним підходом – оцінюванням «знизу – вгору» (англ. «bottom-up estimation»). Починаючи з етапу детального оцінювання та до завершення реалізації проєкту, домінуючим є підхід «знизу – вгору», коли кінцевий результат оцінювання формується шляхом акумулювання оцінок дрібних задач.

Розроблений метод поетапного оцінювання застосовний, в першу чергу, до проєктів із фіксованим бюджетом, коли зобов'язання із реалізації проєкту (зафіксувавши вартість та терміни) необхідно взяти до початку розробки. Як правило, такі проєкти виконуються за водоспадною або спіральною моделями життєвого циклу у поєднанні з agile-підходами на етапі реалізації.

Варто відзначити, що проходження чотирьох заявлених етапів оцінювання, що передують початку реалізації проєкту, має сенс для проєктів, «розмір» яких перевищує певне значення (в якості «розміру» проєкту можна, для прикладу, взяти оцінку трудоємності змісту робіт, див. підрозділ 2.4). Відповідно для оцінювання проєктів меншого «розміру» кількість таких етапів також може скоротитися, наприклад, залишається лише проміжне та детальне оцінювання. Корекція етапів оцінювання залежно від «розміру» чи інших особливостей проєкту не входить до завдань цієї дисертаційної роботи.

Порівняльний аналіз етапів оцінювання представлено у додатку 3. Далі у цьому розділі представлено методи, застосовні на етапах попереднього, проміж-

ного та детального оцінювань. Розроблення методів для інших етапів не входить до завдань цієї дисертаційної роботи.

2.2. Ієрархічна структура елементів оцінювання

Однією з умов отримання якісної оцінки проєкту з розробки ПЗ є декомпозиція змісту проєктних робіт. Як правило, результатом декомпозиції є так звана ієрархічна структура робіт (англ. «work breakdown structure», WBS) [126] або її підтипи, наприклад, компонентна ієрархічна структура робіт (англ. «component-based work breakdown structure», CBWBS) [127]. На практиці, залежно від обраного підходу до організації процесу розробки, елементи ієрархічної структури робіт можуть мати різні назви, наприклад: «епіки» (англ. «epics»), «фічі» (англ. «features»), прецеденти (англ. «use cases»), історії користувача (англ. «user stories»), пакет робіт (англ. «work package») тощо. Тому з метою уніфікації термінології автором запропоновано такі поняття як «елемент оцінювання» та «ієрархічна структура елементів оцінювання».

Елементом оцінювання (англ. «estimable item») x називатимемо частину змісту проєктних робіт, для якої може бути оцінена трудоємність (див. підрозділ 2.4), здійснена декомпозиція на дочірні елементи, проведено аналіз припущень, залежностей та ризиків. Елемент оцінювання x володіє набором атрибутів; з точки зору синтаксису, значення атрибуту може бути отримано за допомогою квадратних дужок – $x[\text{attribute name}]$ (тут і нижче по тексту назви атрибутів вказуються англійською мовою). Наприклад, $x[\text{parent}]$ – батьківський елемент для x, $x[\text{risks}]$ – ризики, пов'язані з елементом оцінювання x. У додатку 5 представлено перелік атрибутів елементів оцінювання, які, з точки зору автора, представляють найбільш важливі аспекти аналізу змісту робіт проєкту.

Елемент оцінювання x називатимемо *простим* (англ. «leaf estimable item»), якщо він не має дочірніх елементів, тобто $x[\text{children}] = \varnothing$. В свою чергу, якщо

елемент оцінювання має дочірні елементи, тобто $x[\text{children}] \neq \varnothing$, то його називатимемо *композитним* (англ. «composite estimable item»).

Ієрархічною структурою елементів оцінювання, ІСЕО (англ. «estimable items breakdown structure», EIBS) X називатимемо дерево (згідно визначення з теорії графів) з вершинами, які є елементами оцінювання, ребрами, що пов'язують батьківські та дочірні елементи оцінювання, та кореневим елементом, який представляє зміст робіт проєкту в цілому.

Побудова ІСЕО, як правило, є досить трудомісткою задачею. Це, зазвичай, зумовлено високим рівнем невизначеності та досить жорсткими часовими обмеженнями, коли необхідно проаналізувати великі обсяги інформації за стислий проміжок часу. Важливим аспектом побудови ІСЕО є розуміння як потреби клієнта, так і глибинні знання відповідної предметної області. В разі порушення цих умов виникають високі ризики упущення значної частини змісту проєктних робіт або «роздування» їх обсягу шляхом включення низькопріоритетних та другорядних задач. Це, в свою чергу, призводить до суттєвих відхилень в оцінках часу та ресурсів, необхідних для реалізації проєкту. Іншою типовою помилкою є «погана» декомпозиція змісту робіт, що створює суттєві труднощі для наступного аналізу та деталізації відповідної ІСЕО. Ще однією проблемою є ігнорування того факту, що зміст проєктних робіт не є «статичним» і, з високою ймовірністю, буде зазнавати змін під час реалізації проєкту – питання лише в тому, наскільки передбачуваними та керованими будуть ці зміни.

З метою підвищення ефективності роботи експертів над побудовою ІСЕО рекомендується візуалізувати зміст проєктних робіт у формі, зручній для сприйняття та аналізу (в кожному конкретному випадку така візуалізація може бути різною). Як показує практика, деревовидна чи таблична структури є важкими для сприйняття, особливо, коли робота над ІСЕО здійснюється командою експертів. Також важливо явно робити припущення та ідентифікувати ризики, що стосуються «невідомих» частин змісту робіт проєкту (для цього, наприклад, можна скористатися відповідними атрибутами елементів оцінювання,

див. додаток 5). Необхідним є також забезпечення певного рівня гнучкості ICEO, іншими словами, важливо ідентифікувати ті її частини, які ймовірно зазнають змін під час розробки. Також, як показує практика, досить часто зміст робіт різних проєктів включають аналогічні задачі (наприклад, аутентифікація та авторизація, профіль користувача, захист персональних даних тощо). Тому завчасна підготовка та наступне використання шаблонних фрагментів ICEO може дати змогу підвищити ефективність роботи експертів.

Розробка методів побудови ICEO потребує додаткового дослідження і безпосередньо не входить до завдань цієї дисертаційної роботи.

2.3. Структура робочого часу інженера-розробника

Як відомо з практики, інженер-розробник затрачає не весь робочий час безпосередньо на написання програмного коду. При цьому структура робочого часу інженера-розробника змінюється залежно від фази реалізації проєкту: на початкових етапах значна частина робочого часу затрачається на реалізацію підготовчих задач, а на фінальних етапах – на задачі стабілізації. В цьому розділі опишемо підхід до структурування робочого часу інженера-розробника.

Нехай W – *проєкний робочий час* (англ. «project working time»), тобто повний робочий час, який затрачає учасник команди (не лише інженер-розробник), будучи залученим до реалізації проєкту. При цьому проєктний робочий час інженера-розробника включає такі складові:

1) M – *повний робочий час розробки* (англ. «full development working time»), що затрачається на реалізацію проєктних задач, визначених у ICEO (включаючи виправлення дефектів, взаємодію із іншими учасниками команди тощо);

2) G – *робочий час загальних проєктних активностей* (англ. «general project activities working time»), наприклад: щоденні наради, планування спринтів, презентації результатів роботи, налаштування робочого середовища на початку проєкту, стабілізаційні задачі перед релізом тощо;

3) N – *проєктний «неробочий» час* (англ. «project non-working time»), наприклад: відпустки, лікарняні, а також час простою, якщо у розробника немає задач чи виконання задач неможливе з певних причин (наприклад, через невирішені залежності);

4) R – *резерв проєктного робочого часу* (англ. «project working time contingency reserve»), що в разі «матеріалізації» ризиків може бути затрачений на кожну із трьох інших компонент структури робочого часу: розробку, загальні проєктні активності чи «неробочий» час.

В свою чергу, повний робочий час розробки M включає наступні дві складові: D – *робочий час розробки* (англ. «development working time»), який затрачається безпосередньо на реалізацію проєктних задач (як правило, на написання коду) та A – робочий час, затрачений на супровідні активності розробки, наприклад: написання модульних тестів, виправлення дефектів, взаємодію з іншими учасниками команди тощо.

«Неробочий» час N також включає дві складові: I – *проєктний час простою* (англ. «project idle time»), який є робочим часом, затраченим розробниками, коли вони фактично не виконують проєктних задач (для прикладу, така ситуація може виникнути в разі недостатності проєктних задач або коли реалізація задач розробника заблокована залежностями від інших ще нереалізованих задач) та O – *проєктний час відсутності* (англ. «project leaves time»), який включає плановані відпустки, незаплановані вихідні, лікарняні тощо. Варто відзначити, що «неробочий» час N не включає дні, що не входять до робочого календаря, наприклад: щотижневі вихідні та державні свята.

Отже, має місце наступний вираз для проєктного робочого часу інженера-розробника:

$$W = M + G + N + R = \left(D + A\right) + G + \left(I + O\right) + R. \qquad (2.1)$$

За визначенням змінні із (2.1) є невід'ємними: $W \geq 0$, $D \geq 0$, $A \geq 0$, $G \geq 0$, $I \geq 0$, $O \geq 0$, $R \geq 0$. Базовою одиницею вимірювання для цих змінних є людино-година.

Запропонована структура робочого часу інженера-розробника включена до системи рівнянь балансу робочого часу (див. пункти 2.7.2 та 2.8.2), що, в свою чергу, дає змогу отримувати оцінку тривалості проєкту, а також обирати оптимальний склад проєктної команди (див. підрозділ 2.11).

2.4. Оцінювання трудоємності

Для оцінювання проєктів з розробки ПЗ важливо мати уніфікований підхід до вимірювання трудоємності. Існуючі підходи до вимірювання трудоємності можна розділити на три головні категорії [33], [37]. До першої категорії належать підходи, що базуються на ідеї функційних точок (англ. «function points») [26], [27]. Згідно цього підходу трудоємність вимірюється у функційних точках, які потім трансформуються у часові одиниці. Такі поняття як «об'єктні точки» (англ. «object points») [28], «прецедентні точки» (англ. «use case points») [29], а також широко застосовні у agile-підходах «сторі-поінти» (англ. «story points») [30], [31], [52], [128] походять від ідеї функційних точок. Іншим відомим підходом до вимірювання трудоємності є визначення кількості логічних ліній програмного коду (так звана SLOC-метрика), що, зокрема, застосовується у COCOMO [24] та COCOMO II [28]. До третьої категорії належать підходи, що базуються на часових одиницях: людино-годинах, людино-днях, людино-місяцях тощо. Метод PERT [22], [23] є одним із найбільш поширених представників з цієї категорії. Ключовим недоліком підходів із першої категорії є труднощі із конвертацією «точкових» одиниць у часові (що, зокрема, є необхідним для оцінювання тривалості проєкту). Аналогічна проблема виникає з кількістю ліній вихідного коду, коли необхідно опиратися на статистичні дані для різних мов програмування, щоб трансформувати значення SLOC-метрики у часові одиниці. У зв'язку з цим методи оцінювання, розроблені в рамках цієї дисертаційної роботи, використовують для вимірювання трудоємності саме часові одиниці – людино-години.

Одним із ключових факторів, які впливають на продуктивність інженера-розробника, є його рівень компетентності (що, зазвичай, визначається згідно

матриці компетентностей організації). У зв'язку з цим на виконання однієї і тієї ж задачі інженери-розробники різних рівнів компетентності будуть затрачати відмінні обсяги робочого часу. Для визначення спроможності розробки команди, до складу якої входять інженери-розробники різних рівнів компетентності, необхідно визначити певний «спільний знаменник». Нижче представлений підхід «нормалізації» до визначення трудоємності, що базується на понятті інженера-розробника середнього рівня компетентності.

Інженером-розробником середнього рівня компетентності, ІРСРК (англ. «software developer of the average competency level», SDACL) називатимемо інженера-розробника, який володіє достатнім рівнем компетентності (тобто знаннями, навичками та досвідом), щоб працювати у складі проєктної команди, виконуючи задачі певного типу із прийнятним рівнем якості без необхідності постійного нагляду зі сторони більш кваліфікованих колег; при цьому такий інженер-розробник все ще має простір для покращення рівня володіння компетентностями з точки зору підвищення ефективності та збільшення складності виконуваних завдань. Згідно з визначенням ІРСРК є близьким до позиції так званого «мідл»-інженера (англ. «middle software engineer»). При цьому варто наголосити, що ІРСРК обумовлений контекстом конкретної організації, перш за все, її матрицею компетентностей, підходами до організації реалізації проєктів, а також корпоративною культурою. У випадку, якщо в організації немає матриці компетентностей для інженерів-розробників (тобто немає уніфікованого підходу до визначення рівня компетентності), підхід нормалізації не рекомендується застосовувати.

Нормалізованим робочим часом розробки, НРЧР (англ. «normalized development working time», NDWT) U називатимемо робочий час розробки, затрачений ІРСРК, що працює на повне навантаження і не залучений до роботи на інших проєктах, для реалізації певного обсягу проєктних задач. Якщо D робочий час розробки, затрачений іншим інженером-розробником на реалізацію тих самих задач, то має місце співвідношення:

$$U = \rho D, \tag{2.2}$$

де ρ називатимемо *коефіцієнтом продуктивності* (англ. «productivity coefficient»). Вважатимемо, що коефіцієнт продуктивності ρ визначається як:

$$\rho = \alpha\eta, \tag{2.3}$$

де α *– коефіцієнт продуктивності за рівнем компетентності, КПРК* (англ. «competency level productivity coefficient», CLPC), а η *– коефіцієнт продуктивності за ступенем залученості, КПСЗ* (англ. «involvement productivity coefficient», IPC).

У випадку ІРСРК $\alpha = 1$, якщо ж рівень компетентності інженера-розробника нижчий за «середній», то $0 < \alpha < 1$, а якщо вищий, то $\alpha > 1$. Якщо інженер-розробник не є компетентним у виконанні певного типу задач, то $\alpha = 0$.

КПСЗ η може набувати наступних значень:

$$\eta = \begin{cases} \{1\}, & \text{якщо } \phi = 1, \\ (0,1), & \text{якщо } 0 < \phi < 1 \text{ або } \phi > 1, \\ \{0\}, & \text{якщо } \phi = 0, \end{cases}$$

де ϕ – еквівалент повної зайнятості, ЕПЗ, інженера-розробника (англ. «full-time equivalent», FTE); $\phi = 1$ – повне залучення, $0 < \phi < 1$ – часткове залучення, $\phi > 1$ – понаднормове залучення, $\phi = 0$ – відсутність залучення.

Нормалізованою спроможністю розробки, НСР (англ. «normalized development capacity», NDC) C називатимемо максимально можливий обсяг НРЧР, який інженер-розробник (або команда інженерів-розробників) може затратити на реалізацію проєктних задач:

$$C = \max_{U \in \mathbb{U}} U, \tag{2.4}$$

де $\mathbb{U}$ множина всіх можливих значень НРЧР.

Нормалізованою оцінкою розробки, НОР (англ. «normalized development estimate», NDE) E певного обсягу проєктних задач є прогноз робочого часу розробки, вважаючи, що розробка буде виконана ІРСРК. Іншими словами, нормалі-

зована оцінка розробки є прогнозом НРЧР, який буде затрачений на етапі реалізації проєкту. Одиницею вимірювання НСР та НОР є людино-години.

На практиці НОР може бути отримана наступним чином. Експерт з метою визначення трудоємності елементу оцінювання «ставить» себе на місце ІРСРК та прогнозує, за скільки часу (в годинах) він виконає відповідну задачу за наступних умов:

1) інженер-розробник працюватиме лише над цією задачею (тобто немає інших задач, які він виконує паралельно);

2) інженер-розробник не спілкуватиметься з іншими учасниками команди;

3) інженер-розробник не братиме участь у загальних проєктних активностях;

4) задача не матиме залежностей від інших задач чи зовнішніх (відносно проєкту) факторів;

5) інженер-розробник працюватиме повний робочий день (тобто не передбачається часткове чи понаднормове залучення).

Очевидно, що вище згадані умови неможливо виконати на практиці. Тому для отримання прогнозу «реального» часу розробки, отримана НОР «доповнюється» іншими часовими параметрами відповідно до структури робочого часу (2.1) та системи рівнянь балансу робочого часу (див. пункти 2.7.2 та 2.8.2).

Варто підкреслити наступні переваги підходу до визначення трудоємності, що базується на часових одиницях (людино-годинах), які пов’язані із продуктивністю ІРСРК:

1) можливість відносно легко обґрунтовувати надану оцінку та порівнювати її з аналогічними оцінками колег;

2) оцінка у людино-годинах (або аналогічних одиницях) досить просто трансформується у тривалість (чого, для прикладу, не скажеш про оцінки у «точкових» одиницях чи значеннях SLOC-метрики, представленнях яких у часовому еквіваленті вимагає, зазвичай, наявності історичних даних);

3) оцінки трудоємності, отримані за допомогою описаного вище підходу, є незалежними від графіку реалізації проєкту чи складу команди (учасники яких

можуть мати різний рівень компетентності та ступінь залученості до проєкту), що, в свою чергу, дає змогу оперувати такими оцінками для побудови різних комбінацій графіку реалізації проєкту та складу команди.

2.5. Графік реалізації проєкту

Під *графіком реалізації проєкту* (англ. «project implementation schedule») будемо розуміти часовий проміжок з ієрархічною структурою фаз, упродовж якого здійснюється реалізація проєкту (рис. 2.2). Ієрархічна структура графіку реалізації проєкту передбачає, що на найвищому рівні знаходиться проміжок часу H, який відповідає реалізації всього проєкту, а на другому рівні визначається поділ на фази – T є множиною фаз проєкту[1]. В свою чергу, кожна із фаз проєкту $T_D \subseteq T$ є множиною підфаз (наприклад, спринтів). Важливою характеристикою кожної із фаз проєкту є її тривалість. У певних випадках (наприклад, для детального оцінювання) можлива прив'язка фаз проєкту до календарних дат.

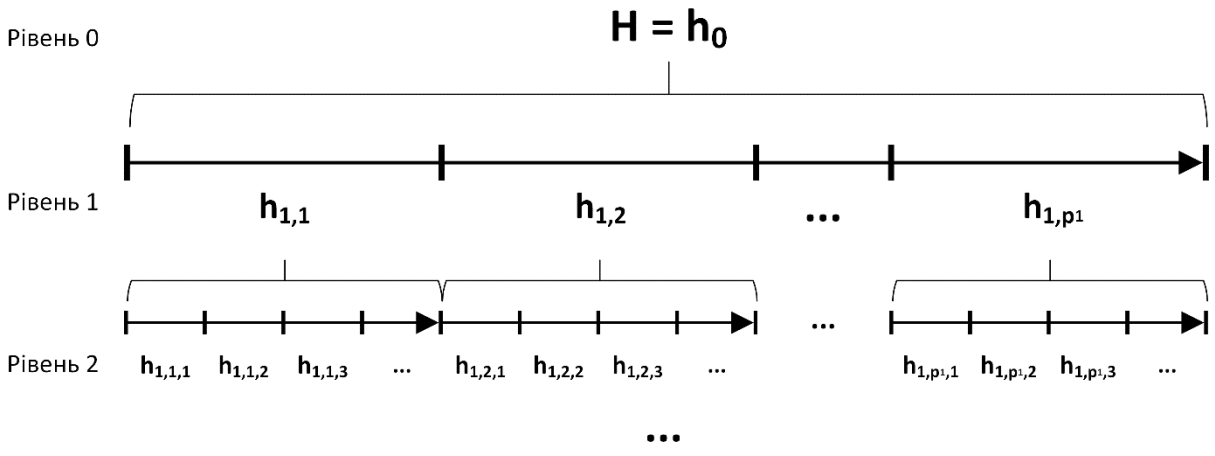


Рис. 2.2. Ієрархічна структура графіку реалізації проєкту з розробки програмного забезпечення

Шаблоном графіку реалізації проєкту (англ. «project implementation schedule template») називатимемо ієрархічну структуру фаз проєкту та значень параметрів, асоційованих з цими фазами (наприклад, тривалості фаз), у якій

[1] Вважатимемо, що на множині фаз реалізації проєкту задано відношення порядку, так що для будь-яких $h_{1,i} \in H$ та $h_{1,j} \in H$ (за умови, що $h_{1,i} \neq h_{1,j}$), $h_{1,i} < h_{1,j}$ тоді і лише тоді, коли $h_{1,i}$ виконується раніше ніж $h_{1,j}$.

закладено певний «простір» для варіації значень цих параметрів (наприклад, можливість зміни в певних межах тривалості фази основної розробки), що дає змогу побудувати різні варіанти графіків реалізації проєкту на основі одного і того ж шаблону.

Кількість рівнів ієрархії графіку реалізації проєкту під час планування залежить від етапу оцінювання: як будемо бачити нижче, для попереднього оцінювання досить лише рівня «0» (тобто відсутнє розбиття проєкту на фази), для проміжного оцінювання обов'язковим є розбиття до рівня «1», а для детального оцінювання необхідне навіть більш «глибоке» розбиття до рівня «2» (наприклад, до спринтів). Під час реалізації проєкту кількість рівнів ієрархії залежить від специфіки конкретного проєкту. Як показує практика, найбільш поширеною є трирівнева ієрархія, що включає розбиття на фази із наступним розбиттям кожної фази на спринти.

2.6. Проєктна команда

Проєкти з розробки ПЗ у переважній більшості випадків виконуються командами, кількість учасників яких може варіювати від декількох до десятків чи навіть сотень осіб.

Під *складом проєктної команди* розумітимемо множину ролей учасників команди T та асоційованих з ними параметрів, наприклад, ЕПЗ. Проєктна команда включає ролі двох типів: *ролі інженерів-розробників* $T_D \subseteq T$ та *«нерозробницькі» ролі* $T_{ND} \subset T$ (проєктні менеджери, архітектори, бізнес-аналітики, дизайнери, інженери з тестування тощо). Варто відзначити, що справедливі наступні твердження: $T_D \cup T_{ND} = T$ та $T_D \cap T_{ND} = \varnothing$. Вважатимемо, що $T_D \neq \varnothing$, в той час як T_{ND} може бути й порожньою множиною. Ключовою відмінністю між цими двома категоріями ролей є те, що робочий час першої з них включається у НСР та НОР, в той час як робочий час ролей з другої категорії не входить у НСР та НОР.

Нормалізованим еквівалентом повної зайнятості (НЕПЗ, англ. «normalized full-time equivalent», NFTE) інженера-розробника ψ називатимемо:

$$\psi = \rho\,\phi, \tag{2.5}$$

де ρ – це коефіцієнт продуктивності, визначений у (2.2) та (2.3), а ϕ – відповідний ЕПЗ. Цей термін дає змогу поєднати значення ЕПЗ інженерів-розробників із НОР та НСР.

Залежно від диференціації спеціалізацій команда розробників може належати до одного із трьох типів:

1. Найпростішим типом є команда *без диференціації спеціалізацій*, коли спеціалізації інженерів-розробників не розрізняються (англ. «non-differentiated specializations»). При цьому це не значить, що ці спеціалізації не визначаються пізніше на етапі реалізації проєкту.

2. Наступним типом є команда з *диференційованими спеціалізаціями* (англ. «differentiated specializations»), в якій кожен із інженерів-розробників належить до однієї (і лише однієї) із спеціалізацій.

3. До третього типу належать команди із *змішаними спеціалізаціями* (англ. «mixed specializations»), коли один інженер-розробник може виконувати задачі, що відносяться до різних спеціалізацій. Така ситуація, наприклад, має місце, коли один із інженерів-розробників може виконувати як задачі з розробки графічного інтерфейсу, так і задачі програмування серверної частини (так звані «full-stack»-розробники). У випадку першого типу, команди без диференціації спеціалізацій, НОР та НСР є скалярними значеннями, в той час як для інших двох типів команд НОР та НСР є векторами, кожен елемент яких відповідає певній спеціалізації.

Уточнимо вираз для структури робочого часу розробника (2.1), враховуючи диференціацію спеціалізацій. Нехай $M^{(s)}$ – повний робочий час розробки, який затрачає інженер-розробник для реалізації проєктних задач, що відносяться до спеціалізації s, тоді сумарний повний робочий час розробки виражається як:

$$M = \sum_{s \in S} M^{(s)} = \sum_{s \in S} \left(D^{(s)} + A^{(s)} \right), \tag{2.6}$$

де S – множина спеціалізацій, якими володіє інженер-розробник. Тоді вираз (2.1) із врахуванням (2.6) запишеться наступним чином:

$$W = \sum_{s \in S} M^{(s)} + G + \left(O + I \right) + R = \sum_{s \in S} \left(D^{(s)} + A^{(s)} \right) + G + \left(O + I \right) + R. \tag{2.7}$$

Нехай $\phi^{(s)}$ – ЕПЗ, що відповідає спеціалізації $s \in S$, при цьому має місце співвідношення:

$$\phi = \sum_{s \in S} \phi^{(s)}, \tag{2.8}$$

тоді повний робочий час розробки $M^{(s^*)}$ визначається так:

$$M^{(s^*)} = \frac{\phi^{(s^*)}}{\phi} \sum_{s \in S} M^{(s)}, \; s^* \in S. \tag{2.9}$$

Для випадку команди з диференційованими спеціалізаціями, коли інженер-розробник володіє лише однією спеціалізацією s^*, $\phi = \phi^{(s^*)} \geq 0$ при цьому $\phi^{(s)} = 0$ для $\forall s \in \left\{ s \in S, s \neq s^* \right\}$; отже, $M = M^{(s^*)}$.

Коефіцієнт продуктивності ρ, визначений згідно (2.3), задається для кожної спеціалізації окремо, оскільки рівень компетентності інженера-розробника у різних спеціалізаціях може відрізнятися, а отже, будуть різними і відповідні значення КПРК α:

$$\rho^{(s)} = \alpha^{(s)} \eta, \; s \in S. \tag{2.10}$$

Підкреслимо, що КПСЗ η залежить від залученості інженера-розробника у проєкт в цілому і не залежить від спеціалізацій. Звідси випливає, що НЕПЗ також залежить від спеціалізації:

$$\psi^{(s)} = \rho^{(s)} \phi^{(s)}, s \in S, \tag{2.11}$$

де $\rho^{(s)}$ і $\phi^{(s)}$ – відповідно коефіцієнт продуктивності та ЕПЗ для спеціалізації s.

Залежно від потреб конкретного проєкту команда може мати різний склад. Однак, як відомо з практики, можна виокремити «схожі» композиції проєктних

команд. *Шаблоном складу проєктної команди* (англ. «project team composition template») називатимемо сукупність множини ролей учасників команди $T = T_D \cup T_{ND}$ та пов'язані з кожною із ролей значення ЕПЗ ϕ, а також множини правил, що визначають залежності між цими значеннями ЕПЗ. Варто підкреслити, що значення ЕПЗ для однієї і тієї ж ролі можуть відрізнятися залежно від фази реалізації проєкту. Відзначимо також, що кожна з ролей інженерів-розробників $t \in T_D$ включає інформацію про його спеціалізації та значення КПРК α для кожної із спеціалізацій. Інший важливий параметр, КПСЗ η, є похідним від значення відповідного ЕПЗ та залежить від кількості спеціалізацій та ролей відповідного розробника. Для прикладу, у випадку, коли учасник команди може на 50% залученості виконувати роль інженера-розробника, а на інші 50% – роль технічного лідера команди, його КПСЗ $0 < \eta < 1$, оскільки на продуктивність ролі інженера-розробника впливатиме виконання лідерської ролі.

Підкреслимо також, що шаблон складу команди нерозривно пов'язаний із шаблоном графіку реалізації проєкту, оскільки значення таких параметрів як, наприклад, ЕПЗ можуть мати різні значення залежно від фази проєкту. Змінюючи варіативні частини шаблонів складу команди та графіку реалізації проєкту (в межах заданої множини правил), можна отримати різні варіанти комбінацій композиції команди та графіку реалізації проєкту. Як показано нижче, така варіативність використовується у методі підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту (див. підрозділ 2.11).

2.7. Попереднє оцінювання

2.7.1. Попереднє оцінювання та його особливості

Попереднє оцінювання (англ. «preliminary estimation») застосовується на початкових етапах життєвого циклу з метою отримання приблизного розуміння обсягів ресурсів та часу, необхідних для реалізації проєкту (етап оцінювання 1 на рис. 2.1). Часто попереднє оцінювання неформально називають «наближеним оцінюванням» (англ. «rough order of magnitude») [121], [124] або «болпарк»-

оцінюванням (англ. «ballpark figures»). Розроблений метод попереднього оцінювання [5] застосовний за таких умов:

1) оцінювання здійснюється на початкових етапах життєвого циклу;

2) обмежений час для підготовки оцінки;

3) прийнятність низької точності оцінки;

4) оцінка не буде використовуватися для взяття зобов'язань щодо реалізації проєкту;

5) низький рівень деталізації змісту проєктних робіт;

6) необхідність порівняння оцінок кількох альтернативних потенційних сценаріїв реалізації проєкту.

Для отримання оцінки на цьому етапі застосовують підхід «згори – вниз», коли опрацьовуються лише «високорівневі» елементи оцінювання із ІСЕО. Згодом, на наступному етапі, проміжному оцінюванні, при збільшенні рівня деталізації ІСЕО попередня оцінка буде підкріплена результатами застосування діаметрально протилежного підходу – оцінювання «знизу – вгору».

Порівняння попереднього оцінювання з іншими етапами оцінювання представлено у додатку 3.

2.7.2. Система рівнянь балансу робочого часу

Система рівнянь балансу робочого часу проєктної команди визначає взаємозв'язки із основними складовими оцінки: структурою робочого часу інженерів-розробників, складом проєктної команди, а також НОР. Для попереднього оцінювання пропонується наступна система рівнянь балансу робочого часу:

$$\begin{cases} W^m = \xi D^m, m \in T_D, \\ W^m = \phi^m L, m \in T, \\ E = \sum_{m \in T_D} \rho^m D^m, \end{cases} \tag{2.12}$$

де T – множина проєктних ролей, $T_D \subseteq T$ – підмножина ролей інженерів-розробників, W^m – проєктний робочий час ролі $m \in T$, D^m – робочий час розробки ролі $m \in T_D$, $\xi > 1$ – коефіцієнт робочого часу розробки, КРЧР (англ. «development

working time coefficient», DWTC), ϕ^m – ЕПЗ ролі $m \in T$, L – тривалість проєкту, E – НОР всього змісту проєктних робіт, ρ^m – коефіцієнт продуктивності ролі $m \in T_D$. Пояснення щодо КРЧР представлені у пунктах 2.7.6 та 2.7.7.

Система рівнянь (2.12) базується на наступних спрощеннях (у порівнянні із відповідною системою рівнянь для проміжного оцінювання, див. пункт 2.8.2):

1) часова шкала реалізації проєкту не розділена на фази чи спринти (іншими словами, застосований лише «нульовий» рівень ієрархічної структури графіку реалізації проєкту, рис. 2.2);

2) проєктна команда не змінюється в процесі реалізації (іншими словами, вважається, що команда буде у своєму максимальному складі упродовж всієї реалізації проєкту);

3) немає диференціації спеціалізацій розробників;

4) лінійна залежність між проєктним робочим часом W^m інженерів-розробників та робочим часом розробки D^m через КРЧР ξ.

Закономірності, визначені системою рівнянь (2.12), будуть далі використані для оцінювання НСР та тривалості проєкту.

2.7.3. Оцінювання нормалізованої спроможності розробки

Із визначення НСР (2.3) та системи рівнянь балансу робочого часу (2.12) отримуємо наступний вираз:

$$C^m = \frac{L}{\xi}\psi^m, m \in T_D, \tag{2.13}$$

де C^m – НСР ролі $m \in T_D$, ψ^m – НЕПЗ ролі $m \in T_D$, ξ – КРЧР, L – тривалість проєкту. Тоді для всієї проєктної команди НСР C обчислюється наступним чином:

$$C = \sum_{m \in T_D} C^m = \frac{L}{\xi} \sum_{m \in T_D} \psi^m . \tag{2.14}$$

Зауважимо, що НСР, отримана згідно (2.14), є скалярним значенням та застосовна до команди розробників з недиференційованими спеціалізаціями. Якщо

ж результати попередньої оцінки все таки повинні враховувати спеціалізації інженерів-розробників, то необхідно визначити відсоткове співвідношення трудоємності для кожної із спеціалізацій та застосувати його до НСР (2.14).

2.7.4. Оцінювання тривалості проєкту

Для попереднього оцінювання тривалості проєкту головним критерієм є те, що проєктна команда повинна мати достатню НСР C, щоб реалізувати проєктні роботи, НОР яких складає E:

$$C = \frac{L}{\xi} \sum_{m \in T_D} \psi^m \geq E. \tag{2.15}$$

Тоді тривалість проєкту оцінюється за наступною нерівністю:

$$L \geq \frac{\xi E}{\sum_{m \in T_D} \psi^m}. \tag{2.16}$$

Легко бачити, що оцінка тривалості (2.16) явно не враховує залежностей між елементами оцінювання, наприклад, коли реалізація одного елементу можлива після завершення іншого. Вплив таких залежностей між елементами оцінювання на тривалість проєкту в цілому визначається через значення КРЧР ξ (див. пункти 2.7.6 та 2.7.7).

2.7.5. Оцінювання трудоємності групування за схожістю

Для оцінювання трудоємності змісту проєктних робіт у випадку попереднього оцінювання, коли зміст проєктних робіт, зазвичай, не є деталізованим, можна скористатися методом, який, з однієї сторони, дає змогу скоротити час роботи експертів, а з іншої передбачає техніку візуалізації.

Нехай x – простий елемент оцінювання, що належить до ІСЕО X. «*Розміром» елементу оцінювання* (англ. «estimable item point», EIP) x називатимемо додатне число $x[\text{EIP}]$, що представляє відносну трудоємність елементу x (у порівнянні з іншими простими елементами оцінювання, що належать ІСЕО X). *Невизначеністю елементу оцінювання* (англ. «estimable item uncertainty», EIU) x називатимемо безрозмірне додатне число $x[\text{EIU}]$, що позначає рівень невідомо-

го, пов'язаного з x. Іншими словами, чим більшим є значення $x[\mathrm{EIU}]$, тим менш точним є значення $x[\mathrm{NDE}]$ і навпаки – менше значення $x[\mathrm{EIU}]$ означає вищу точність $x[\mathrm{NDE}]$. Підкреслимо, що атрибути EIP та EIU визначаються лише для простих елементів оцінювання і не застосовуються до композитних.

Значення атрибутів EIP та EIU базуються на експертній оцінці. В якості прикладу візьмемо ICEO, зображену на рис. 2.3. Тоді, відштовхуючись від ідеї методу «групування за схожістю» (англ. «affinity grouping») [52], розмістимо прості елементи оцінювання на координатній площині, горизонтальна вісь якої відповідає за значення EIP, а вертикальна – за EIU (рис. 2.4). В результаті отримаємо оцінки EIP та EIU, базуючись на відносному розміщенні елементів оцінювання на цій координатній площині.

Для трансформації EIP у HOP будемо користуватися співвідношенням:

$$x[\mathrm{NDE}_{\mathrm{basic}}] = p \cdot x[\mathrm{EIP}], \tag{2.17}$$

де $p > 0$ – HOP, що відповідає одному EIP. Тоді оптимістична та песимістична HOP отримуються за формулами:

$$\begin{aligned} x[\mathrm{NDE}_{\mathrm{opt}}] &= (1 - u_1 \cdot x[\mathrm{EIU}]) \cdot x[\mathrm{NDE}_{\mathrm{basic}}], \\ x[\mathrm{NDE}_{\mathrm{psm}}] &= (1 + u_2 \cdot x[\mathrm{EIU}]) \cdot x[\mathrm{NDE}_{\mathrm{basic}}], \end{aligned} \tag{2.18}$$

де $x[\mathrm{NDE}_{\mathrm{opt}}]$ та $x[\mathrm{NDE}_{\mathrm{psm}}]$ – відповідно оптимістична та песимістична HOP, $0 \le u_1 < 1$ та $u_2 \ge 0$ – коефіцієнти, що визначають відхилення оптимістичної та песимістичної HOP від базової. Значення параметрів p, u_1, u_2 визначаються експертами або обчислюються на основі історичних даних.

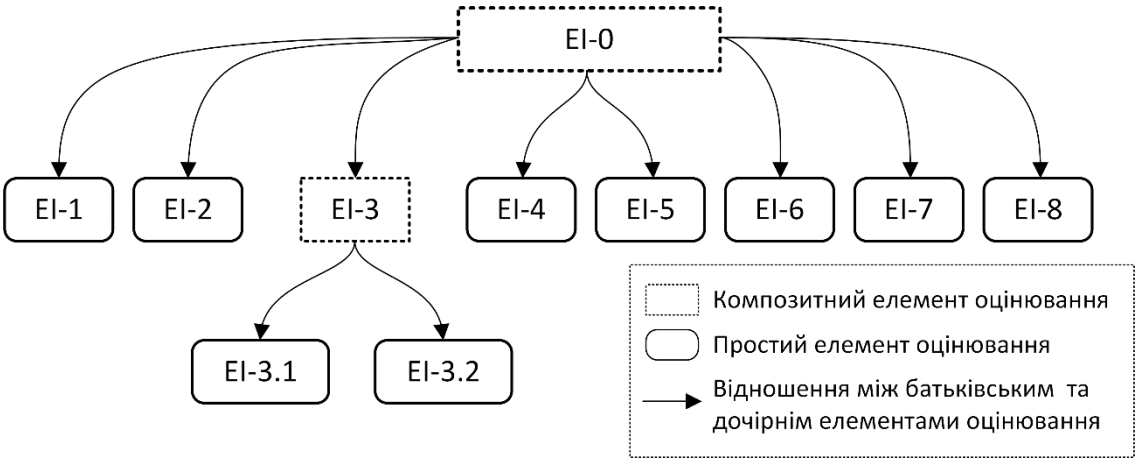


Рис. 2.3. Приклад ієрархічної структури елементів оцінювання

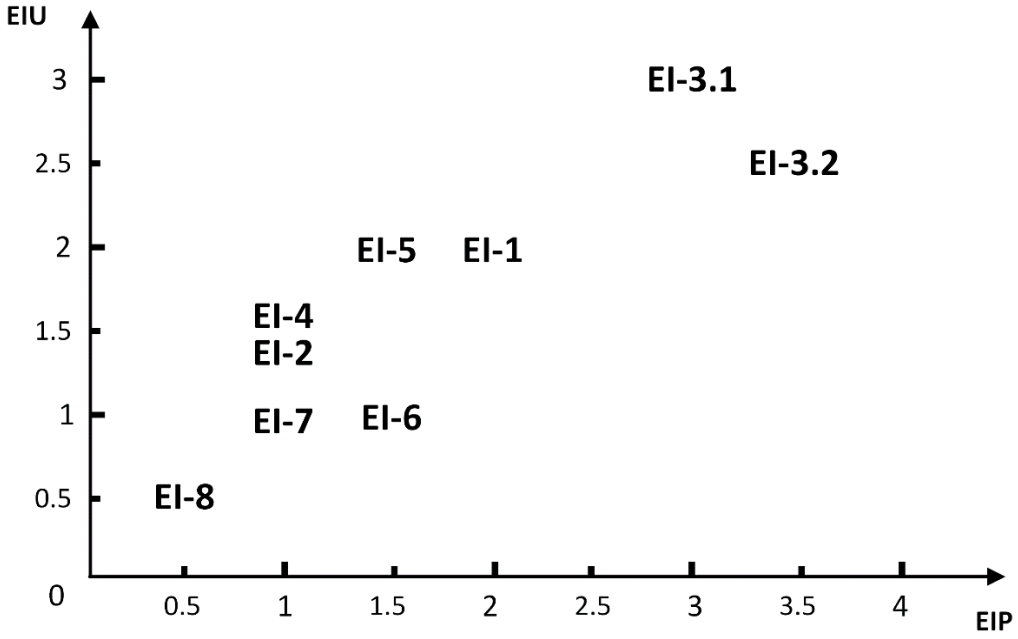


Рис. 2.4. Приклад групування елементів оцінювання за трудоємністю та рівнем невизначеності

Очевидно, що розроблений метод оцінювання трудоємності застосовний до відносно невеликої кількості елементів оцінювання (що прийнятно для попереднього оцінювання, для якого не передбачається високий рівень деталізації ІСЕО). В разі необхідності застосування цього методу до великих ІСЕО (тобто, коли виникають складнощі з візуальним розміщенням елементів оцінювання на координатній площині через їх кількість), елементи можна розділити на частини, кожну з яких опрацьовувати окремо.

2.7.6. Коефіцієнт робочого часу розробки

В цьому пункті розкриємо природу коефіцієнту робочого часу розробки (КРЧР) ξ, застосованого у системі рівнянь балансу робочого часу (2.12).

Згідно (2.7) визначено наступне співвідношення між проєктним робочим часом та робочим часом розробки для команди інженерів-розробників:

$$\bar{W}_D = \sum_{m\in\bar{T}_D}\sum_{h\in\bar{H}} \bar{W}_h^m = \sum_{m\in\bar{T}_D}\sum_{h\in\bar{H}} \left[\sum_{s\in\bar{Q}} \left(\bar{D}_h^{m(s)} + \bar{A}_h^{m(s)} \right) + \bar{G}_h^m + \left(\bar{O}_h^m + \bar{I}_h^m \right) + \bar{R}_h^m \right], \tag{2.19}$$

де $\bar{T}_D$ –команда інженерів-розробників; $\bar{W}_D$ – проєктний робочий час команди інженерів-розробників; $\bar{H}$ – множина фаз проєкту; $\bar{Q}$ – множина спеціалізацій інженерів-розробників; $\bar{W}_h^m$, $\bar{D}_h^m$, $\bar{A}_h^m$, $\bar{G}_h^m$, $\bar{O}_h^m$, $\bar{I}_h^m$, $\bar{R}_h^m$ – відповідно проєктний робочий час, робочий час розробки, робочий час супровідних активностей розробки, робочий час загальних проєктний активностей, час відсутності, час простою та резерв проєктного робочого часу ролі $m \in \bar{T}_D$ на фазі проєкту $h \in \bar{H}$.

У (2.12) КРЧР ξ використаний з метою зменшення складності шляхом скорочення кількості змінних у системі рівнянь балансу робочого часу, відштовхуючись від припущення, що між проєктним робочим часом та робочим часом розробки існує прямопропорційна залежність:

$$W_D = \sum_{m\in T_D} W^m = \xi \sum_{m\in T_D} D^m, \tag{2.20}$$

де T_D – команда інженерів-розробників, ξ – КРЧР, W_D – проєктний робочий час команди інженерів-розробників, D^m – робочий час розробки ролі $m \in T_D$.

Зауважимо, що вираз (2.20) застосовний до попереднього оцінювання, в той час як (2.19) використовується для проміжного та детального оцінювань (див. підрозділи 2.8 та 2.9). Тоді, відштовхуючись від концепції «конусу невизначеності», оцінка, отримана із застосуванням (2.20), є «ширшою» за оцінку, що базується на (2.19) (для того ж самого змісту проєктних робіт); при цьому (2.20) не повинно призводити до надто великої оцінки, щоб никнути формування хибного враження щодо бюджету та тривалості проєкту:

$$q_1 \cdot \bar{W}_D \le W_D \le q_2 \cdot \bar{W}_D, \qquad (2.21)$$

де $0 < q_1 \le 1$ та $q_2 \ge 1$ – визначають відповідно нижню та верхню межу для оцінки W_D відносно оцінки $\bar{W}_D$.

Підсумовуючи вище сказане, КРЧР ξ з системи рівнянь балансу робочого часу (2.12) залежить від:

1) структури робочого часу інженера-розробника;

2) графіку реалізації проєкту;

3) складу команди інженерів-розробників, враховуючи їх змінне залучення до різних фаз проєкту.

Для практичного визначення КРЧР рекомендується використовувати дерево рішень, яке, базуючись на історичних даних, давало б змогу визначати значення КРЧР залежно від специфіки конкретного проєкту, що оцінюється. Підкреслимо, що побудова такого дерева рішень не входить до завдань цієї дисертаційної роботи.

2.7.7. Альтернативний підхід до визначення коефіцієнту робочого часу розробки

В цьому пункті розглянемо підхід до визначення КРЧР ξ, альтернативний до того, який був продемонстрований вище. Згідно системи рівнянь (2.8) КРЧР ξ визначається наступним чином:

$$W_D = \xi D, \qquad (2.22)$$

де $W_D = \sum_{m \in T_D} W^m$ – сумарний проєктний робочий час інженерів-розробників, $D = \sum_{m \in T_D} D^m$ – сумарний робочий час розробки. Однак, до визначення КРЧР можна підійти дещо по-іншому:

$$W_D = \kappa E, \qquad (2.23)$$

де W_D – сумарний проєктний робочий час інженерів-розробників, $\kappa > 1$ – альтернативний КРЧР, E – НОР змісту проєктних робіт. Тоді з (2.23) випливає наступна система рівнянь балансу робочого часу:

$$\begin{cases} \sum\limits_{m \in T_D} W^m = \kappa E, \\ W^m = \phi^m L,\ m \in T, \\ E = \sum\limits_{m \in T_D} \rho^m D^m. \end{cases} \tag{2.24}$$

В свою чергу, з системи (2.24) отримуємо вирази для оцінювання НСР:

$$C = \frac{L}{\kappa} \sum_{m \in T_D} \phi^m \tag{2.25}$$

та тривалості проєкту:

$$L \geq \frac{\kappa E}{\sum\limits_{m \in T_D} \phi^m}. \tag{2.26}$$

Легко бачити, що (2.25) та (2.26) є досить схожими відповідно до (2.14) та (2.16) з тією різницею, що останні використовують НЕПЗ $\sum\limits_{m \in T_D} \psi^m$. Для вибору між двома підходами до визначення КРЧР рекомендується брати до уваги такі фактори як необхідність врахування коефіцієнту продуктивності (що забезпечує «нормалізацію» ЕПЗ), а також наявні історичні дані, на основі яких визначається значення КРЧР. Зауважимо, що КРЧР, визначений згідно (2.12), вимагає вищої гранулярності історичних даних, ніж альтернативний, що заданий у (2.23).

2.7.8. Схема застосування попереднього оцінювання

В цьому розділі представимо схему застосування попереднього оцінювання на практиці.

Вхідними даними для попереднього оцінювання є:

1) інформація про проєкт, що, зокрема, включає дані про замовника, сферу бізнесу, відомості про аналогічні проєкти;

2) функційні та нефункційні вимоги до проєкту;

3) технічна архітектура.

Для отримання попередньої оцінки необхідно виконати наступні кроки:

Крок 1. Побудувати ІСЕО. Цей крок, зокрема, включає аналіз елементів оцінювання та присвоєння їм атрибутів (див. підрозділ 2.2 та додаток 5). Зауважимо, що попереднє оцінювання не вимагає високого ступеня деталізації ІСЕО.

Крок 2. Надати оптимістичну та песимістичну НОР. Для оцінювання трудоємності можна скористатися, для прикладу, 3-точковим оцінюванням PERT або підходом, описаним у пункті 2.7.5.

Крок 3. Сформувати склад проєктної команди для оптимістичної та песимістичної оцінок. Для цього необхідно визначити множину ролей та задати значення ЕПЗ для кожної з них. Для ролей інженерів-розробників потрібно також вказати значення КПРК, КПСЗ та визначити на їх основі значення коефіцієнту продуктивності і НЕПЗ (див. підрозділ 2.6).

Крок 4. Задати значення КРЧР, користуючись історичними даними або, якщо історичних даних немає, покладаючись на експертну оцінку. Для визначеності вважатимемо, що КРЧР задано згідно (2.8).

Крок 5. Оцінити оптимістичну та песимістичну тривалості проєкту, користуючись (2.12).

Візуалізація схеми застосування попереднього оцінювання представлена у вигляді UML-діаграми активностей на рис. 2.5.

Застосування попереднього оцінювання забезпечує отримання наступних результатів:

1) ІСЕО, що представляє структурований зміст проєктних робіт;

2) оптимістична та песимістична НОР для елементів оцінювання, що належать сформованій ІСЕО;

3) склад проєктної команди для оптимістичної та песимістичної оцінок;

4) оптимістична та песимістична оцінки тривалості проєкту.

Попереднє оцінювання застосовується для кожного із потенційних сценаріїв реалізації проєкту, даючи змогу порівнювати ці сценарії. Зауважимо, що описані вище оптимістична та песимістична оцінки є складовими одного сценарію, а не двох різних сценаріїв.

Легко бачити, що простота попереднього оцінювання робить можливою його автоматизацію найпростішими засобами, наприклад з використанням Microsoft Excel. Разом з тим, підбір складу проєктної команди та тривалості проєкту (кроки 3 та 5) можуть бути реалізовані із застосуванням методу підтримки прийняття рішень, що ґрунтується на задачі багатоцільової оптимізації та розв'язанні задач цілочисельного програмування (див. підрозділ 2.11).

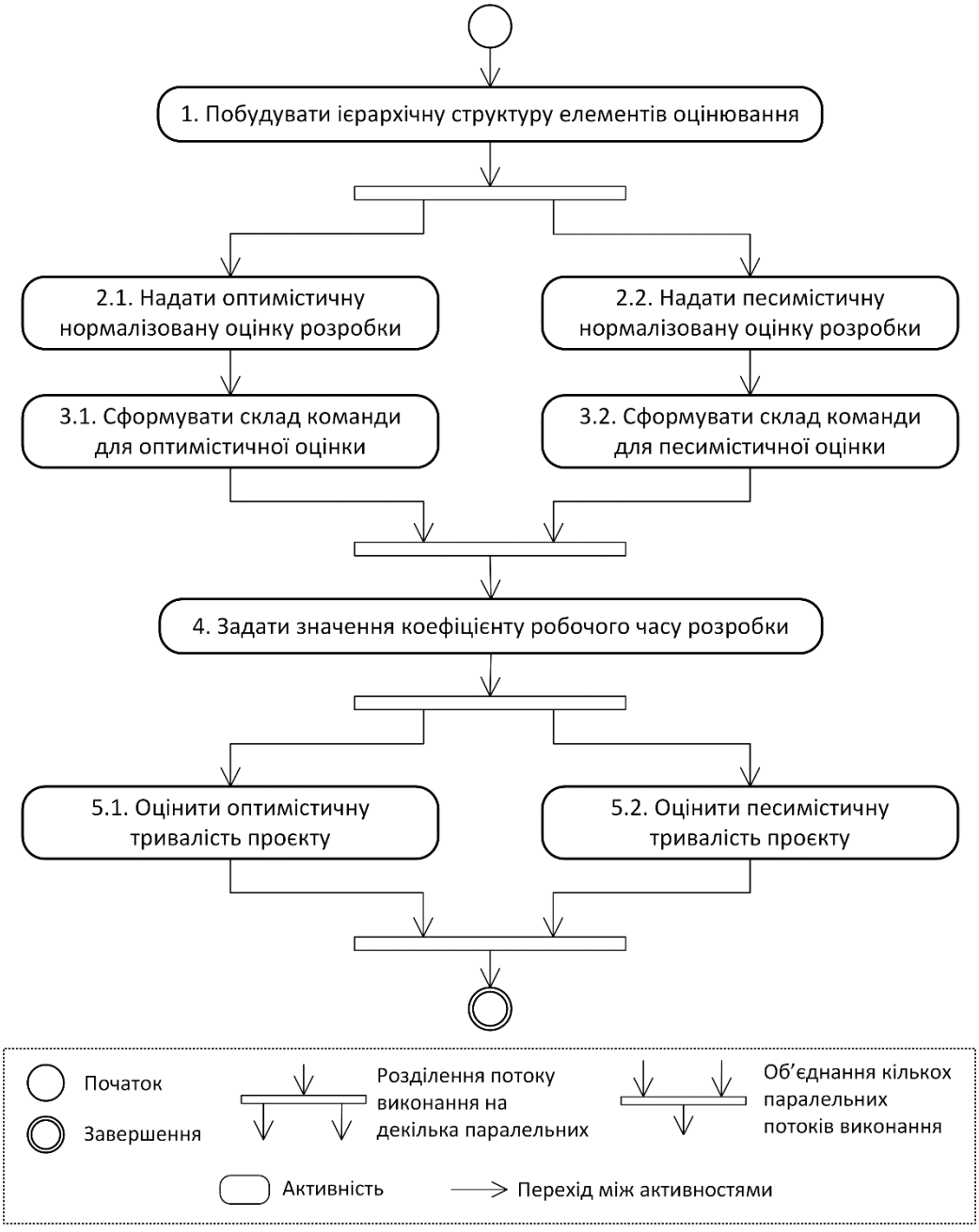


Рис. 2.5. UML-діаграма активностей схеми застосування попереднього оцінювання

Підкреслимо, що результати застосування попереднього оцінювання не рекомендується використовувати для взяття зобов'язань щодо реалізації проєкту (наприклад, для підписання договору із замовником). Їх призначення полягає у інформуванні зацікавлених сторін щодо оцінки та порівнянні оцінок для різних сценаріїв. Результати попереднього оцінювання передаються в якості вхідних даних для наступного етапу – проміжного оцінювання (див. підрозділ 2.8).

2.7.9. Застереження щодо застосування попереднього оцінювання

Попереднє оцінювання не слід застосовувати у таких випадках:

1. Коли рівень невизначеності настільки великий, що неможливо визначити зміст проєктних робіт та побудувати навіть високорівневу ICEO. В таких випадках пропонується застосовувати початкове оцінювання (етап оцінювання «0» на рис. 2.1), що передбачає застосування методів оцінювання за «схожістю» [57], [58], [59], [60].

2. В ситуаціях, коли важко задати КРЧР, використання попереднього оцінювання також не рекомендується. Наприклад, така ситуація може мати місце, коли продуктивність проєктної команди важко спрогнозувати (наприклад, у випадку використання нових технологій розробки). Як рішення, може бути відразу застосовано проміжне оцінювання (див. підрозділ 2.8) [4], [16], що, в свою чергу, вимагає детальнішого аналізу та більшої кількості параметрів.

3. Попереднє оцінювання не рекомендується застосовувати також у випадках, коли склад проєктної команди суттєво змінюється упродовж реалізації проєкту. В таких ситуаціях краще застосовувати проміжне оцінювання (див. підрозділ 2.8) [4], [15].

4. Як видно з (2.15), тривалість проєкту визначається значеннями HCP та HOP. Однак, у випадку «жорстких» залежностей між проєктними задачами, а також залежностей від зовнішніх (відносно проєкту) факторів такий підхід до визначення тривалості проєкту не буде працювати належним чином. Для визначення тривалості таких проєктів рекомендується застосовувати інші методи, наприклад, CPM [25] та PERT [22], [23].

2.8. Проміжне оцінювання

2.8.1. Проміжне оцінювання та його особливості

Ціллю проміжного оцінювання (англ. «intermediate estimation») є надання оцінки, точнішої за ту, яка була отримана на етапі попереднього оцінювання (звісно, у випадку підготовки попередньої оцінки). Як і попереднє оцінювання, проміжне оцінювання виконується під час аналізу та проєктування і, не зважаючи на вищу точність, його результати все ж не рекомендується використовувати для взяття зобов'язань щодо реалізації проєкту. Проміжне оцінювання відрізняється від попереднього наступним:

1) часова шкала реалізації проєкту розділена на фази (іншими словами, застосований рівень «1» ієрархічної структури графіку реалізації проєкту, див. підрозділ 2.5 та рис. 2.2);

2) склад проєктної команди може відрізнятися в для різних фаз проєкту;

3) команда інженерів-розробників має диференційовані або змішані спеціалізації (див. підрозділ 2.6);

4) залежність між проєктним робочим часом інженерів-розробників та робочим часом розробки визначається згідно структури робочого часу (2.7) без застосування КРЧР.

Порівняння проміжного оцінювання з іншими етапами оцінювання представлено у додатку 3.

2.8.2. Система рівнянь балансу робочого часу

Система рівнянь балансу робочого часу для проміжного оцінювання аналогічна системі (2.12) з тією основною відмінністю, що застосовується структура робочого часу інженера-розробника (2.7) і не використовується КРЧР:

$$\begin{cases} W_h^m = \sum\limits_{s\in S} \delta_h^{m(s)} \left(D_h^{m(s)} + A_h^{m(s)} \right) + G_h^m + \left(O_h^m + I_h^m \right) + R_h^m, h \in H, m \in T_D, \\ W_h^m = \phi_h^m L_h, h \in H, m \in T, \\ E^{(s)} = \sum\limits_{h\in H} \sum\limits_{m\in T_D} \alpha^{m(s)} \eta_h^m D_h^{m(s)}, s \in S, \end{cases} \quad (2.27)$$

де T – множина ролей проєктної команди; $T_D \subseteq T$ – підмножина ролей інженерів-розробників; H – множина фаз проєкту; S – множина спеціалізацій інженерів-розробників; W_h^m – проєктний робочий час ролі $m \in T$ на фазі $h \in H$; $D_h^{m(s)}$ – робочий час розробки на фазі $h \in H$ для ролі $m \in T_D$ та спеціалізації $s \in S$; $A_h^{m(s)}$ – робочий час супровідних активностей розробки на фазі $h \in H$ для ролі $m \in T_D$ та спеціалізації $s \in S$; G_h^m – робочий час загальних проєктних активностей ролі $m \in T_D$ на фазі $h \in H$; O_h^m – час відсутності ролі $m \in T_D$ на фазі $h \in H$; I_h^m – робочий час простою ролі $m \in T_D$ на фазі $h \in H$; R_h^m – резерв проєктного робочого часу ролі $m \in T_D$ на фазі $h \in H$; ϕ_h^m – ЕПЗ ролі $m \in T$ на фазі $h \in H$; $\alpha^{m(s)}$ – КПРК ролі $m \in T_D$ та спеціалізації $s \in S$; η_h^m – КПСЗ ролі $m \in T_D$ на фазі $h \in H$; $E^{(s)}$ – НОР змісту проєктних робіт для спеціалізації $s \in S$; L_h – тривалість фази реалізації проєкту $h \in H$. Параметр $\delta_h^{m(s)} \in \{0,1\}$ введено у систему (2.27) з метою збільшення гнучкості, отримавши можливість виключати або включати розробку на певній фазі $h \in H$ для ролі $m \in T_D$ та спеціалізації $s \in S$.

Припустимо, що робочий час, затрачений на супровідні активності розробки $A_h^{m(s)}$ прямопропорційно залежить від робочого часу розробки $D_h^{m(s)}$:

$$A_h^{m(s)} = d_h^{(s)} D_h^{m(s)},\ h \in H,\ m \in T_D,\ s \in S, \tag{2.28}$$

де $d_h^{(s)}$ називатимемо *коефіцієнтом супровідних активностей розробки*, КСАР (англ. «supplementary development activities coefficient», SDAC). Зауважимо, що $d_h^{(s)}$ залежить від спеціалізації s (іншими словам, для різних технологій програмування робочий час супровідних активностей розробки відрізняється) та фази реалізації проєкту h (наприклад, на початкових та завершальних фазах реалізації проєкту супровідні активності розробки можуть вимагати більше робочого часу у порівнянні із основною фазою розробки).

Аналогічно (2.28) зробимо припущення, що робочий час загальних проєктних активностей G_h^m прямопропорційно залежить від проєктного робочого часу

W_h^m, за виключенням часу відсутності O_h^m, коли інженер-розробник не бере участі у загальних проєктних активностях:

$$G_h^m = g_h\left(W_h^m - O_h^m\right), h \in H, m \in T_D, \tag{2.29}$$

де $0 \le g_h \le 1$ – коефіцієнт загальних проєктних активностей, КЗПА (англ. «general project activities coefficient», GPAC). Значення цього коефіцієнту залежить від фази реалізації проєкту: зазвичай, його значення вищі на початкових та завершальних етапах реалізації.

Варто підкреслити, що параметри $d_h^{(s)}$ та g_h, зокрема, визначають обсяг робочого часу, що затрачається на командну роботу. Це, в свою чергу, дає можливість закласти в системі рівнянь балансу робочого часу закономірності життєвого циклу проєктної команди [129].

Аналогічно (2.28) та (2.29) вважатимемо, що резерв проєктного робочого часу інженера-розробника R_h^m прямопропорційний проєктному робочому часу W_h^m:

$$R_h^m = r_h W_h^m, h \in H, m \in T_D, \tag{2.30}$$

де $0 \le r_h < 1$ – *коефіцієнт резерву проєктного робочого часу*, КРПРЧ (англ. «coefficient of project working time contingency», CPWTC). Рекомендується обирати більші значення цього коефіцієнту для фаз проєкту, де вища ймовірність та «серйозність» проєктних ризиків. Зазвичай, ймовірність «матеріалізації» ризиків є вищою на завершальних етапах реалізації проєкту ніж на початкових, тому й значення коефіцієнту r_h рекомендується обирати по зростаючій від початку до завершення реалізації проєкту.

З врахуванням (2.28) та (2.29) система (2.27) запишеться наступним чином:

$$\begin{cases} \left(1 - g_h - r_h\right) W_h^m = \sum\limits_{s \in S}\left[\delta_h^{m(s)}\left(1 + d_h^{(s)}\right) D_h^{m(s)}\right] + \left(1 - g_h\right) O_h^m + I_h^m, h \in H, m \in T_D, \\ W_h^m = \phi_h^m L_h, h \in H, m \in T, \\ E^{(s)} = \sum\limits_{h \in H} \sum\limits_{m \in T_D} \alpha^{m(s)} \eta_h^m D_h^{m(s)}, s \in S. \end{cases} \tag{2.31}$$

Система рівнянь балансу робочого часу для проміжного оцінювання (2.31), будучи логічним продовженням аналогічної системи для попереднього оцінювання (2.12), оперує більшою кількістю параметрів у порівнянні з (2.12). Зокрема, за рахунок цього результати проміжного оцінювання є більш деталізованими та забезпечують вищу точність.

2.8.3. Оцінювання нормалізованої спроможності розробки

Із першого та другого рівнянь системи балансу робочого часу (2.31), враховуючи (2.9), випливає наступний вираз для робочого часу розробки:

$$D_h^{m(s)} = \frac{\phi_h^{m(s)}}{\phi_h^m\left(1+d_h^{(s)}\right)}\left[\left(1-g_h-r_h\right)\phi_h^m L_h-\left(1-g_h\right)O_h^m-I_h^m\right], h\in H, m\in T_D. \quad (2.32)$$

Зауважимо, що (2.32) застосовна для випадку $\delta_h^{m(s)}=1$, а для $\delta_h^{m(s)}=0$ вважатимемо, що $D_h^{m(s)}=0$.

Тоді, відштовхуючись від визначення НСР (2.4), отримаємо:

$$\begin{aligned} C_h^{m(s)} &= \frac{\phi_h^{m(s)}\alpha^{m(s)}\eta_h^m}{\phi_h^m\left(1+d_h^{(s)}\right)}\times \\ &\times\left[\left(1-g_h-r_h\right)\phi_h^m L_h-\left(1-g_h\right)\min_{\mathbb{O}_h^m} O_h^m-\min_{\mathbb{I}_h^m} I_h^m\right], h\in H, m\in T_D, \end{aligned} \quad (2.33)$$

де $\mathbb{O}_h^m$ та $\mathbb{I}_h^m$ – відповідно множини всіх можливих значень часу відсутності та часу простою ролі $m\in T_D$ на фазі $h\in H$.

Для практичного застосування (2.33) можна дещо спростити, зробивши наступні припущення:

1) у команді немає інженерів-розробників із змішаними спеціалізаціями, тобто $\phi_h^m=\phi_h^{m(s)}$ і $\alpha^m=\alpha^{m(s)}$;

2) КПСЗ не залежить від фази реалізації проєкту, тобто $\eta_h^m=\eta^m$, тоді $\rho^m=\alpha^m\eta^m$;

3) мінімально можливе значення часу відсутності $\min\limits_{\mathbb{O}_h^m} O_h^m$ прямопропорційно залежить від проєктного робочого часу $W_h^m = \phi_h^m L_h$:

$$\min_{\mathbb{O}_h^m} O_h^m = o_h \phi_h^m L_h,\ h \in H,\ m \in T_D, \tag{2.34}$$

де $0 < o_h < 1$ – *коефіцієнт робочого часу відсутності*;

4) мінімальне значення часу простою дорівнює нулю: $\min\limits_{\mathbb{I}_h^m} I_h^m = 0$.

Враховуючи зроблені вище припущення 1-4, із (2.33) отримаємо вираз для НСР:

$$C_h^m = \frac{\rho^m \phi_h^m L_h}{1 + d_h^{(s)}} \left[\left(1 - g_h - r_h\right) - \left(1 - g_h\right) o_h \right],\ h \in H,\ m \in T_D. \tag{2.35}$$

Легко бачити, що НСР для фази проєкту $h \in H$ є вектором $C_h = \left(C_h^{(s_1)}, C_h^{(s_2)}, \ldots, C_h^{(s_q)} \right)$, кожен елемент якого відповідає спеціалізації, де $S = \left\{ s_1, s_2, \ldots, s_q \right\}$ – множина спеціалізацій, а q – кількість спеціалізацій.

2.8.4. Оцінювання тривалості проєкту

Як і для попереднього оцінювання, критерієм для визначення тривалості проєкту є достатність НСР для реалізації змісту проєктних робіт. Нехай НОР проєктних робіт становить $E = \left(E^{(s_1)}, E^{(s_2)}, \ldots, E^{(s_q)} \right)$, де $S = \left\{ s_1, s_2, \ldots, s_q \right\}$ – множина спеціалізацій, а q – кількість спеціалізацій, тоді для оцінки кількості спринтів, необхідних для реалізації проєкту, застосуємо критерії:

$$\begin{cases} k_{\min} \le k^*, \\ \forall s \in S, \left[C^{(s)} \right]^{k^*} \ge E^{(s)}, \\ \exists \tilde{s} \in S, \left[C^{(\tilde{s})} \right]^{k^*-1} < E^{(\tilde{s})}, \end{cases} \tag{2.36}$$

де $k_{\min}$ – мінімально допустима кількість спринтів, $\left[C^{(s)} \right]^k$ – НСР для спеціалізації $s \in S$ за умови, що тривалість проєкту становить k спринтів.

Зауважимо, що спринти відповідають рівню «2» в ієрархічній структурі графіку реалізації проєкту (див. підрозділ 2.5 та рис. 2.2), в той час як фази, якими ми оперували до цього часу, знаходяться на другому рівні. Використання спринтів є досить зручним, оскільки часто на практиці спринт (що, зазвичай, триває 2 тижні) є мінімальним кроком зміни тривалості проєкту.

2.8.5. Схема застосування проміжного оцінювання

Схема застосування проміжного оцінювання (рис. 2.6) схожа до аналогічної схеми для попереднього оцінювання (див. пункт 2.7.8 та рис. 2.5).

В якості вхідних даних проміжне оцінювання використовує:

1) інформацію про проєкт, функційні та нефункційні вимоги, архітектуру (з вищим ступенем деталізації у порівнянні з попереднім оцінюванням);

2) результати попереднього оцінювання (якщо таке проводилося): ICEO, оптимістичну та песимістичну HOP для елементів оцінювання, склад проєктної команди для оптимістичної та песимістичної оцінок, оптимістичну та песимістичну оцінки тривалості проєкту.

Схема застосування проміжного оцінювання включає наступні кроки:

Крок 1. Підвищити рівень деталізації ICEO, беручи за основу ICEO, отриману в результаті попереднього оцінювання. Якщо попереднє оцінювання не проводилося, то на цьому кроці потрібно побудувати ICEO.

Крок 2. Надати оптимістичну та песимістичну HOP для елементів оцінювання із ICEO, отриманої на кроці 1, застосовуючи 3-точкове оцінювання PERT [22], [23]. Зауважимо, що метод оцінювання трудоємності з розділу 2.7.5 не рекомендується для проміжного оцінювання, через вищий ступінь деталізації ICEO (а отже більшу кількість елементів оцінювання).

Крок 3. Визначити шаблон графіку реалізації проєкту та складу команди, що, зокрема, включає розбиття графіку реалізації проєкту на фази із визначенням варіативності тривалості цих фаз (нагадаємо, що тривалість фази визначається як кількість спринтів), а також визначення значень таких параметрів, як тривалість спринта, КСАР $d_h^{(s)}$, КЗПА g_h, КРПРЧ r_h та $\delta_h^{(s)}$ (див. пункт 2.8.2).

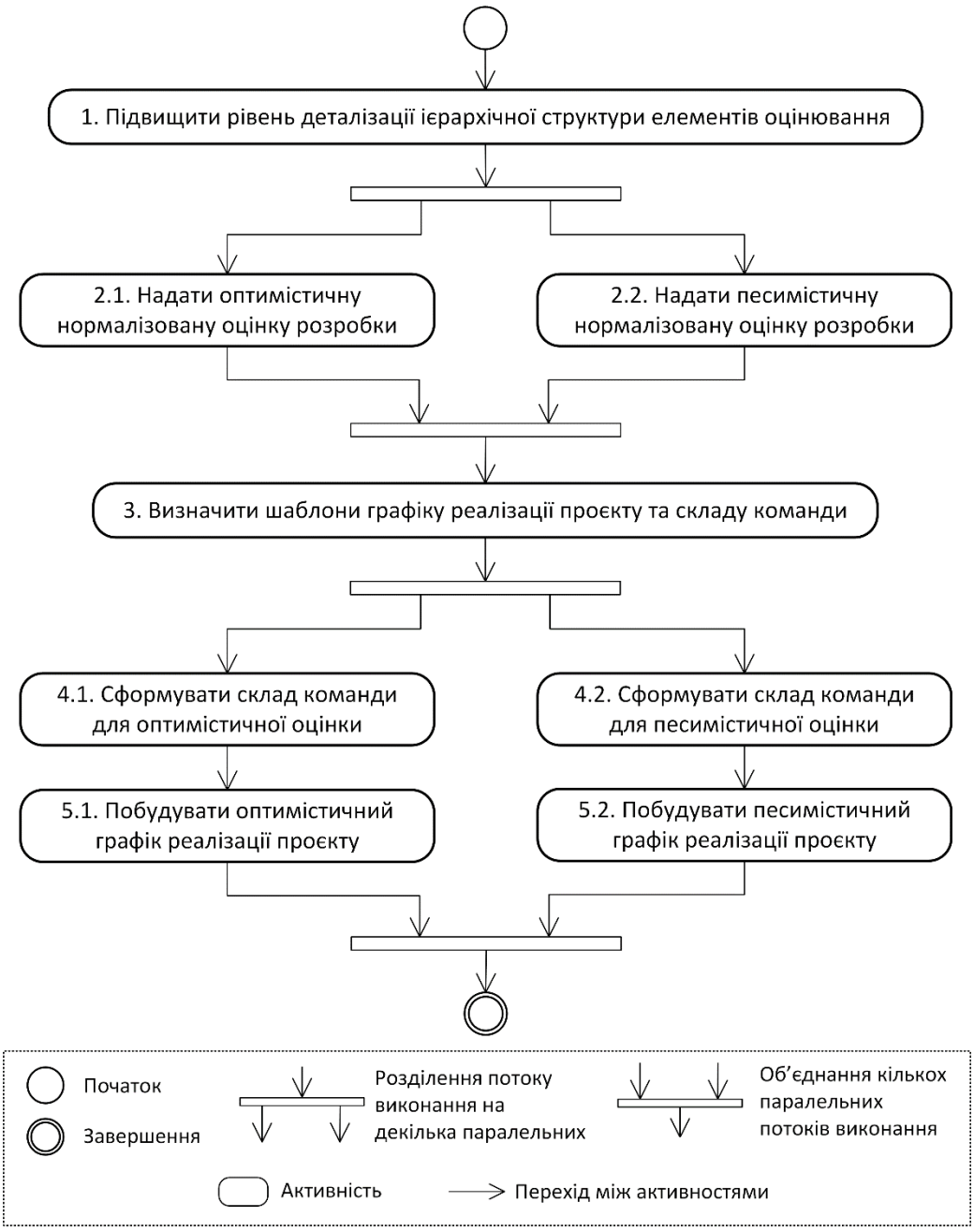


Рис. 2.6. UML-діаграма активностей схеми застосування проміжного оцінювання

Крок 4. Сформувати склад проєктної команди для оптимістичної та песимістичної оцінок, вказавши множину ролей учасників команди, спеціалізації інженерів-розробників, ЕПЗ для кожної із фаз проєкту.

Крок 5. Побудувати оптимістичний та песимістичний графіки реалізації проєкту, змінюючи тривалості фаз так, щоб виконувалася умова (2.36).

Результати застосування проміжного оцінювання:

1) ICEO (більш детальна ніж та, що отримана на етапі попереднього оцінювання);

2) оптимістична та песимістична НОР для елементів оцінювання, що належать отриманій ICEO;

3) склад проєктної команди для оптимістичної та песимістичної оцінок (включаючи ЕПЗ для кожної з фаз проєкту);

4) оптимістичний та песимістичний графіки реалізації проєкту.

Аналогічно попередньому оцінюванню, простота проміжного оцінювання робить можливою його реалізацію у Microsoft Excel. Як і у випадку попереднього оцінювання, результати застосування проміжного оцінювання не рекомендовані для взяття зобов'язань щодо реалізації проєкту – їх призначення полягає в інформуванні зацікавлених сторін щодо оцінки та передачі в якості вхідних даних для наступного етапу – детального оцінювання (див. підрозділ 2.9).

2.8.6. Застереження щодо застосування проміжного оцінювання

Як видно із (2.36), оцінка тривалості проєкту базуються на критерії достатності НСР для реалізації проєктних робіт, трудоємність яких виражена як НОР. Однак на практиці можливі ситуації, коли тривалість проєкту залежить не скільки він НСР та НОР, як від залежностей між елементами оцінювання та зовнішніх (по відношенню до проєкту) факторів. В таких сценаріях не рекомендується застосування методу проміжного оцінювання, зокрема, в тій його частині, що відповідає за визначення тривалості проєкту. Більш ефективними в таких випадках будуть CPM [25], PERT [22], [23] або детальне оцінювання (див. підрозділ 2.9).

2.9. Детальне оцінювання

2.9.1. Детальне оцінювання та його особливості

Основним завданням детального оцінювання (англ. «precise estimation») є отримання оцінки, точність якої дає змогу її використання для взяття зобов'язань щодо реалізації проєкту (наприклад, для підписання контракту з клієнтом). Детальне оцінювання відрізняється від попереднього етапу, проміжного оцінювання, наступним:

1) вищий рівень деталізації ICEO, з якої на початку реалізації можна сформувати «беклог» проєкту;

2) вищий рівень деталізації графіку реалізації проєкту: якщо на етапі проміжного оцінювання передбачається лише деталізація графіку до рівня «1», тобто фаз проєкту, то для детального оцінювання – до рівня «2», тобто спринтів (див. підрозділ 2.5 та рис. 2.2); можлива також прив'язка спринтів до календарних дат;

3) визначення залежностей елементів оцінювання та побудова розкладу реалізації елементів оцінювання на основі цих залежностей (див. підрозділ 2.10);

4) ідентифікація та аналіз припущень, ризиків, залежностей (не входить до завдань цієї дисертаційної роботи).

Для детального оцінювання будемо користуватися системою рівнянь балансу робочого часу (2.27) або (2.31), HCP будемо оцінювати, користуючись (2.33) або (2.35), тривалість проєкту оцінюватимемо, комбінуючи (2.36) та метод побудови розкладу реалізації елементів оцінювання (див. підрозділ 2.10). Іншими словами, для детального оцінювання пропонується використовувати інструменти проміжного оцінювання, за винятком методу побудови розкладу реалізації елементів оцінювання.

Порівняння детального оцінювання з іншими етапами оцінювання представлено у додатку 3.

2.9.2. Схема застосування детального оцінювання

Схема застосування детального оцінювання (рис. 2.7) аналогічна схемі застосуванням проміжного оцінювання (див. пункт 2.8.5 та рис. 2.6). Ключовою відмінністю між ними є те, що детальне оцінювання передбачає побудову розкладу реалізації елементів оцінювання (кроки 6.1 та 6.2 на рис. 2.7). У випадку, якщо не вдається побудувати допустимий розклад реалізації елементів оцінювання, відбувається збільшення тривалості проєкту на один спринт (кроки 6.1.1 та 6.2.1 на рис. 2.7) і побудова розкладу реалізації елементів оцінювання повторюється для графіку реалізації проєкту із збільшеною тривалістю. Зауважимо, що збільшення тривалості проєкту здійснюється згідно з правилами, визначеними обраним шаблоном графіку реалізації проєкту (див. підрозділ 2.5).

Для побудови розкладу реалізації елементів оцінювання можна скористатися класичними методами такими як діаграми Ганта [130], [131] чи мережні діаграми [132] із наступним визначенням критичного шляху [25] для уточнення тривалості проєкту. Однак, при використанні класичних методів можуть виникнути труднощі їх поєднання із підходом нормалізації, зокрема, балансуванням НСР проєктної команди та НОР елементів оцінювання. Для вирішення цієї проблеми розроблено метод побудови розкладу реалізації елементів оцінювання (див. підрозділ 2.10), комплементарний до методу поетапного оцінювання, що представлений в цьому розділі.

Результати детального оцінювання можуть використовуватися для:

1) інформування зацікавлених сторін (аналогічно як і результати попереднього та проміжного оцінювань);

2) взяття зобов’язань щодо реалізації проєкту (наприклад, під час підписання договору з клієнтом);

3) передачі команді реалізації проєкту, яка, в свою чергу, буде деталізуватиме оцінку до рівня, необхідного для безпосереднього виконання проєктних задач.

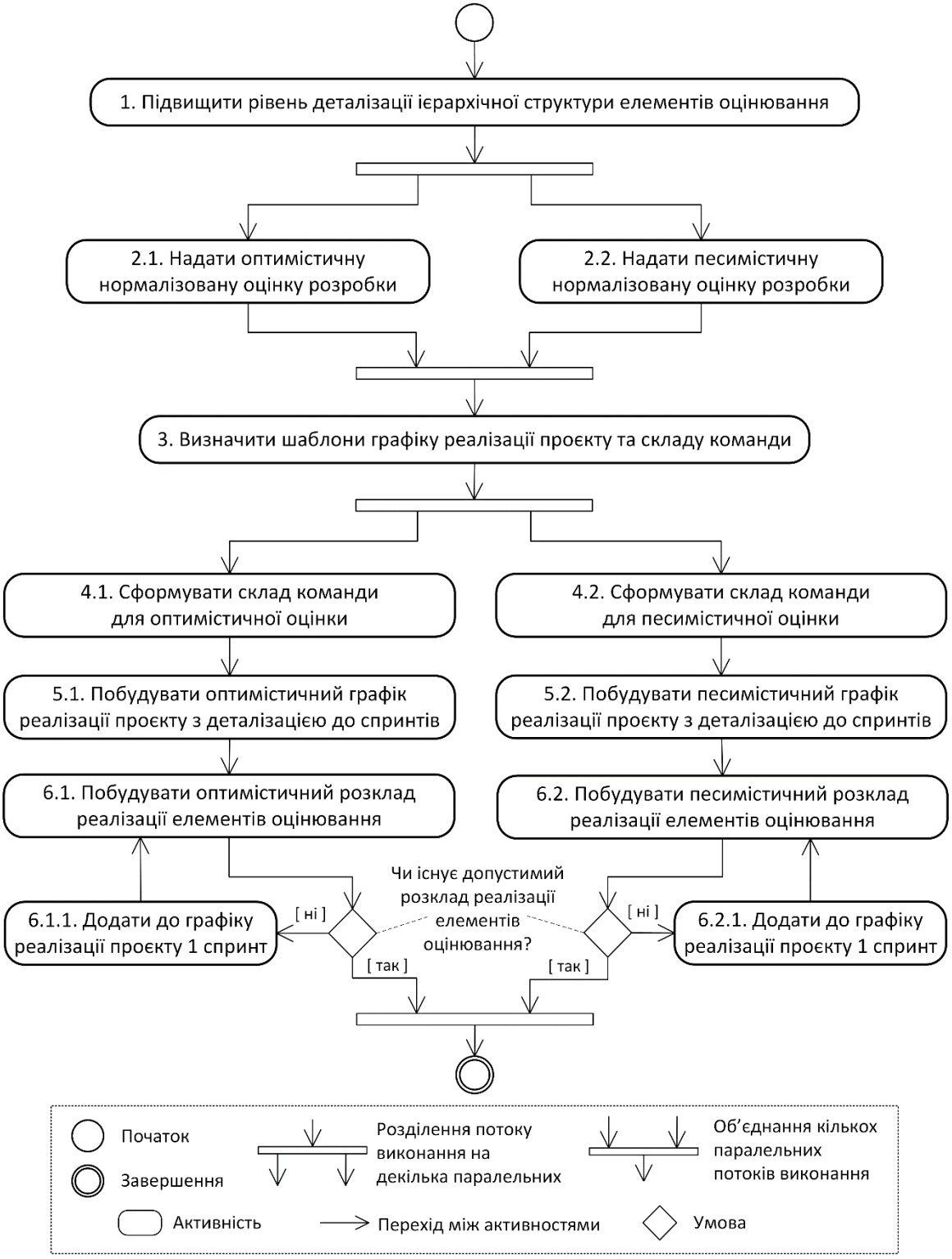


Рис. 2.7. UML-діаграма активностей схеми застосування детального оцінювання

2.9.3. Застереження щодо застосування детального оцінювання

Як можна бачити вище, тривалість проєкту у детальному оцінюванні визначається шляхом поєднання двох підходів: спершу тривалість визначається відповідно до НСР команди згідно (2.36), а тоді збільшується в разі неможливості

побудови допустимого розкладу реалізації елементів оцінювання. Однак, такий підхід працюватиме на практиці, якщо залежності елементів оцінювання не є надто складними. У випадку складних залежностей краще скористатися методами CPM [25] та PERT [22], [23]. Якщо ж ці залежності містять умовні переходи та враховують ризики, то побудову розкладу реалізації елементів оцінювання та оцінку тривалості проєкту можна виконати, застосувавши метод GERT [39].

2.10. Метод побудови розкладу реалізації елементів оцінювання

2.10.1. Задача побудови розкладу виконання робіт проєкту

В цьому підрозділі представлено метод побудови розкладу реалізації елементів оцінювання [3], що є подальшим розвитком методу List Scheduling [133], [134], [135]. Основна ідея методу полягає у розподілі проєктних задач на графіку реалізації проєкту так, щоб НОР елементів оцінювання не перевищувала НСР проєктної команди на кожній із фаз реалізації проєкту.

У проєктному менеджменті відповідна задача належить до дисципліни планування проєктів [132]. Класичними підходами до побудови розкладу реалізації проєктних задач є застосування діаграм Ганта [130], [131] або мережних діаграм [132]. Зокрема, мережні діаграми використовуються в таких відомих методах як PERT [22], [23] та CPM [25].

У математичній науці побудова розкладу реалізації задач проєкту вивчається в рамках дисципліни «математичні методи дослідження операцій», зокрема, у теорії розкладів. Задачу, метод розв’язання якої представлено в цьому підрозділі, відносять до класу задач побудови розкладу виконання робіт проєкту з врахуванням ресурсних обмежень (англ. «resource-constrained project scheduling problem», RCPSP) [136], [137], [138], [139]. В літературі виділяють наступні основні групи методів розв'язування задачі RCPSP: точні методи (сюди, зокрема, належать методи лінійного, цілочисельного, цільового та динамічного програмування), евристичні та мета-евристичні методи [136], [140], [141], [142]. В основі

ряду підходів до розв'язування задачі RCPSP лежить ідея методу List Scheduling (в україномовному варіанті трапляється також назва «метод диспетчеризації») [133], [134], [135], що базується на жадібному алгоритмі розстановки задач у розкладі реалізації згідно їх пріоритету.

Не зважаючи на велику кількість існуючих методів розв'язання задачі RCPSP, жоден з них не є універсальним, що, перш за все, зумовлено складністю та різноманіттям відповідних задач. Актуальність розробки методу побудови розкладу реалізації задач проєкту з розробки ПЗ продиктована, перш за все, вимогою його комплементарності до розроблених автором методу поетапного оцінювання (див. підрозділи 2.1-2.9) [4], [5] та методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту (див. підрозділ 2.11) [5], [16].

Варто зауважити, що у цьому дисертаційному дослідженні терміни «графік реалізації проєкту» та «розклад реалізації елементів оцінювання» пов'язані між собою, але мають різні значення. Під графіком реалізації проєкту розумітимемо часовий проміжок з ієрархічною структурою фаз, упродовж якого здійснюється виконання проєкту (див. підрозділ 2.5). В свою чергу, розклад реалізації елементів оцінювання визначає часові проміжки (на графіку реалізації проєкту), упродовж яких відбувається реалізація кожного з цих елементів оцінювання. Наприклад, у розкладі реалізації вказується, що виконання певного елементу оцінювання здійснюється упродовж другого, третього та п'ятого спринтів.

2.10.2. Залежності елементів оцінювання

Як правило, елементи оцінювання, які входять до складу певної ІСЕО, мають різного роду залежності, що впливають на їх місце у графіку реалізації проєкту. Такі залежності розділимо на дві категорії: залежності від графіку реалізації проєкту та залежності між елементами оцінювання.

Нехай маємо елемент оцінювання $x \in X$, де X – ієрархічна структура елементів оцінювання; $x[\text{start}] \in H$ та $x[\text{end}] \in H$ планові час початку та завершення реалізації x. Для визначеності вважатимемо, що H – множина спринтів (тобто рівень «2» ієрархічної структури графіку реалізації проєкту, див.

підрозділ 2.5). Очевидно, що $x[\text{start}] \le x[\text{end}]$. Тоді можливі наступні залежності часу реалізації елементу оцінювання від графіку реалізації проєкту:

$$x[\text{start}] \in \Delta_{[x]}, \tag{2.37}$$

$$x[\text{end}] \in \Delta^{[x]}, \tag{2.38}$$

$$x[\text{end}] - x[\text{start}] \in \Delta^{[x]}_{[x]}. \tag{2.39}$$

Зауважимо, що використання операції належності множині у (2.37)-(2.39) дає змогу визначати як точне значення різниці виразів у лівих частинах, так і діапазон таких значень. Множина Δ в переважній більшості випадків є підмножиною цілих чисел – $\Delta \subset \mathbb{Z}$ (наприклад, коли час початку та завершення є порядковими номерами спринтів). Хоча можна допустити ситуації, коли Δ є підмножиною раціональних чи навіть дійсних чисел. Наприклад, якщо необхідно, щоб реалізація x розпочалася у спринті 1, то (2.37) запишеться як $x[\text{start}] = 1$. У випадку, якщо тривалість реалізації x не повинна перевищувати 4 спринти, то (2.39) матиме вигляд $x[\text{end}] - x[\text{start}] \le 3$. На практиці залежності від графіку реалізації проєкту можуть бути продиктовані зовнішніми (відносно проєкту) залежностями, наприклад, необхідністю очікування певних результатів від паралельного проєкту.

Залежності другої категорії визначають співвідношення між часом початку та завершення різних елементів оцінювання. Нехай маємо два елементи оцінювання $x_1 \in X$ та $x_2 \in X$, де X – ICEO; $x_1[\text{start}]$, $x_2[\text{start}]$ – планований час початку реалізації цих елементів, а $x_1[\text{end}]$, $x_2[\text{end}]$ – плановий час завершення їх реалізації. Тоді мають місце наступні залежності:

$$x_1[\text{start}] - x_2[\text{start}] \in \Delta_{[x_1, x_2]}, \tag{2.40}$$

$$x_1[\text{start}] - x_2[\text{end}] \in \Delta^{[x_2]}_{[x_1]}, \tag{2.41}$$

$$x_1[\text{end}] - x_2[\text{start}] \in \Delta^{[x_1]}_{[x_2]}, \tag{2.42}$$

$$x_1[\text{end}] - x_2[\text{end}] \in \Delta^{[x_1, x_2]}. \tag{2.43}$$

Наприклад, якщо реалізація x_2 повинна розпочатися не пізніше, як через 2 спринти після завершення реалізації задачі x_1, то вираз (2.42) запишеться наступним чином: $x_1[\text{end}] - x_2[\text{start}] \in \{-2,-1,0\}$. У складніших випадках можливе одночасне поєднання кількох типів залежностей, наприклад, реалізації x_1 та x_2 повинні розпочатися та завершитися одночасно: $x_1[\text{start}] - x_2[\text{start}] = 0$ і $x_1[\text{end}] - x_2[\text{end}] = 0$.

На практиці найбільш частою є залежність, що є частковим випадком (2.42):

$$x_1[\text{end}] - x_2[\text{start}] < 0, \tag{2.44}$$

тобто реалізація x_2 повинна розпочатися після завершення x_1 (так звана залежність типу «завершення-початок»).

2.10.3. Тривалість реалізації елементу оцінювання

Одним із ключових завдань, що потрібно вирішити під час побудови розкладу реалізації елементів оцінювання, є визначення тривалості реалізації кожного з елементів оцінювання, тобто значень $x[\text{start}]$ та $x[\text{end}]$, де $x \in X$, X – ICEO. Зауважимо, що HOP елементу оцінювання (тобто $x[\text{NDE}]$) має суттєвий вплив на тривалість його реалізації, але, очевидно, не є тотожною цій тривалості.

На те, скільки триватиме реалізація елементу оцінювання, впливає ряд факторів, основні з яких наступні:

1) HOP елементу оцінювання;

2) залежності елементу оцінювання (2.37)-(2.43);

3) HCP команди (у часовому проміжку реалізації елементу оцінювання);

4) життєвий цикл елементу оцінювання, що, зокрема, визначає розподіл робочого часу, який затрачається на супровідні активності розробки (див. підрозділ 2.3).

2.10.4. Розклад реалізації елементів оцінювання

Очевидно, що значення $x[\text{start}]$ та $x[\text{end}]$, для кожного $x \in X$ не є достатніми для побудови розкладу реалізації – крім них, необхідно вказати, скільки часу на реалізацію затрачатиметься на кожному із спринтів. *Розкладом реалізації елементів оцінювання* (англ. «estimable items implementation schedule») називатимемо матрицю:

$$J = \begin{pmatrix} M_{h_1}^{[x_1]} & M_{h_2}^{[x_1]} & \cdots & M_{h_k}^{[x_1]} \\ M_{h_1}^{[x_2]} & M_{h_2}^{[x_2]} & \cdots & M_{h_k}^{[x_2]} \\ \cdots & \cdots & \ddots & \cdots \\ M_{h_1}^{[x_m]} & M_{h_2}^{[x_m]} & \cdots & M_{h_k}^{[x_m]} \end{pmatrix}, \tag{2.45}$$

де $X = \{x_1, x_2, \ldots, x_m\}$ – ІСЕО, $H = \{h_1, h_2, \ldots, h_k\}$ – графік реалізації проєкту, заданий множина спринтів, $M_h^{[x]}$ – повний робочий час розробки, затрачений інженерами-розробниками на реалізацію елементу оцінювання $x \in X$ упродовж спринта $h \in H$.

Оскільки трудоємність елементів оцінювання та спроможність розробки проєктної команди оцінюються згідно з підходом нормалізації (див. підрозділ 2.4), то для побудови розкладу (2.45) зручно спершу отримати так званий нормалізований розклад реалізації. Нехай маємо множину простих елементів оцінювання $x \in X$, для кожного з яких визначено НОР $x[\text{NDE}]$. Також маємо графік проєкту, виражений як множина спринтів H; при цьому для кожного із спринтів $h \in H$ відома НСР $C_h \geq 0$. Тоді *нормалізованим розкладом реалізації елементів оцінювання* (англ. «normalized estimable items implementation schedule») [3] називатимемо матрицю:

$$J_N = \begin{pmatrix} U_{h_1}^{[x_1]} & U_{h_2}^{[x_1]} & \cdots & U_{h_k}^{[x_1]} \\ U_{h_1}^{[x_2]} & U_{h_2}^{[x_2]} & \cdots & U_{h_k}^{[x_2]} \\ \cdots & \cdots & \ddots & \cdots \\ U_{h_1}^{[x_m]} & U_{h_2}^{[x_m]} & \cdots & U_{h_k}^{[x_m]} \end{pmatrix}, \tag{2.46}$$

де $U_h^{[x]} \geq 0$ – НРЧР, затрачений на реалізацію елементу оцінювання $x \in X$ на спринті $h \in H$. При цьому справедливі наступні умови балансу для рядків:

$$\sum_{h \in H} U_h^{[x]} = x\left[\text{NDE}\right], x \in X, \tag{2.47}$$

а також стовпців матриці:

$$\sum_{x \in X} U_h^{[x]} \leq C_h, h \in H. \tag{2.48}$$

Для кожного $x \in X$ має місце є наступна умова, що лімітує мінімальний та максимальний обсяги НРЧР, що можуть бути затрачені на реалізацію елементу оцінювання упродовж спринта:

$$\left[U^{[x]}\right]_{\min} \leq U_h^{[x]} \leq \left[U^{[x]}\right]^{\max}, x \in X, h \in H. \tag{2.49}$$

Важливою умовою є те, що реалізація елементу оцінювання не може перериватися.

Зауважимо, що вирази (2.46)-(2.49) застосовні до команди інженерів-розробників без диференціації спеціалізацій (див. підрозділ 2.6). Для команди із диференційованими чи змішаними спеціалізаціями $x\left[\text{NDE}\right]$, C_h, $U_h^{[x]}$ будуть векторами, елементи яких відповідають спеціалізаціям інженерів-розробників.

Легко бачити, що нормалізований розклад реалізації елементів оцінювання передбачає наступні спрощення:

1) враховується лише НРЧР $U_h^{[x]}$ і при цьому не враховується робочий час, що затрачається на супровідні активності розробки (див. підрозділ 2.3);

2) ІСЕО X є множиною лише простих елементів оцінювання (тобто не враховується ієрархічність ІСЕО);

3) не враховується ієрархічність графіку реалізації проєкту.

Варто підкреслити, що нормалізований розклад реалізації елементів оцінювання (2.46) слугує своєрідним «скелетом», на основі якого будується так званий «повний» розклад (2.45), який, зокрема, враховує супровідні активності розробки. Розробка методу побудови «повного» розкладу реалізації елементів оцінювання не входить до завдань цього дисертаційного дослідження.

2.10.5. Побудова нормалізованого розкладу реалізації елементів оцінювання

Для побудови нормалізованого розкладу реалізації елементів оцінювання необхідні наступні вхідні дані:

1) команда інженерів-розробників без диференціації спеціалізацій;

2) графік реалізації проєкту, виражений як множина спринтів H з можливістю додавання нового спринта у випадку необхідності збільшення тривалості проєкту (згідно з правилами, визначеними обраним шаблоном графіку реалізації проєкту, див. підрозділ 2.5);

3) для кожного із спринтів $h \in H$ відома НСР C_h;

4) ІСЕО X є множиною простих елементів оцінювання, для кожного з яких визначено НОР $x[\mathrm{NDE}]$;

5) сумарна НСР команди $C = \sum_{h \in H} C_h$ достатня для реалізації всіх елементів оцінювання $x \in X$:

$$\sum_{h \in H} C_h \geq \sum_{x \in X} x[\mathrm{NDE}]; \qquad (2.50)$$

6) між елементами оцінювання $x \in X$ допускаються залежності типу «завершення-початок» (2.44), при цьому вважається, що відповідний граф залежностей не містить циклів.

Завдання методу полягає у побудові нормалізованого розкладу реалізації елементів оцінювання (2.46) так, щоб виконувалися умови (2.47)-(2.49) і при цьому забезпечувалася нерозривність реалізації кожного з елементів оцінювання $x \in X$. Такий розклад називатимемо допустимим. Очевидно, що можливе існування одного і більше допустимих розкладів. У випадку, якщо допустимого розкладу не існує, необхідно збільшити кількість спринтів у графіку реалізації проєкту H (що, очевидно, гарантує існування допустимого розкладу).

У випадку існування більше як одного допустимого розкладу реалізації елементів оцінювання необхідно визначити критерій їх порівняння. В якості такого критерію візьмемо функцію штрафу від нормалізованого часу простою команди інженерів-розробників:

$$\Theta = \sum_{h \in H} \left[\theta_h \left(C_h - \sum_{x \in X} U_h^{[x]} \right) \right] \to \min, \qquad (2.51)$$

де $\theta_h \geq 0$ – коефіцієнт штрафу, який залежить від фази реалізації проєкту $h \in H$. Для прикладу, коефіцієнт штрафу θ_h дає змогу визначити, що простій інженерів-розробників (спричинений залежностями елементів оцінювання) є більш критичним на початкових фазах реалізації проєкту, ніж на завершальних.

Для побудови нормалізованого розкладу реалізації елементів оцінювання необхідно виконати наступні кроки, що базуються на методі List Scheduling [133], [134], [135]:

Крок 1. Побудувати перестановку (англ. «permutation») елементів оцінювання S (куди виходять всі елементи з ICEO X), відштовхуючись від залежностей (2.44). Для прикладу, першим елементом такої перестановки є елемент оцінювання, який не має залежностей від інших елементів оцінювання. Легко бачити, що в загальному випадку можливо побудувати досить велику кількість варіантів такої перестановки: у випадку відсутності залежностей між елементами оцінювання кількість варіантів такої перестановки дорівнює $n!$, де n – кількість елементів оцінювання. Для зручності викладок вважатимемо, що перестановка S «працює» за принципом стеку, тобто при виборі першого елементу цей елемент видаляється із S.

Крок 2. Вибрати перший елемент оцінювання із S та задати розподіл його НРЧР за спринтами, дотримуючись виконання обмежень (2.47)-(2.49), намагаючись при цьому розмістити реалізацію якомога раніше, а також враховуючи те, що реалізація не повинна перериватися.

Крок 3. Якщо розміщення елементу оцінювання не можливе (через недостатність НСР команди), необхідно додати ще один спринт згідно правил, визначених шаблоном графіку реалізації проєкту. Після цього повторити крок 2.

Крок 4. Завершити виконання, якщо у множині S не залишилося більше елементів оцінювання.

Очевидно, що запропонований вище підхід завжди забезпечує побудову допустимого розкладу реалізації елементів оцінювання. Кількість варіантів такого розкладу визначається кількістю можливих значень перестановки S, що генерується на кроці 1. Порівняння отриманих варіантів розкладу здійснюватимемо згідно (2.51).

2.10.6. Застосування методу побудови розкладу реалізації елементів оцінювання

Згідно з концепцією конусу невизначеності [28], [122], метод поетапного оцінювання проєктів з розробки ПЗ (див. підрозділи 2.1-2.9) передбачає послідовну підготовку трьох типів оцінок (попередньої, проміжної та детальної), кожна наступна з яких є детальнішою та точнішою за попередню. В контексті методу поетапного оцінювання розроблений метод побудови розкладу реалізації елементів оцінювання доцільно застосовувати на третьому етапі, детальному оцінюванні, коли рівень деталізації ICEO є достатнім. На етапі реалізації проєкту побудований розклад буде взятий за основу для розподілу проєктних задач за спринтами.

Розроблений метод побудови розкладу реалізації елементів оцінювання не рекомендується до застосування, коли:

1) реалізація проєкту значно залежить від зовнішніх факторів;

2) існують ризики, що істотно впливають на перебіг реалізації проєкту;

3) між елементами оцінювання існують складні залежності;

4) реалізація проєкту організована без використання спринтів (тобто не використовується scrum).

В цих ситуаціях рекомендується розглянути можливість використання, наприклад, діаграм Ганта [130], [131], [132], методів PERT [22], [23] та CPM [25], а у складніших випадках – методу GERT [39].

У найпростішому варіанті розроблений метод побудови розкладу реалізації елементів оцінювання може бути реалізований у табличному редакторі. Можливість застосування табличного редактора зумовлено, зокрема, тим, що для

проєктної команди не передбачена диференціація спеціалізацій. У випадку команди із диференційованими чи змішаними спеціалізаціями, коли значення НОР та НСР є векторами, розмірність яких відповідає кількості спеціалізацій інженерів-розробників, використання табличного редактора істотно ускладнюється – в цьому випадку ефективнішою буде реалізація із застосуванням однієї із мов програмування (наприклад, Python, C#, Java тощо).

Окремо варто звернути увагу на те, Microsoft Project, не зважаючи на його широкі можливості з підготовки розкладів реалізації проєктів, досить складно застосувати для програмної реалізації розробленого методу. Ця складність зумовлена, головним чином, відсутністю вбудованих функцій для контролю виконання умов балансу НСР та НОР (2.47)-(2.49), а також недостатньою гнучкістю при визначенні тривалості задачі залежно від призначених для її реалізації ресурсів.

2.10.7. Перспективи розвитку методу побудови розкладу реалізації елементів оцінювання

Вбачаються наступні кроки з розвитку розробленого методу побудови розкладу реалізації елементів оцінювання:

1) конкретизувати підхід до визначення тривалості реалізації елементу оцінювання та розподілу його НРЧР між спринтами (крок 2 методу);

2) додати підтримку команд із диференційованими та змішаними спеціалізаціями;

3) розширити метод можливістю переходу від нормалізованого розкладу реалізації до «повного», враховуючи життєвий цикл елементів оцінювання та структуру робочого часу інженерів-розробників;

4) дослідити можливість визначення критичного шляху (в тому сенсі, як його розуміють в контексті методу CPM [25]);

5) дослідити можливість додавання інших типів залежностей елементів оцінювання, визначених (2.37)–(2.43);

6) дослідити можливість застосування методу гілок та меж [143], який потенційно може оптимізувати розроблений метод, «відсікаючи» «безперспективні» варіанти розкладів реалізації. Зауважимо, що в загальному випадку кількість цих варіантів може бути досить значною: для ІСЕО з n елементів оцінювання у випадку відсутності залежностей кількість можливих значень множини S, що формується на кроці 1 методу, дорівнює числу перестановок з n, тобто становить $n!$.

Підкреслимо, що заявлені вище перспективи розвитку розробленого методу не входять до завдань цієї дисертаційної роботи.

2.11. Метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту

Як можна бачити вище, оцінювання оперує такими основними складовими як зміст проєктних робіт (представленим у вигляді ІСЕО разом із НОР), склад проєктної команди та графік реалізації проєкту (який, зокрема, визначає тривалість реалізації як кожної із фаз, так і проєкту в цілому). Припустивши, що зміст проєктних робіт є фіксованим (а отже, сталою є і відповідна НОР), дві інші складові є взаємозалежними: зміни в складі команди впливають на графік реалізації проєкту та навпаки – різні графіки реалізації проєкту потребують команд різного складу. Розроблений метод підтримки прийняття рішень має за мету надавати «найкращі» варіанти комбінацій складу команди та графіку реалізації проєкту [5], [16]. Метод базується на розв'язанні послідовності задач цілочисельного програмування із наступним застосуванням методу аналізу ієрархій [144] для ранжування отриманих альтернатив.

2.11.1. Визначення цільових функцій

Критерії, за яким здійснюватимемо порівняння варіантів комбінації складу команди та графіку реалізації проєкту, визначимо як набір цільових функцій, кожна з яких представлятиме окремий критерій для порівняння.

Однією із ключових характеристик оцінки є собівартість проєкту. Зробивши припущення, що за мету ставиться мінімізація собівартості реалізації проєкту, визначимо наступну цільову функцію:

$$F = \sum_{h \in H} \sum_{m \in T} r^m W_h^m \to \min, \tag{2.52}$$

де F – нормалізована собівартість реалізації проєкту; T – множина ролей проєктної команди; $m \in T$ – роль учасника проєктної команди; H – множина фаз графіку реалізації проєкту; $h \in H$ – фаза реалізації проєкту; $r^m \in [0,1]$ – нормалізована собівартість[1] однієї людино-години для ролі проєктної команди $m \in T$.

Крім собівартості, важливими характеристиками оцінки також є сумарна тривалість проєкту, розмір проєктної команди та нормалізований час простою. Для мінімізації цих характеристик застосуємо наступні цільові функції:

$$L = \sum_{h \in H} L_h \to \min, \tag{2.53}$$

$$\Phi = \frac{\sum_{h \in H} \left(L_h \sum_{m \in T} \phi_h^m \right)}{L} \to \min, \tag{2.54}$$

$$\mathrm{I}_N = \left\| \sum_{h \in H} C_h - E \right\| \to \min, \tag{2.55}$$

де L – тривалість реалізації проєкту; L_h – тривалість фази $h \in H$; ; ϕ^m – ЕПЗ ролі $m \in T$; Φ – середнє зважене значення ЕПЗ учасників команди упродовж реалізації проєкту (вважаємо, що на кожній із фаз $h \in H$ склад проєктної команди може змінюватися); $C_h = \left(C_h^{(s_1)}, C_h^{(s_2)}, \ldots, C_h^{(s_q)} \right)$ – НСР проєктної команди на фазі $h \in H$; $E = \left(E^{(s_1)}, E^{(s_2)}, \ldots, E^{(s_q)} \right)$ – сумарна НОР всіх елементів оцінювання відповідно до спеціалізації інженерів-розробників; кожен елемент векторів C_h та E відповідає спеціалізації інженерів-розробників, де $S = \{ s_1, s_2, \ldots, s_q \}$ – множина

[1] Застосування нормалізації до змінних, пов'язаних із грошовими сумами, надає такі переваги: (1) уникнення ризику розкриття комерційної таємниці; (2) уникнення великих чисел як значень цільової функції під час розв'язання задач оптимізації; (3) незалежність від конкретної грошової одиниці.

спеціалізацій, а q – кількість спеціалізацій; I_N – норма різниці векторів C_h та E (для визначеності вважатимемо, що застосовується евклідова норма).

2.11.2. Формулювання обмежень

В цьому пункті визначимо правила, що накладаються на змінні, якими оперують під час оцінювання. Математичним виразом цих правил будуть лінійні обмеження, накладені на змінні відповідних задач оптимізації.

Однією із ключових умов, що накладаються на оцінку, є те, що проєктна команда повинна мати досить спроможності, щоб реалізувати заданий зміст проєктних робіт, іншими словами, НСР команди C не повинна бути меншою за НОР E:

$$C = \sum_{h \in H} C_h \geq E, \tag{2.56}$$

де C_h – НСР проєктної команди на фазі $h \in H$.

Нехай H – множина фаз реалізації проєкту, при цьому кожна з цих фаз має певну тривалість L_h, $h \in H$. Тоді обмеження щодо тривалостей фаз проєкту представимо у вигляді:

$$\left[L_h\right]_{\min} \leq L_h \leq \left[L_h\right]^{\max}, h \in H. \tag{2.57}$$

На практиці, для прикладу, початкова та завершальна фази реалізації проєкту, зазвичай, мають фіксовану тривалість, в той час як тривалість фази активної розробки може змінюватися в залежності складу команди та обсягу проєктних робіт.

Наступна група обмежень стосується ЕЗП для підмножини проєктних ролей на фазі $h \in H$:

$$\left[\phi_h^{T^*}\right]_{\min} \leq \sum_{m \in T^*} \phi_h^m \leq \left[\phi_h^{T^*}\right]^{\max}, \tag{2.58}$$

де $T^* \subseteq T$ – підкоманда у проєктній команді T. Наприклад, умова, що в команді повинен бути хоча б один старший інженер-розробник, запишеться як наступне обмеження: $1 \leq \sum_{m \in T^{\text{senior}}} \phi^m$, де $T^{\text{senior}} \subseteq T$ – підмножина старших інженерів-розроб-

ників у команді T. Іншим прикладом обмеження цього типу може слугувати умова, що розмір всієї команди не повинен бути меншим за 5 та перевищувати 25 ЕЗП: $5 \le \sum_{m \in T} \phi^m \le 25$.

Для накладання умов на співвідношення між ЕПЗ різних підмножин ролей проєктної команди на одній або різних фазах реалізації проєкту скористаємося обмеженням наступного виду:

$$\gamma_1 \sum_{m_1 \in T_1} \phi_{h_1}^{m_1} \le \gamma_2 \sum_{m_2 \in T_2} \phi_{h_2}^{m_2}, \qquad (2.59)$$

де $T_1 \subseteq T$ і $T_2 \subseteq T$ – підкоманди у проєктній команді T, $\gamma_1 > 0$ і $\gamma_2 > 0$ – константи, $h_1, h_2 \in H$ – фази реалізації проєкту (при цьому допустимим є випадок, коли $h_1 = h_2$). Наприклад, із використанням обмеження цього типу може бути виражена умова, що один проєктний менеджер не може мати у підпорядкуванні більше як 15 учасників команди: $\sum_{m_1 \in T \setminus T^{\text{mgt}}} \phi^{m_1} \le 15 \sum_{m_2 \in T^{\text{mgt}}} \phi^{m_2}$, $T^{\text{mgt}} \subseteq T$ – підкоманда проєктних менеджерів. Також (2.59) можна застосувати для накладання обмежень на зміни складу команди при переході між фазами проєкту, наприклад, при переході від фази підготовки до фази активної розробки розмір команди інженерів-розробників не може зрости більше як у 1.5 рази: $\sum_{m_1 \in T_{D,1}} \phi_1^{m_1} \le \frac{2}{3} \sum_{m_2 \in T_{D,2}} \phi_2^{m2}$.

Очікується, що кількість обмежень у конкретній задачі оптимізації варіюватиме в межах кількох десятків чи сотень. Ця кількість залежить від шаблонів складу команди та графіку реалізації проєкту, а також від етапу оцінювання: для попереднього оцінювання ця кількість буде меншою, в той час як для проміжного та детального оцінювань – більшою.

2.11.3. Задачі цілочисельного програмування та рекомендації щодо їх розв'язання

ЕПЗ ролей проєктної команди є невід'ємним дійсним числом. Однак, очевидно, що далеко не всі дробові значення ЕПЗ мають практичний зміст.

Наприклад, немає сенсу залучати до проєкту інженера-розробника з ЕПЗ 0.67. Важливо підкреслити, що те, наскільки можна «дробити» ЕПЗ, залежить від проєктної ролі: якщо для старшого інженера-розробника мають сенс такі дробові значення ЕПЗ як 0.25 чи 0.5, то для молодшого інженера-розробника можливе лише повне залучення, тобто ЕПЗ, що дорівнює 1. Для математичного вираження цього замінимо дійсні змінні $\phi_h^m \geq 0$ на відповідні цілочисельні змінні f_h^m:

$$\phi_h^m = \mu^m f_h^m, m \in T, h \in H, \tag{2.60}$$

де $f_h^m \in \mathbb{N}_0 = \{0,1,2,\ldots\}$ – невід'ємне ціле число мінімально можливих ЕЗП ролі $m \in T$ на фазі $h \in H$, $\mu^m > 0$ – мінімальний крок зміни ЕЗП ролі $m \in T$.

Враховуючи обмеження (2.57) сформуємо множину із $p \in \mathbb{N}$ можливих графіків реалізації проєкту:

$$\mathrm{H} = \left\{ H_k \,\middle|\, k = \overline{1,p} \right\}. \tag{2.61}$$

Припустимо, що обмеження (2.57) дають змогу сформувати множину H із відносно невеликою кількістю елементів (що важливо з точки наступного розв'язання задач цілочисельного програмування). Зауважимо, що на практиці тривалість фази проєкту $h \in H$ зручно визначати як кількість спринтів, завдяки чому кількість елементів множини H буде відносно невеликою, в межах кількох десятків.

Як результат, для кожного із графіків реалізації проєкту $H_k, k = \overline{1,p}$ отримаємо задачу цілочисельного програмування із цільовою функцією (2.52), обмеженнями (2.56)-(2.59) та змінними (2.60). Розв'язавши ці задачі цілочисельного програмування, одержимо множину із p альтернатив:

$$\mathrm{A} = \left\{ \left(H_k, T_k, F_k, \Phi_k, \mathrm{I}_k \right) \,\middle|\, k = \overline{1,p} \right\} \tag{2.62}$$

при цьому, якщо якась із задач оптимізації не має розв'язку, то відповідна альтернатива не буде сформована.

Для розв'язання задач цілочисельного програмування (2.52), (2.56)-(2.59) можна застосувати, наприклад, такі відомі методи як Гоморі чи гілок та меж. При

цьому важливо підкреслити, що кількість змінних у одній із таких задач може коливатися від кількох десятків до кількох сотень (залежно від кількості фаз реалізації проєкту та кількості ролей у проєктній команді). Очевидно, що за таких умов «ручне» розв'язання цих задач оптимізації є надто громіздким, а отже, позбавлене практичного сенсу.

Тому для знаходження оптимальних розв'язків послідовності задач (2.52), (2.56)-(2.59) на практиці рекомендується скористатися однією із існуючих програмних реалізацій, наприклад: комерційною бібліотекою FICO Xpress (https://www.fico.com/, дата звернення: 09.08.2024) або бібліотекою з відкритим вихідним кодом COIN-R CBC (https://www.coin-or.org/Cbc/, дата звернення: 09.08.2024). Кожна з цих бібліотек має реалізацію на мові Python. Зокрема, Python-бібліотека Pyomo (https://www.pyomo.org/, дата звернення: 09.08.2024) [145], [146], що надає зручний інтуїтивний синтаксис для формулювання постановок задач математичного програмування, підтримує обидві ці бібліотеки. У праці [5] продемонстровано застосування розробленого методу підтримки прийняття рішень до попереднього оцінювання, використовуючи для розв'язання задач цілочисельного програмування поєднання бібліотек Pyomo та FICO Xpress.

2.11.4. Ранжування альтернатив

На практиці множина A із (2.62) може включати кілька десятків елементів, що може дещо ускладнити вибір «найкращої» альтернативи під час оцінювання. З метою спрощення прийняття рішення щодо «найкращої» альтернативи впорядкуємо елементи множини A, застосувавши метод аналізу ієрархій [144]. В якості критеріїв порівняння при цьому можна, для прикладу, взяти значення цільових функцій (2.52)-(2.55).

2.11.5. Застосування методу підтримки прийняття рішень на різних етапах оцінювання

Розроблений метод підтримки прийняття рішень застосовний на етапах попереднього, проміжного та детального оцінювань. Головна відмінність на кожному з цих етапів полягає у ступені деталізації, що впливає на кількість змін-

них та обмежень у відповідних задачах оптимізації. Так, на етапі попереднього оцінювання графік реалізації проєкту не розділяється на фази, а у складі команди, як правило, не розрізняються спеціалізації [5]. Ступінь деталізації для проміжного та детального оцінювань вищий за рахунок розбиття графіку реалізації проєкту на фази та визначення спеціалізацій учасників команди [16].

Висновки до розділу 2

Розроблений метод поетапного оцінювання, перш за все, спрямований на вирішення таких взаємопов'язаних проблем як «одноразові» оцінки, які надаються безпосередньо перед початком реалізації, та застосування оцінювання на всіх етапах життєвого циклу проєктів з розробки ПЗ (див. підрозділ 1.6). Не зважаючи на те, що поточна версія методу включає лише попереднє, проміжне та детальне оцінювання, які проводяться під час аналізу та проєктування (тобто до початку реалізації), потенціал, закладений теоретичними основами методу, дає змогу в майбутньому охопити й інші етапи життєвого циклу, зокрема, реалізацію проєкту. Метод поетапного оцінювання органічно доповнюється методом підтримки прийняття рішень, що має за мету підвищення ефективності роботи експертів, пропонуючи комбінації складу команди та графіку реалізації проєкту, отримані як результат розв'язання задач цілочисельного програмування. Ще одним методом, розробленим в рамках цієї дисертаційної роботи, є метод автоматизації побудови розкладу реалізації елементів оцінювання, який враховує НОР елементів оцінювання, НСР команди, а також залежності між елементами оцінювання. Цей метод рекомендований до застосування на етапі детального оцінювання. Важливо підкреслити, що кожен із трьох розроблених методів враховує agile-природу підходів, що застосовуються під час реалізації проєкту, наприклад, розбиття графіку реалізації на спринти.

Отримано наступні наукові результати:

1. *Вперше розроблено метод* поетапного оцінювання проєктів з розробки ПЗ, що, відштовхуючись від ідеї «конусу невизначеності», охоплює етапи

аналізу та проєктування, які передують початку реалізації проєкту. Метод передбачає послідовне виконання трьох етапів оцінювання: попереднього, проміжного та детального. Теоретичні основи методу, зокрема, базуються на представленні змісту проєктних робіт як ієрархічної структури елементів оцінювання, структурі робочого часу інженерів-розробників, системі рівнянь балансу робочого часу, підході «нормалізації» до вимірювання трудоємності змісту проєктних робіт та спроможності розробки проєктної команди. Розроблений метод дає змогу поступово покращувати точність оцінки по мірі поглиблення розуміння вимог до проєкту, враховуючи agile-природу підходів, використовуваних на етапі реалізації. Метод є відносно простим у застосуванні, що, з однієї сторони, не вимагатиме значних інвестицій у навчання експертів, які його використовують, а, з іншої, допускає порівняно нескладну програмну реалізацію зі застосуванням, наприклад, табличного редактора.

2. *Отримав подальший розвиток* метод List Scheduling, використаний для побудови графіку реалізації елементів оцінювання, враховуючи умову балансу між НОР змісту проєктних робіт та НСР проєктної команди; сформульовано критерій якості графіку реалізації елементів оцінювання, який базується на мінімізації функції штрафу від нормалізованого часу простою проєктної команди.

3. *Вперше розроблено метод* підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту. Метод передбачає застосування математичних методів дослідження операцій до розв’язання задачі багатоцільового програмування, яка, в свою чергу, зводиться до послідовності задач цілочисельного програмування із наступним ранжуванням отриманих результатів, використовуючи метод аналізу ієрархій. У поєднанні з методом поетапного оцінювання розроблений метод підтримки прийняття рішень дає змогу підвищити ефективність роботи експертів над оцінкою, пропонуючи оптимальні склад проєктної команди та графік реалізації проєкту.

РОЗДІЛ 3. ОЦІНЮВАННЯ ПРОЄКТІВ З РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЯК СЛАБОСТРУКТУРОВАНИЙ БІЗНЕС-ПРОЦЕС

3.1. Концептуальна схема бізнес-процесу оцінювання

Методи оцінювання проєктів з розробки ПЗ, представлені у розділі 2, призначені для застосування у межах організації, яка надає послуги з розробки ПЗ або розробляє ПЗ для власних потреб. Інтеграція методів оцінювання у виробниче середовище організації можлива у рамках бізнес-процесу, який шляхом координації команди експертів, забезпечує отримання оцінок проєктів з розробки ПЗ. Як зазначається у пункті 1.5.2, за своєю природою такий бізнес-процес є слабоструктурованим, а отже, значною мірою опирається на компетентність та досвід його учасників.

Під *концептуальною схемою слабоструктурованого бізнес-процесу* будемо розуміти множину активностей та приблизний порядок їх виконання; при цьому активність може мати атрибути, наприклад, перелік ролей учасників, що залучені до її виконання. Концептуальна схема відображає певний «ідеальний» перебіг слабоструктурованого бізнес-процесу. В свою чергу, *актуальна схема слабоструктурованого бізнес-процесу* є орієнтованим графом вершинами якого є активності, які мали місце у конкретних екземплярах бізнес-процесу, а дуги відображають порядок виконання цих активностей. Актуальна схема є свого роду узагальненням того, як відбувався перебіг екземплярів бізнес-процесу на практиці. При цьому, як показано у підрозділі 3.3, з метою підвищення зручності користування схемою відображатися може «більш важлива» поведінка, в той час як «менш важливі» активності та переходи між ними приховуються. Концептуальну схему бізнес-процесу оцінювання зображено на рис. 3.1, а для побудови його актуальної схеми розроблено метод, описаний у підрозділі 3.3.

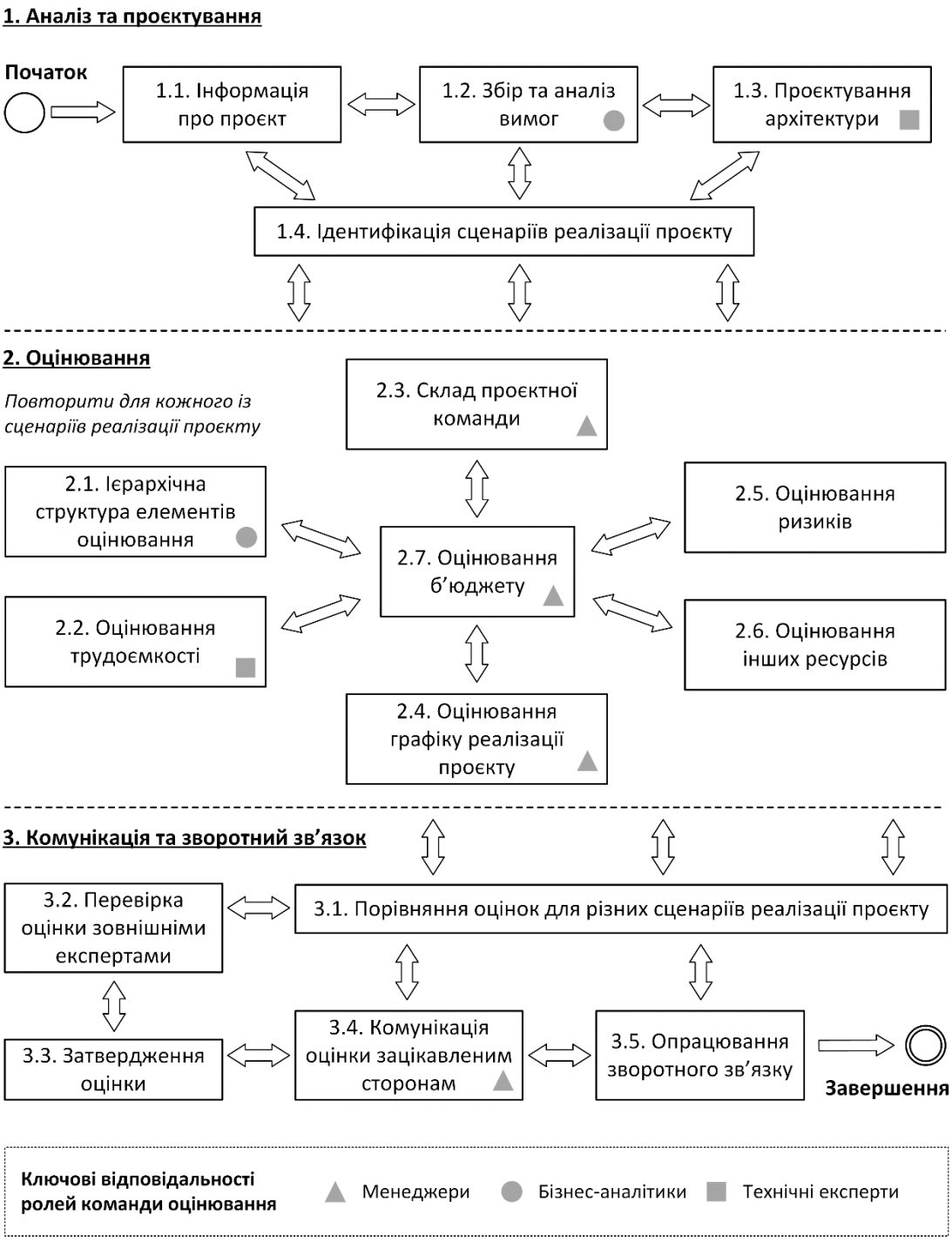


Рис. 3.1. Концептуальна схема слабоструктурованого бізнес-процесу оцінювання проєктів з розробки програмного забезпечення

Зауважимо, що для візуалізації концептуальної схеми обрано довільну нотацію, яка є зручною для читання та відображає суть бізнес-процесу. Такі відомі мови візуального моделювання як UML [147] та BPMN [99] не підходять для такої задачі, оскільки вони призначені для візуалізації структурованих бізнес-процесів. Нотація CMMN (англ. «Case Management Model and Notation») [148],

яка власне й призначена для візуалізації слабоструктурованих бізнес-процесів, також є не найкращим варіантом, оскільки вона відносно не часто застосовується на практиці і є дещо складною для розуміння; також CMMN не дуже добре підтримується сучасними інструментами візуалізації.

Бізнес-процес оцінювання працює на етапах аналізу та проєктування, які, як правило, є початковими етапами життєвого циклу проєкту з розробки ПЗ. Цей процес застосовний до кожного з трьох етапів оцінювання: попереднього, проміжного та детального. Варто підкреслити, що, залежно від етапу оцінювання, на певних активностях бізнес-процесу може бути зроблено більший акцент, а інші взагалі можуть бути пропущені.

Вхідними даними бізнес-процесу оцінювання є постановка задачі на реалізацію проєкту з розробки ПЗ. Як правило, початкове завдання, отримане від замовника, проходить кілька ітерацій уточнення та деталізації на етапі аналізу та проєктування. *Ціль бізнес-процесу* полягає в отриманні оцінки проєкту заданої точності, враховуючи часові та ресурсні обмеження, накладені на надання цієї оцінки. Власне оцінка проєкту, як результат виконання бізнес-процесу, включає складові, описані у підрозділі 1.1.

Як зображено на рис. 3.1., бізнес-процес включає три етапи:

1) аналіз та проєктування;

2) оцінювання;

3) комунікація та зворотний зв’язок.

Як правило, виконання цих етапів відбувається послідовно, хоча не виключаються ситуації, коли, наприклад, з етапу 3 відбувається повернення на етап 2 чи 1. Гнучкість, забезпечена відсутністю жорсткої фіксації послідовності кроків, є одним із ключових факторів, що сприяють ефективному застосуванню учасниками команди оцінювання своїх знань, досвіду та інтуїції, що, в свою чергу, є необхідною умовою отримання унікальних конкурентоспроможних результатів.

Головною метою етапу аналізу та проєктування є уточнення та деталізація функційних та нефункційних вимог до проєкту, а також проєктування архітек-

тури. Важливо відзначити, що аналіз та проєктування є самодостатнім етапом життєвого циклу проєкту, і в бізнес-процесі він відображений лише тією мірою, якою він пов’язаний з оцінюванням. Важливою активністю на цьому етапі є 1.4, метою якої є визначення потенційних сценаріїв реалізації проєкту. Для кожного із таких сценаріїв готується окрема оцінка – далі ці оцінки порівнюються між собою. По мірі поглиблення розуміння вимог до проєкту кількість потенційних сценаріїв реалізації проєкту скорочується: наприклад, на етапі попереднього оцінювання може розглядатися 2-3 сценарії, а на етапі детального оцінювання залишається лише один з них.

За безпосередньо оцінювання проєкту відповідає другий етап бізнес-процесу. Його метою є отримання оцінки для кожного із сценаріїв, визначених на попередньому етапі. Як показує практика, найбільш трудоємними на цьому етапі є активності 2.1 та 2.2, метою яких є відповідно формування ICEO та оцінювання трудоємності елементів оцінювання. Складність цих активностей, в першу чергу, пов’язана з необхідністю опрацювання великого обсягу інформації в стислий проміжок часу. Розроблений метод підтримки прийняття рішень щодо складу команди та тривалості проєкту, описаний у підрозділі 2.11, може застосовуватися під час виконання активностей 2.3 та 2.4. Ключовою активністю на другому етапі є 2.7 – оцінювання бюджету проєкту. Зауважимо, що методи оцінювання, розроблені в рамках цієї дисертаційної роботи, охоплюють активності 2.1-2.4; розробка методів для активностей 2.5-2.7 не входить до завдань даної дисертаційної роботи.

На третьому етапі здійснюється опрацювання оцінки, отриманої на попередньому етапі. Зокрема, проводиться перевірка та, за потреби, затвердження, а також комунікація оцінки зацікавленим сторонам. Збирається та аналізується зворотний зв’язок щодо оцінки. Залежно від того, чи досягнуто цілі бізнес-процесу, відбувається або його завершення, або повернення на попередні етапи для доопрацювання оцінки.

В переважній більшості випадків команда оцінювання об’єднує учасників, що належать до наступних трьох категорій: менеджери, аналітики та технічні експерти.

Ключовою відповідальністю менеджерів є забезпечення виконання цілей бізнес-процесу. Крім управлінських функцій, до сфери їх відповідальності, в першу чергу, входять оцінювання бюджету та комунікація оцінки зацікавленим сторонам. Залежно від конкретних обставин, в яких виконується бізнес-процес оцінювання, менеджери можуть бути представлені проєктними менеджерами, менеджерами з продажів тощо. У випадку, якщо до команди оцінювання входить кілька менеджерів, то один із них вважається головним, тобто саме його відповідальністю є виконання цілей бізнес-процесу.

Аналітики, головним чином, відповідають за збір та аналіз вимог до проєкту (активність 1.2) та формування змісту проєктних робіт у вигляді ієрархічної структури елементів оцінювання (активність 2.1). До категорії аналітиків можуть належати такі фахівці як бізнес-аналітики, експерти у предметній області, дизайнери тощо.

Ключовими сферами відповідальності технічних експертів є розробка архітектури рішення (активність 1.3) та оцінювання трудоємності (активність 2.2). До технічних експертів належать, в першу чергу, архітектори з розробки ПЗ. Також в якості консультантів можуть залучатися інженери-розробники, що володіють високим рівнем компетентності у певних технологіях, які планується застосовувати під час реалізації проєкту. До технічних експертів також можуть належати інженери-тестувальники, які відповідають за оцінку у тій її частині, що пов’язана з тестуванням та контролем якості.

Важливо підкреслити, що, незважаючи на те, що існують задачі, за які експерти з тієї чи іншої категорії відповідають більшою мірою, бізнес-процес не передбачає строгого розмежування сфер відповідальності, створюючи умови для того, щоб кожен із учасників команди оцінювання фокусувався на цілях бізнес-процесу, а не на своїй обмеженій сфері відповідальності.

Крім учасників команди оцінювання, до бізнес-процесу мають відношення також зовнішні (по відношенню до основної команди) експерти, які здійснюють перевірку оцінки та надають рекомендації щодо її покращення. В деяких випадках, наприклад, при перевищенні певного обсягу бюджету, оцінка потребує затвердження керівництвом. Серед зацікавлених сторін, яким оцінка надається для ознайомлення і, можливо, затвердження, є, перш за все, замовники проєкту, керівництво відділу або організації.

Описаний бізнес-процес, в першу чергу, застосовний до оцінювання проєктів, з так званими фіксованим бюджетом, коли необхідно спрогнозувати обсяг часу та ресурсів з метою взяття зобов'язань щодо реалізації проєкту згідно із зробленою оцінкою.

Інформаційна технологія, спроєктована в рамках цієї дисертаційної роботи, автоматизує описаний бізнес-процес (див. розділ 4 та додаток 6). Оскільки бізнес-процес є слабоструктурованим, то інформаційна технологія забезпечує максимальну гнучкість для учасників бізнес-процесу, не фіксуючи послідовність його кроків. З іншої сторони, інформаційна технологія забезпечує моніторинг виконання бізнес-процесу, надсилаючи інформацію про його проходження у модуль RTBPM-E (англ. «Real-Time Business Process Monitoring for Estimation»), який, в свою чергу, здійснює обробку цих даних, користуючись методами процес-майнингу [2], [6], [15] (див. підрозділ 3.5).

3.2. Актуальна схема бізнес-процесу оцінювання

Якість оцінок проєктів з розробки ПЗ, а також витрати на їх підготовку суттєво залежать від того, як бізнес-процес оцінювання застосовується на практиці. Контролювати виконання цього бізнес-процесу в кожному конкретному випадку через наперед визначену модель процесу (наприклад, BPMN) не можливо через те, що бізнес-процес є слабоструктурованим за своєю природою (див. пункт 1.5.2). Тому для моніторингу того, як бізнес-процес оцінювання працює на практиці, пропонується протилежно діаметральний підхід:

1) інформація про кожну виконану активність процесу зберігається у базі даних, таким чином, формуючи «цифровий відбиток» бізнес-процесу;

2) на основі «цифрового відбитку» аналізується фактичне проходження бізнес-процесу, наприклад, такі параметри, як тривалість (бізнес-процесу в цілому та його окремої активності), послідовність виконання активностей тощо;

3) використовуючи дані «цифрового відбитку», будується актуальна схема бізнес-процесу, що показує, які активності та в якій послідовності виконувалися на практиці.

Ефективні методи розв'язання задач, описаних вище, пропонує процес-майнинг [102], [109]. Зокрема, в термінах процес-майнингу задачі 1 та 2 розв'язують шляхом збору так званих event-даних (англ. «event data») [102]. В свою чергу, задача 3 відома під назвою процес-діскавері (англ. «process discovery»). Більше інформації щодо задачі процес-діскавері представлено у пункті 1.5.3.

Розв'язання задач 1-3 відкриває можливості для надання інструментів ефективного моніторингу бізнес-процесу оцінювання, що робить можливим покращення бізнес-процесу, базуючись на реальних даних про його перебіг.

3.3. Метод побудови актуальної схеми слабоструктурованого бізнес-процесу оцінювання

3.3.1. Постановка задачі

Задача полягає у розробці методу, призначенням якого є побудова схеми бізнес-процесу оцінювання з потоку event-даних. Аналогічно до загальної задачі процес-діскавері [102] постановка задачі щодо розробки такого методу є наступною.

Нехай S – неперервний потік event-даних. Тоді метод процес-діскавері D_S забезпечує відображення потоку S на схему бізнес-процесу G так, що схема G представляє реальне виконання бізнес-процесу, відображене у event-даних. Метод повинен задовольняти наступним критеріям:

1) схема бізнес-процесу обновлюється, як тільки надходять нові event-дані (див. також нефункційні вимоги до швидкодії у пункті 3.5.2);

2) візуальне представлення схеми бізнес-процесу є зрозумілим для кінцевих користувачів (які не є експертами у процес-майнингу);

3) схема бізнес-процесу підтримує спрощення так, що поведінка, яка зустрічається рідше має нижчий пріоритет порівняно з поведінкою, яка зустрічається частіше;

4) підтримка еволюції схеми бізнес-процесу (англ. «concept drift») [149], коли «старіша» поведінка плавно заміщується «новішою».

3.3.2. Event-дані: основні поняття

Event-дані (англ. «event data») є основним типом наборів даних, що використовуються у процес-майнингу. Принцип №1, задекларований у процес-майнинг маніфесті [109], стверджує, що наявність event-даних є необхідною передумовою застосування методів процес-майнингу. Терміни, представлені в цьому пункті, потрібні для опису методу побудови актуальної схеми слабоструктурованого бізнес-процесу.

Так, *event-лог* (англ. «event log») є набором даних, згенерованих в результаті багаторазового виконання одного і того ж бізнес-процесу (так званий «цифровий відбиток» бізнес-процесу). Дані, згенеровані виконанням одного екземпляру бізнес-процесу, називатимемо *кейсом* (англ. «case»). Тобто, event-лог є множиною кейсів. В свою чергу, кожен кейс є множиною *подій* (англ. «event»), так що кожна подія належить лише до одного кейсу.

Варто відзначити, що слід розрізняти поняття «подія» та «активність» (англ. «activity»). Під *активністю* розумітимемо крок, виконаний в рамках бізнес-процесу. Кожна подія відноситься до однієї активності, в той час як активність може включати кілька подій. Наприклад, коли активність є тривалою в часі, то з нею можуть бути пов’язані події початку та завершення. Event-лог, кейс та подія мають атрибути, які містять пов’язані з ними дані. Наприклад, типовими атрибутами події є унікальний ідентифікатор та час її виникнення.

Класом подій називатимемо групу однотипних подій (критерії однотипності подій будуються на основі значень їх атрибутів), що належать до одного чи різних кейсів. Визначення класів подій є необхідним для побудови актуальної схеми бізнес-процесу, в якій класи подій (а не самі події) виступають вершинами графу.

Поняття, сформульовані вище, отримали відображення у структурі даних, що представлена у додатку 8. Формальні визначення відповідних термінів можна знайти у [102].

3.3.3. Оригінальна версія методу Fuzzy Miner

Як зазначено у пункті 1.5.3, для вирішення задачі побудови схеми слабоструктурованого бізнес-процесу оцінювання найкраще підходить метод Fuzzy Miner [117], [118]. Початкова версія цього методу розроблена для «стаціонарних» наборів event-даних. Так як на практиці, зазвичай, необхідно працювати із потоками event-даних, існуюча версія методу Fuzzy Miner потребує подальшого розвитку для забезпечення підтримки потоків даних [14].

Оригінальна версіям методу Fuzzy Miner [117], [118] передбачає виконання наступних кроків:

1) розрахунок лог-характестик[1];

2) розрахунок похідних характеристик;

3) візуалізація схеми бізнес-процесу, використовуючи значення характеристик, отриманих на кроках 1 та 2.

Варто підкреслити, що крок 1 вимагає опрацювання всього набору event-даних, який на практиці, з однієї сторони, може бути досить об’ємним, а з іншої – постійно обновлюватися за рахунок надходження нових даних. В свою чергу, крок 2 базується на результатах, отриманих на кроці 1, і не вимагає опрацювання всього набору даних. Крок 3, базуючись на значеннях характеристик, отриманих

[1] Як зазначено у пункті 1.5.3, в оригінальній версії методу Fuzzy Miner використовується англомовний термін «metric» [117], [118]. Оскільки у математичній науці термін «метрика» застосовується у контексті метричних просторів, то в цій дисертаційні роботі в якості україномовного відповідника терміну «metric» використано термін «характеристика».

на кроках 2 та 3, також не вимагає опрацювання всього набору event-даних. Характеристики методу Fuzzy Miner (включаючи як ті, що були визначені в оригінальній версії, так і ті, які були додані в цій дисертаційній роботі) представлені у додатку 7.

В якості програмної реалізації оригінальної версії Fuzzy Miner обрано плагін до програмного продукту ProM [150], вихідний код якого (станом на 29.01.2019) був відкритим та поширювався за ліцензією GNU General Public License (або за будь-якою пізнішою версією цієї ліцензії).

Отже, лише крок 1 оригінальної версії Fuzzy Miner потребує адаптації до потоків event-даних. Метою такої адаптації є організація розрахунку значень лог-характеристик таким чином, щоб зміни у схемі бізнес-процесу оцінювання були доступні для кінцевих користувачів в режимі, наближеному до реального часу (див. пункт 3.5.2), мінімізуючи при цьому використання обчислювальних ресурсів.

3.3.4. Початок та завершення екземплярів бізнес-процесу

В оригінальній версії Fuzzy Miner [117], [118], [150] не було передбачено події початку екземпляру бізнес-процесу – вважалося, що екземпляр бізнес-процесу може розпочатися з будь-якого кроку. Однак, з точки зору кінцевого користувача, важливо розуміти, якою є точка входу в бізнес-процес. У зв’язку з цим у потоковій версії методу додано подію початку екземпляру процесу, що передує будь-якій події цього екземпляру.

Оскільки оригінальна версія Fuzzy Miner [117], [118], [150] була призначена для опрацюваннях «стаціонарних» наборів event-даних, то справедливим було припущення, що всі екземпляри бізнес-процесу є завершеними. Очевидно, що у випадку потоків event-даних, потрібно опрацьовувати незавершені екземпляри бізнес-процесу також. Для кожного конкретного різновиду бізнес-процесів критерії завершеності необхідно визначати окремо. Так, для бізнес-процесу оцінювання визначимо наступні критерії завершеності його екземпляру:

1) коли користувач явно вказує, що оцінювання завершено;

2) якщо не було явно вказано, що оцінювання завершено, але упродовж заданого проміжок часу немає активностей, щодо цього екземпляру.

Аналогічно події початку екземпляру бізнес-процесу у потоковій версії методу Fuzzy Miner передбачено подію завершення, яка є останньою подією у завершеному екземплярі бізнес-процесу. Відповідно, незавершений екземпляр бізнес-процесу не містить події завершення.

Події початку та завершення екземпляру бізнес-процесу опрацьовуються так само, як і всі інші події під час обчислення значень характеристик Fuzzy Miner.

3.3.5. Еволюція схеми бізнес-процесу

З плином часу схема бізнес-процесу може зазнавати суттєвих змін. У процес-майнингу такий «зсув» схеми отримав англомовну назву «concept drift» [149]. У потоковій версії Fuzzy Miner «зсув» схеми бізнес-процесу опрацьовується згідно принципу: більш пізні події мають вищий пріоритет перед більш ранніми подіями. Для опрацювання еволюції схеми бізнес-процесу у потоковій версії Fuzzy Miner застосовано групування екземплярів бізнес-процесу за наступними критеріями:

1) кожна з груп включає не більше як M екземплярів;

2) початок будь-якого екземпляру бізнес-процесу із групи i не перевищує часу початку будь-якого екземпляру з групи $i+1$.

Застосований підхід дає змогу проводити преагрегацію лог-характеристик для кожної з груп екземплярів бізнес-процесу. Зробивши припущення, що значення кожної із лог-характеристик є невід'ємним, значення лог-характеристик для всього набору даних визначається за формулою:

$$L = \sum_{i=1}^{n} \lambda_i L_i , \tag{3.1}$$

де L – значення лог-характеристики для всього набору event-даних, L_i – значення лог-характеристики для групи екземплярів i, λ_i – ваговий коефіцієнт, що ви-

значає важливість екземплярів з групи i, n – кількість груп екземплярів бізнес-процесу.

Очевидно, що для опрацювання еволюції схеми бізнес-процесу, коли більш пізня поведінка вважається більш «важливою», ніж більш рання, вагові коефіцієнти з (3.1) повинні задовольняти умові:

$$0 < \lambda_1 \le \lambda_2 \le \ldots \le \lambda_{n-1} \le \lambda_n \le 1. \quad (3.2)$$

Для прикладу, виконання умови (3.2) можна добитися, якщо задати вагові коефіцієнти наступним чином:

$$\lambda_i = \left(\frac{i}{n}\right)^p, i = \overline{1,n}, \quad (3.3)$$

де параметр $p \ge 0$ дає змогу корегувати ступінь важливості більш ранніх екземплярів: чим більше з значення p, тим менш «важливими» є більш ранні екземпляри бізнес-процесу; очевидно, що при $p = 0$ ігнорується еволюція схеми бізнес-процесу, оскільки $\lambda_i = 1, i = \overline{1,n}$.

В свою чергу, значення L_i визначаються наступним чином:

$$L_i = L_i^* + \sum_k l_i^k, i = \overline{1,n}, \quad (3.4)$$

де L_i^* – значення лог-характеристики для завершених екземплярів із групи екземплярів i, а l_i^k – значення лог-характеристики для k-ого незавершеного екземпляру бізнес-процесу із групи i. Вираз (3.4) підкреслює, що значення метрики для завершених екземплярів дозволяють преагрегацію, оскільки вже не будуть змінюватися, в той час як значення лог-характеристики для ще незавершених екземплярів допускають перерахунок кожного разу, коли надходить нова подія (рис. 3.2).

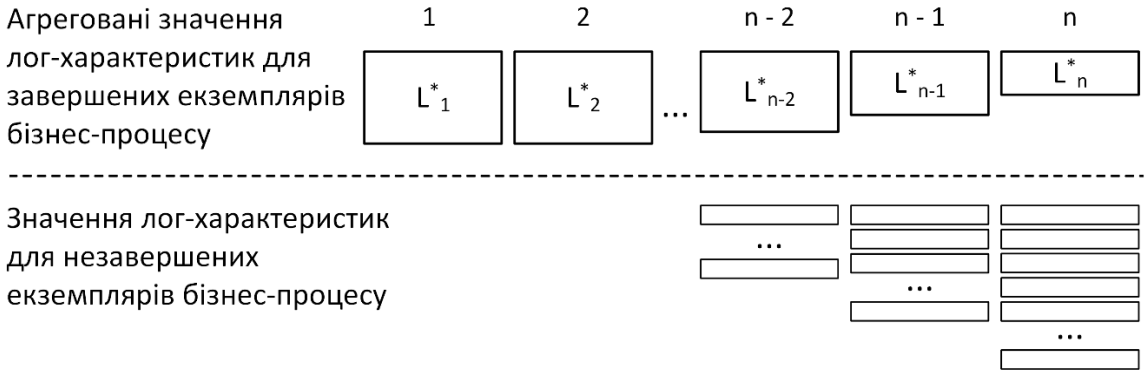


Рис. 3.2. Преагрегація лог-характеристик для завершених та незавершених екземплярів бізнес-процесу

Для опрацювання еволюції схеми бізнес-процесу додано такі нові характеристики до оригінальної версії Fuzzy Miner:

1) «drift unary log characteristic», що забезпечує агрегацію значень унарних лог-метрик, враховуючи еволюцію схеми бізнес-процесу;

2) «drift binary log characteristic», яка здійснює агрегацію бінарних лог-характеристик кореляції (англ. «correlation») та важливості (англ. «significance»), враховуючи еволюцію схеми бізнес-процесу.

Додавання нових характеристик з метою врахування еволюції схеми бізнес-процесу здійснено, не порушуючи парадигму оригінальної версії методу Fuzzy Miner. Тому навіть після зроблених змін підхід до обчислення значень характеристик, закладений ще в оригінальній версії методу, не зазнав принципових змін. Повний перелік характеристик нової версії Fuzzy Miner представлено у додатку 7.

3.4. Реалізація методу побудови актуальної схеми слабоструктурованого бізнес-процесу оцінювання

В рамках цієї дисертаційної роботи програмну реалізацію розробленого методу побудови актуальної схеми бізнес-процесу здійснено у вигляді прототипу, метою якого є проведення експериментів з методом, відкидаючи при цьому другорядні функційні можливості. Варто підкреслити, що обраний підхід до реа-

лізації відкриває можливості для подальшого розвитку до повноцінного програмного продукту, архітектура якого описана у підрозділі 3.5 [1], [6].

Програмну реалізацію розробленого методу візуалізовано у вигляді діаграми компонентів на рис. 3.3.

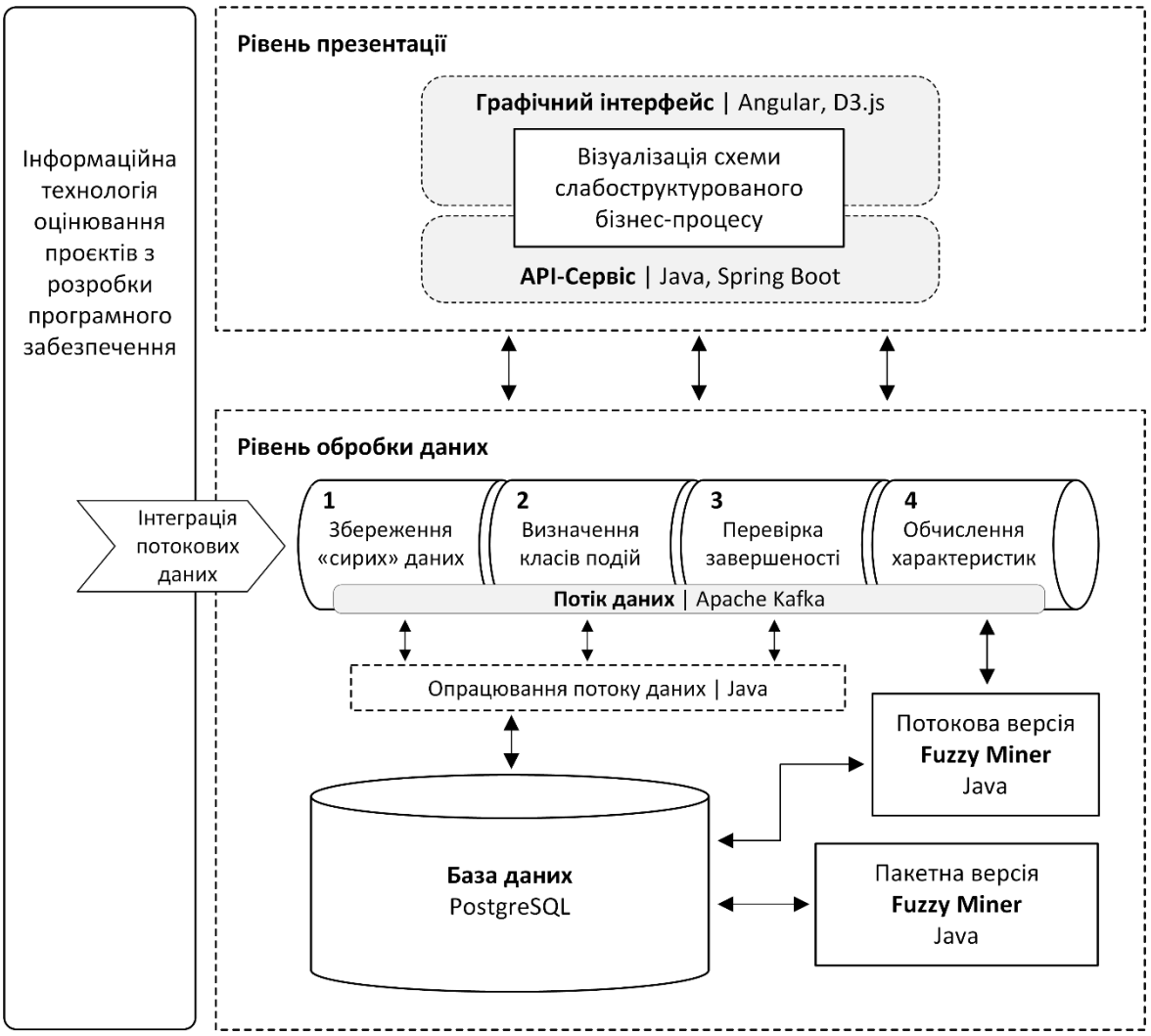


Рис. 3.3. Діаграма компонентів реалізації методу побудови схеми слабоструктурованого бізнес-процесу оцінювання

Ключовим аспектом реалізації розробленого методу побудови схеми бізнес-процесу є застосування лямбда-архітектури [151], [152], що забезпечує обробку потоку event-даних [7], [9], [10], [11], [12]. При такій архітектурі event-дані неперервно надходять у чергу повідомлень, де відбувається їх поетапна обробка у режимі, наближеному до реального часу:

Етап 1. Збереження «сирих» event-даних у базі даних.

Етап 2. Визначення класів подій.

Етап 3. Перевірка завершеності відповідного екземпляру бізнес-процесу.

Етап 4. Розрахунок значень лог-характеристик.

У базі даних відбувається накопичення як «сирих» event-даних, так і результатів їх обробки, а також зберігаються значення характеристик Fuzzy Miner. Опис реляційної структури бази даних представлено у додатку 8.

Як видно із діаграми на рис. 3.3, реалізація включає дві версії Fuzzy Miner:

1. *Пакетна* (від англ. «batch») версія методу [2] передбачає опрацювання всіх накопичених event-даних для обчислення значень лог-характеристик. Ця версія схожа до оригінального методу Fuzzy Miner, але при цьому враховує завершеність екземплярів (див. пункт 3.3.4) та еволюцію схеми бізнес-процесу (див. пункт 3.3.5).

2. *Потокова* версія методу [14] працює на етапі 4 обробки потоку event-даних, здійснюючи обчислення значень лог-характеристик по мірі надходження подій через чергу повідомлень.

Власне побудова та візуалізація схеми бізнес-процесу відбувається на рівні презентації, що включає серверну (API-сервіс) та клієнтську (веб-програма) частини. На запит користувача API-сервіс здійснює розрахунок значень похідних характеристик Fuzzy Miner на основі відповідних значень лог-характеристик, збережених в базі даних (крок 2 на рис. 3.4). Відштовхуючись від значень характеристик Fuzzy Miner, API-сервіс будує граф актуальної схеми бізнес-процесу (крок 3 на рис. 3.4). Побудований граф передається у веб-програму для його подальшої візуалізації (крок 4 на рис. 3.4).

Вибір технологій програмної реалізації методу побудови схеми слабоструктурованого бізнес-процесу (табл. 3.1) здійснювався за наступними головними критеріями:

1) ліцензія з відкритим вихідним кодом;

2) стабільність та «зрілість»;

3) можливість розвитку до програмного продукту (архітектура якого описана у підрозділі 3.5).

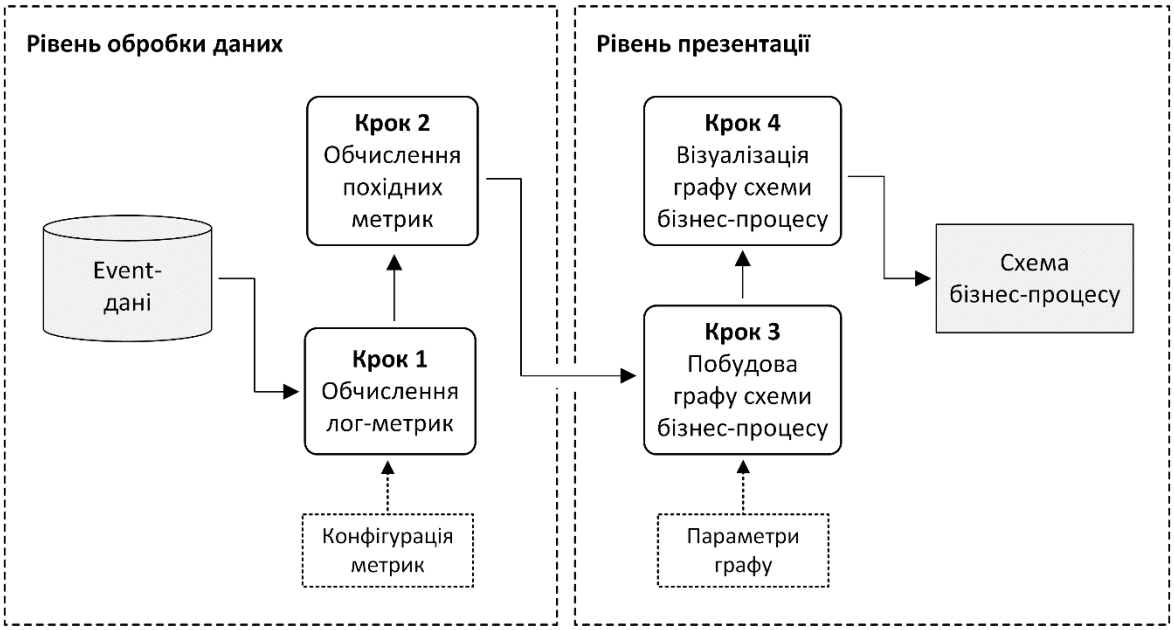


Рис. 3.4. Реалізація візуалізації схеми слабоструктурованого бізнес-процесу

Таблиця 3.1

Технології програмної реалізації методу побудови
схеми слабоструктурованого бізнес-процесу

№	*Технологія реалізації*	*Ліцензія технології реалізації*	*Реалізовані компоненти*
1	Java (OpenJDK)	GNU General Public License v2.0	Інтеграція даних, опрацювання потоку event-даних, пакетна версія Fuzzy Miner, потокова версія Fuzzy Miner API-сервіс
2	Apache Kafka	Apache License 2.0	Черга повідомлень
3	PostgreSQL	PostgreSQL License	База даних
4	SpringBoot	Apache License 2.0	API-сервіс
5	Angular	MIT License	Графічний інтерфейс
6	D3.js	ISC License	Візуалізація схеми слабоструктурованого бізнес-процесу

Вибір на Java впав, оскільки так склалося історично, що ця технологія використовувалася для реалізації методів процес-майнингу. Зокрема, ProM (програмний продукт процес-майнингу для академічних цілей) [153] та плагіни до нього (включаючи реалізацію оригінальної версії Fuzzy Miner [150]) також реалізовані на Java.

Описана вище архітектура прототипу реалізації методу побудови схеми слабоструктурованого бізнес-процесу була взята за основу архітектури модуля RTBPM-E, що описана у наступному підрозділі.

3.5. Архітектура модуля RTBPM-E

У попередньому підрозділі було представлено реалізацію розробленого методу побудови схеми слабоструктурованого бізнес-процесу. Однак, здійснена реалізація є прототипом, метою якого є демонстрація можливостей методу. В цьому підрозділі представлена архітектура модуля моніторингу слабоструктурованих бізнес-процесів, що є логічним продовженням прототипу із підрозділу 3.4. Цей модуль називатимемо RTBPM-E, що є скороченням від англомовного «Real-Time Business Process Monitoring for Estimation».

3.5.1. Призначення та функційні можливості модуля RTBPM-E

Головною функцією модуля RTBPM-E є побудова та візуалізація актуальної схеми слабоструктурованого бізнес-процесу оцінювання в режимі, наближеному до реального часу, застосовуючи розроблений метод, описаний у підрозділі 3.3. При цьому архітектура модуля повинна підтримувати можливість реалізації й інших функцій (що не входять до задач цієї дисертаційної роботи), наприклад: прогноз часу завершення екземпляру бізнес-процесу, рекомендації щодо наступних кроків, попередження в разі відхилень, порівняння концептуальної та актуальної схем тощо. Підкреслимо, що в основу модуля RTBPM закладено більш глобальний задум – моніторинг слабоструктурованих бізнес-процесів у різних предметних галузях [1], [6], [8]. В рамках цієї дисертаційної роботи RTBPM адап-

товано до бізнес-процесу оцінювання [15] (саме тому до назви модуля додається літера «E», що означає «estimation»).

3.5.2. Нефункційні вимоги до модуля RTBPM-E

Крім функційних можливостей, описаних у попередньому пункті, на архітектуру модуля RTBPM-E мають суттєвий вплив нефункційні вимоги:

1. *Швидкодія* (англ. «performance»). Головною вимогою є те, що модуль повинен опрацьовувати event-дані, що надходять від інших систем, у режимі наближеному до реального часу. Як відомо, швидкодія характеризується такими параметрами:

– *час затримки* (англ. «latency») – період, що проходить від моменту виникнення події у інформаційній технології, що автоматизує бізнес-процес оцінювання (див. розділ 4), до моменту її опрацювання модулем RTBPM-E;

– *пропускна здатність* (англ. «throughput») – обсяг event-даних, який опрацьовується модулем RTBPM-E за одиницю часу.

На цьому етапі проєктування архітектури модуля RTBPM-E важко оцінити числові значення цих параметрів, оскільки вони дуже залежать від конкретних умов експлуатації, як наприклад, інфраструктура розгортання. При цьому зауважимо, що вимога щодо опрацювання event-даних в режимі, наближеному до реального часу, стосується, головним чином, першого з параметрів – часу затримки. Для RTBPM-E значення часу затримки до 30 хв вважатимемо прийнятним для режиму часу, наближеного до реального.

2. *Масштабованість* (англ. «scalability») в даному контексті означає здатність системи змінювати час затримки та пропускну здатність залежно від обсягу даних, які необхідно опрацьовувати. На практиці варіацію значень цих параметрів можна досягнути, зокрема, змінюючи кількість екземплярів компонентів системи, наприклад, черги повідомлень: чим більша кількість таких екземплярів, тим краща пропускна здатність. Так як кількість екземплярів компонентів впливає на вартість експлуатації системи, то необхідно забезпечити функціонування системи у її «мінімальній» конфігурації, коли обсяги даних не є надто великими,

та у «масштабованій» конфігурації, коли обсяги даних суттєво зростають. Така варіативність повинна забезпечуватися без зміни архітектури системи.

3. *Сумісність* (англ. «interoperability»). Модуль RTBPM-E повинен підтримувати інтеграцію із різними типами джерел даних, зокрема, інтеграцію з API-сервісами інших систем, зчитування повідомлень з черг повідомлень, файлову інтеграцію, інтеграцію з базами даних. Такі інтеграції повинні підтримувати два режими: пакетний (періодичне зчитування порцій даних) та потоковий (неперервне надходження даних). З іншої сторони, модуль повинен забезпечувати зворотну інтеграцію з іншими системами, надаючи для цього інтеграційний API.

4. *Незалежність від платформи розгортання* означає, що архітектура модуля, зокрема, технології реалізації, повинні дозволяти розгортання системи як у приватних центрах обробки даних, так і, використовуючи «хмарні» сервіси (наприклад, Microsoft Azure чи Amazon Web Services).

5. *Розширюваність* (англ. «extensibility») полягає у можливості додавання нових функційних можливостей модуля без суттєвих змін у його архітектурі.

6. *Гнучкість* (англ. «flexibility»). Адаптація до нового типу слабоструктурованого бізнес-процесу без суттєвих змін у архітектурі модуля. При цьому така адаптація не виключає зміни у вихідному коді та необхідність випуску нового релізу.

7. Технології реалізації повинні поширюватися за ліцензіями з відкритим вихідним кодом, бути досить стабільними та відомими у сфері розробки ПЗ. Це дозволяє з однієї сторони мінімізувати вартість ліцензій, а з іншої дає певну гарантію того, що технології реалізації будуть актуальними ще наступні кілька років. Аналогічний підхід до вибору технологій реалізації застосований для прототипу методу побудови схеми слабоструктурованого бізнес-процесу (див. підрозділ 3.4).

3.5.3. Діаграма компонентів RTBPM-E

Архітектура модуля RTBPM-E зображена у вигляді діаграми компонентів на рис. 3.5. Аналогічно до прототипу (див. підрозділ 3.4), архітектура модуля RTBPM-E включає дві основні складові:

1) *рівень презентації*, який відповідає за графічний інтерфейс та взаємодію з користувачами;

2) *рівень обробки даних*, призначенням якого є збір, обробка та накопичення даних.

Як видно із діаграми на рис. 3.5, модуль RTBPM-E, як основну функційну можливість, реалізує метод побудови схеми слабоструктурованого бізнес-процесу як на рівні презентації, так і на рівні обробки даних. При цьому варто підкреслити, згідно з вимогою розширюваності (див. пункт 3.5.2) розроблена архітектура підтримує також (без суттєвих змін) додавання й інших функційних можливостей, наприклад: прогнозування часу завершення бізнес-процесу, перевірку відповідності актуальної та концептуальної схем бізнес-процесу, виявлення відхилень у проходженні бізнес-процесу тощо.

Модуль RTBPM-E передбачає інтеграцію з інформаційною технологією оцінювання проєктів з розробки ПЗ (див. розділ 4). Зокрема, у модуль RTBPM-E передаються event-дані, що є «цифровим відбитком», залишеним проходженням бізнес-процесу оцінювання. При цьому передбачено два варіанти такої інтеграції:

1. *Потокова інтеграція*, коли дані надходять неперервно в режимі часу, наближеному до реального. Як видно із реалізації прототипу (див. підрозділ 3.4), такий підхід передбачає опрацювання потокових даних із застосуванням лямбда-архітектури [151], [152].

2. *Пакетна інтеграція*, що передбачає порційне завантаження даних за визначеним розкладом, наприклад, раз в годину чи раз на добу. Така інтеграція базується на патерні ETL (англ. «Extract, Transform, Load»).

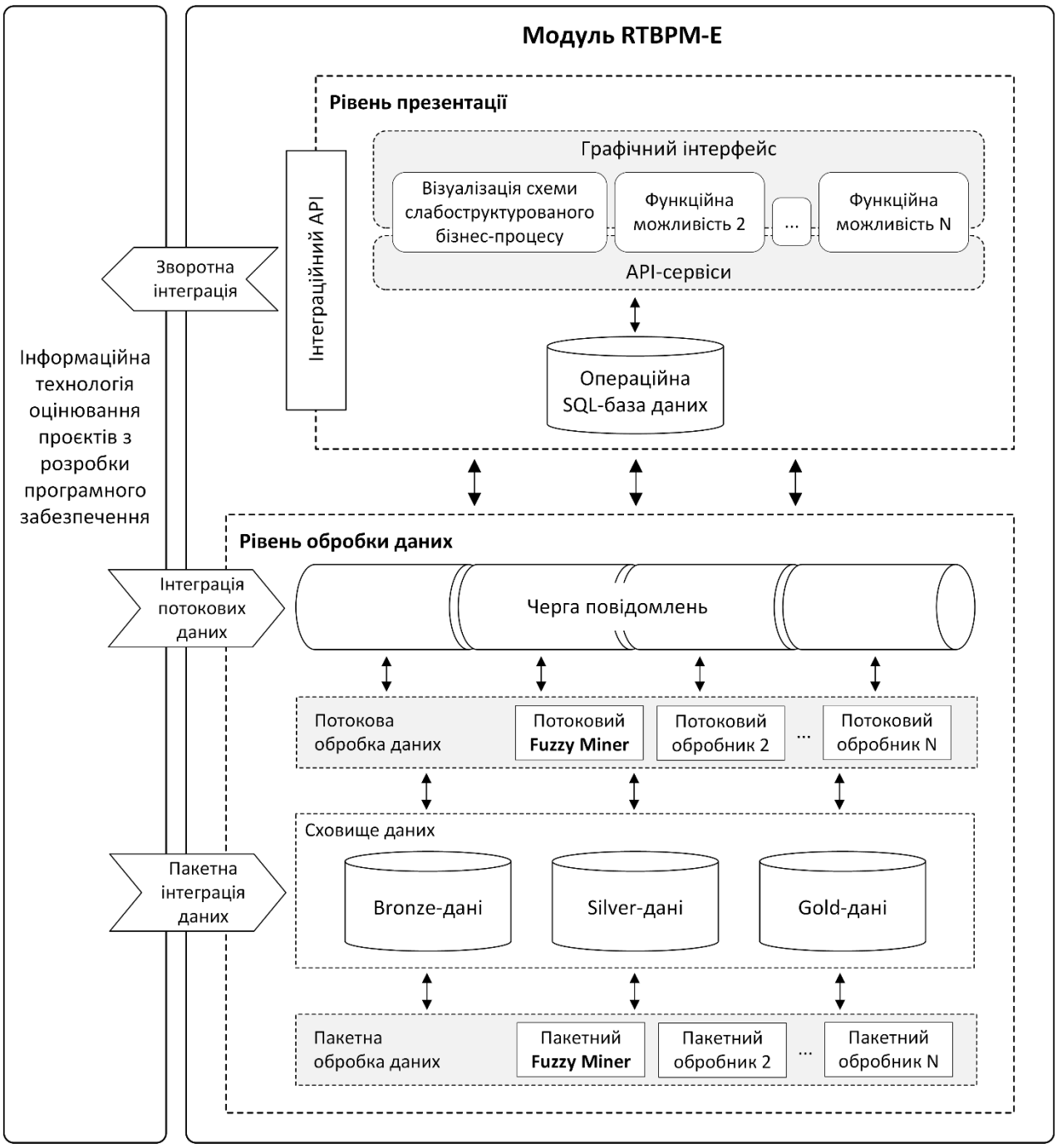


Рис. 3.5. Діаграма компонентів модуля RTBPM-E

Окрім інтеграції даних, які надходять від інформаційної технології у модуль RTBPM-E, передбачена також зворотна інтеграція: коли інформаційна технологія отримує повідомлення від RTBPM-E. Прикладом застосування такої інтеграції може слугувати рекомендації користувачам щодо наступних кроків під час оцінювання або надсилання попереджень користувачам в разі відхилень у проходженні бізнес-процесу оцінювання.

Рівень презентації складається із серверної та клієнтської частин. Серверна частина базується на мікросервісній архітектурі, а клієнтська є веб-програмою. До складу рівня презентації входить також операційна SQL-база даних, з якою безпосередньо взаємодіють API-сервіси. В цій базі даних, наприклад, зберігається інформація про користувачів модуля. Варто відзначити, що API-сервіси також зчитують gold-дані з рівня обробки даних.

Рівень обробки даних побудований на основі поєднання лямбда-архітектури [151], [152] та патерну «медальйон» (англ. «medallion») [154]. Як описано у підрозділі 3.4, лямбда-архітектура, застосовуючи чергу повідомлень, забезпечує опрацювання потоку event-даних у режимі, наближеному до реального часу. З іншої сторони, згідно патерну «медальйон» обробка та накопичення даних передбачає наступні три етапи:

1. *Bronze-дані* є «сирими» даними, отриманими від джерел даних. В першу чергу, це event-дані, згенеровані виконанням бізнес-процесу оцінювання. На цьому етапі структура даних відповідає структурі у відповідних джерела даних.

2. *Silver-дані* проходять очистку, нормалізацію структури та встановлення зв’язків між таблицями. Для прикладу, до silver-даних належать even-дані, для яких визначені класи подій та належність до екземпляру бізнес-процесу.

3. *Gold-дані* призначені для безпосереднього використання «споживачами» даних. До таких даних, наприклад, належать значення характеристик бізнес-процесу. Доступ до gold-даних мають API-сервіси із рівня презентації.

Важливо, що потокова обробка даних доповнюється пакетною обробкою, яка, в свою чергу, базується на застосуванні патерну ETL. На відміну від обробки потоку даних, яка здійснюється в режимі часу, наближеному до реального, пакетна обробка даних спрацьовує періодично згідно розкладу (наприклад, один раз на добу). Пакетна обробка, як правило, є тривалою в часі (від кількох хвилин до кількох десятків хвилин) і передбачає одночасне опрацювання великого обсягу даних. Підкреслимо, що розроблений метод побудови схеми слабострук-

турованого бізнес-процесу передбачає як «потокову», так і «пакетну» складові (див. підрозділ 3.4).

3.5.4. Технології реалізації RTBPM-E

Вибір технологій реалізації модуля RTBPM-E здійснювався за такими критеріями:

1. *Мінімізація витрат на ліцензії*. В першу чергу розглядаються технології, що поширюються згідно ліцензій з відкритим вихідних кодом з якомога меншою кількістю обмежень (наприклад, Apache License 2.0, MIT License та аналогічні їм).

2. *Популярність, стабільність та «зрілість» технології*. Цей критерій передбачає, що:

– технологія набула популярності та активно застосовується упродовж кількох років;

– на ринку праці є достатня кількість фахівців, які володіють цією технологією та мають бажання з нею працювати;

– технологія постійно розвивається, розробники технології регулярно випускають нові версії;

– з високою ймовірністю, технологія збереже свою актуальність упродовж наступних 5-7 років.

3. *Незалежність від інфраструктури розгортання*. Технологія може розгортатися як на інфраструктурі центру обробки даних компанії, так і на одному із «хмарних» сервісів (наприклад, Microsoft Azure або Amazon Web Services).

Технології реалізації компонентів рівня презентації представлені у табл. 3.2. Слід відзначити, що D3.js обрано, як технологію візуалізації схеми слабоструктурованого бізнес-процесу, перш за все, через її гнучкість, що робить можливою реалізацію візуалізацій досить високої складності. Альтернативна технологія, Plotly, не є настільки гнучкою, але вимагає при цьому менше часу на розробку.

Таблиця 3.2

Технології програмної реалізації рівня презентації модуля RTBPM-E

№	***Компонент модуля RTBPM-E***	***Рекомендовані технології реалізації***	***Альтернативні технології реалізації***
1	Графічний інтерфейс	Angular	ReactJS, NextJS
2	Візуалізація схеми бізнес-процесу	D3.js	Plotly
3	API-сервіси	Java SpringBoot	ASP.NET, Python Fast API
4	Операційна база даних	PostgreSQL	MySQL, MS SQL Server

Технології реалізації компонент рівня обробки даних представлені у табл. 3.3. Зроблений вибір технологій мотивований наступними причинами:

– *Apache Kafka* є однією із провідних технологій для реалізації черги повідомлень, що характеризується високою надійністю та здатністю до масштабування залежно від обсягів даних (зауважимо, що схожа технологія Apache Storm вже втратила свою актуальність [7]).

– *Java* обрана для обробки потокових даних згідно з рекомендаціями розробників Apache Kafka. Зауважимо, що Python є також у цьому переліку рекомендованих технологій.

– Вибір *Python* для пакетної обробки даних (іншими словами, для реалізації ETL-обробників) зумовлений тим, що Python є однією із найбільш поширених та потужних технологій роботи з даними.

– Використання *PostgreSQL* для реалізації патерну «медальйон» зумовлено її безкоштовністю, високою надійністю та швидкодією, а також підтримкою роботи з «часовими» даними (англ. «time series data»).

Таблиця 3.3

Технології програмної реалізації
рівня обробки даних модуля RTBPM-E

№	*Компонент модуля RTBPM-E*	*Рекомендовані технології реалізації*	*Альтернативні технології реалізації*
1	Черга повідомлень	Apache Kafka	RabbitMQ, Apache ActiveMQ
2	Обробка потокових даних	Java	Python
3	Пакетна обробка даних	Python	Java, SQL
4	Сховище даних	PostgreSQL	InfluxDB, MS SQL Server

Альтернативним підходом до вибору технологій реалізації модуля RTBPM-E є використання «хмарних» сервісів, наприклад, Amazon Web Services [13]. В цьому випадку можна досягнути певного зменшення вартості та часу розробки. Але при цьому втрачається гнучкість, оскільки у випадку заміни одного «хмарного» сервісу іншим необхідно переробляти значну частину системи.

3.5.5. Інтеграція RTBPM-E з інформаційною технологією оцінювання проєктів з розробки програмного забезпечення

У пункті 3.5.3 описано технічні аспекти інтеграції модуля RTBPM-E та інформаційної технології оцінювання проєктів з розробки ПЗ. Варто також відзначити, що зі сторони інформаційної технології також необхідно провести технічну підготовку до інтеграції даних, забезпечивши генерацію event-даних, які є цифровим відбитком виконання бізнес-процесу (див. розділ 4). Однак, для забезпечення взаємодії цих систем на практиці, крім технічної реалізації інтеграції даних з обох сторін, необхідно виконати наступну послідовність кроків:

1. *Підготовка* включає початковий збір даних та побудову першої версії актуальної схеми бізнес-процесу оцінювання. Підкреслимо, що необхідною умовою для виконання цього кроку є попереднє накопичення певного обсягу

event-даних, необхідних для побудови першої версії схеми бізнес-процесу. На цьому етапі важливо провести аналіз зібраних event-даних, наприклад, для отримання інформації про атрибути подій, їх класи та частоту виникнення. Однією із ключових складових цього етапу є збір зворотного зв'язку від користувачів модуля RTBPM-E та учасників бізнес-процесу оцінювання щодо першої версії актуальної схеми бізнес-процесу.

2. *Калібрування* налаштувань модуля RTBPM-E має за мету уточнення значень параметрів, щоб отримати версію актуальної схеми бізнес-процесу, яка є більш «якісною» ніж попередня. Зауважимо, що критерії якості визначаються в кожному конкретному випадку окремо, при цьому вагомим орієнтиром для їх визначення є зворотний зв'язок, отриманий від користувачів модуля RTBPM-E та учасників бізнес-процесу оцінювання. Калібрування, зокрема, стосується налаштування процедури визначення класів подій, уточнення критеріїв завершеності екземпляру бізнес-процесу, визначення вагових коефіцієнтів для характеристик Fuzzy Miner тощо.

3. *Моніторинг* перебігу бізнес-процесу оцінювання власне і є ключовою функцією модуля RTBPM-E.

4. *Покращення* бізнес-процесу оцінювання, використовуючи результати моніторингу його перебігу, зокрема, побудовану на основі накопичених event-даних актуальну схему.

Кроки 2-4 повинні повторюватися ітераційно, забезпечуючи неперервний цикл контролю та покращення бізнес-процесу оцінювання. Зауважимо, що описані вище кроки 1-4, перегукуються з ідеями, закладеними у підході L^* [102], що застосовується у процес-майнингу.

Висновки до розділу 3

Оцінювання проєктів з розробки ПЗ в рамках організації (яка розробляє ПЗ для власних потреб чи на замовлення) здійснюється як бізнес-процес, який за своєю природою є слабоструктурованим (див. обґрунтування у пункті 1.5.2). Слабка структурованість бізнес-процесу оцінювання зумовлена, з однієї сторони, необхідністю опиратися на унікальні знання та досвід його учасників, а з іншої – самою природою проєктів з розробки ПЗ, які часто є інноваційними і містять значний рівень невизначеності та ризику.

Для роботи з оцінюванням проєктів з розробки ПЗ як слабоструктурованим бізнес-процесом у цій дисертаційній роботі виконано наступне:

– розроблено концептуальну схему бізнес-процесу оцінювання, що відображає основні активності, приблизну послідовність їх виконання, а також ключові відповідальності виконавців;

– розроблено метод побудови актуальної схеми бізнес-процесу оцінювання, який є подальшим розвитком методу процес-майнингу Fuzzy Miner;

– здійснено програмну реалізацію розробленого методу побудови актуальної схеми бізнес-процесу у вигляді прототипу, метою якого є проведення експериментів та демонстрація можливостей методу;

– розроблено архітектуру модуля RTBPM-E, що призначений для моніторингу бізнес-процесу оцінювання.

При цьому отримано наступні наукові результати:

Отримав подальший розвиток метод Fuzzy Miner (початкова версія якого була розроблена Ch.W. Gunther та W.M.P. van der Aalst), до основної здатності якого будувати схеми слабоструктурованих бізнес-процесів додано також опрацювання потоків даних, що дає змогу підтримувати схему бізнес-процесу в актуальному стані, опрацьовуючи нові event-дані, що постійно надходять. Розроблений метод також включає можливість врахування еволюції схеми бізнес-процесу, вважаючи пізнішу поведінку більш пріоритетною ніж ранішу.

Важливо підкреслити, що розроблений метод побудови схеми слабоструктурованого бізнес-процесу та його програмне забезпечення відкривають можливості для майбутньої реалізації наступних задач (що не входять до завдань цієї дисертаційної роботи):

– прогнозування часу завершення екземпляру бізнес-процесу;

– порівняння актуальної та концептуальної схем бізнес-процесу (англ. «conformance checking»);

– попередження щодо відхилень під час проходження бізнес-процесу;

– надання рекомендацій учасникам команди оцінювання щодо наступних кроків;

– побудова актуальної схеми взаємодії між учасниками бізнес-процесу на основі накопичених event-даних (англ «social mining»).

РОЗДІЛ 4. АНАЛІЗ ВИМОГ, ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ОЦІНЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

В цьому розділі представлено аналіз вимог та проєктування архітектури інформаційної технології оцінювання проєктів з розробки ПЗ. Розроблені в цій дисертаційні роботі методи (див. розділ 2) застосовані до оцінювання інформаційної технології, зокрема, послідовно підготовано попередню, проміжну та детальну оцінки, поступово поглиблюючи розуміння вимог, деталізуючи зміст проєктних робіт та архітектуру.

4.1. Завдання інформаційної технології

Перед інформаційною технологією оцінювання проєктів з розробки ПЗ поставлено наступні завдання:

1. Автоматизація бізнес-процесу оцінювання проєктів з розробки ПЗ (див. підрозділ 3.1), застосовуючи розроблені в цій дисертаційні роботи методи:

– метод поетапного оцінювання (див. підрозділи 2.1-2.9);

– метод побудови розкладу реалізації елементів оцінювання (див. підрозділ 2.10);

– метод підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди (див. підрозділ 2.11).

2. Забезпечення збору, збереження, обробки та візуалізації даних про реальний перебіг оцінювання як слабоструктурованого бізнес-процесу, користуючись методами процес-майнингу, зокрема, розробленим методом побудови схеми слабоструктурованого бізнес-процесу (ця частина інформаційної технології отримала назву «модуль RTBPM-E», див. розділ 3).

3. Накопичення даних щодо оцінювання проєктів з розробки ПЗ у базі даних інформаційної технології, що в перспективі відкриває можливості для реалізації завдань, що не входять до цієї дисертаційної роботи, зокрема: застосування методів оцінювання за аналогією, калібрування параметрів методів оцінювання шляхом застосування ШІ та машинного навчання, ефективного

опрацювання зворотного зв’язку, порівнюючи оцінені та фактично затрачені обсяги ресурсів та часу.

4.2. Сфера застосування інформаційної технології

Інформаційна технологія призначена для використання організаціями, в яких відбувається багатократне повторення бізнес-процесу оцінювання проєктів з розробки ПЗ. До таких, зокрема, належать організації, які:

1) спеціалізуються на розробці ПЗ на замовлення;

2) розробляють власні програмні продукти;

3) розробляють ПЗ для власних потреб.

Передбачається, що інформаційна технологія буде функціонувати в IT-екосистемі організації-користувача, взаємодіючи (в разі необхідності) з іншим програмним забезпеченням цієї організації.

4.3. Користувачі інформаційної технології

Інформаційна технологія призначена для застосування наступними категоріями користувачів:

1. *Команда оцінювання*, до якої входять менеджери, бізнес-аналітики та технічні експерти.

2. *Експерти з перевірки оцінок*, які надають відгуки щодо підготованих оцінок (забезпечуючи тим самим підвищення рівня їх якості та усунення недоліків).

3. *Керівництво*, відповідальністю якого є затвердження оцінок (що, зокрема, актуально для проєктів, оцінки бюджету яких перевищують визначені межі).

Інформація про учасників бізнес-процесу оцінювання також представлена у підрозділі 3.1 та додатку 6.

4.4. Функційні можливості інформаційної технології

З метою забезпечення виконання своїх завдань інформаційна технологія надає користувачам наступні функційні можливості (рис. 4.1):

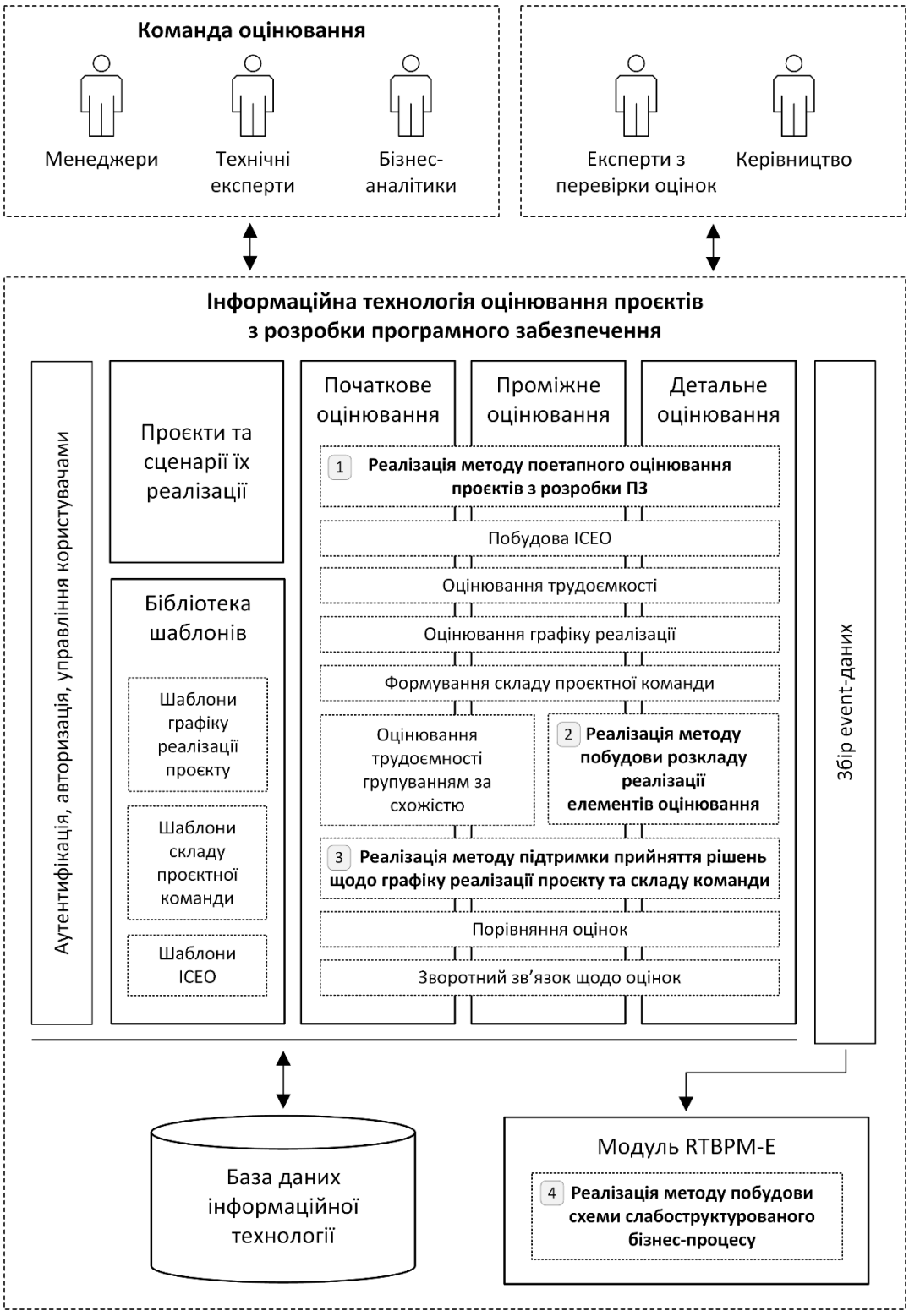


Рис. 4.1. Користувачі та функційні можливості інформаційної технології оцінювання проєктів з розробки програмного забезпечення

1. *Проєкти та сценарії їх реалізації.* Ця функція включає CRUD-операції (англ. «Create, Read, Update, Delete») над проєктами та сценаріями їх реалізації.

Зауважимо, що один проєкт може мати один і більше потенційних сценаріїв реалізації.

2. *Оцінки проєктів.* Підкреслимо, що в рамках інформаційної технології об'єктом оцінювання є не сам проєкт, а сценарій його реалізації. Для одного сценарію реалізації може бути надано кілька оцінок; зокрема, такі оцінки можуть бути поетапними (див. підрозділ 2.1). Ця функція включає: CRUD-операції над оцінками, збереження різних версій оцінки, порівняння оцінок.

3. *Побудова ICEO* включає як CRUD-операції над ICEO, так і CRUD-операції над елементами оцінювання. При цьому ICEO може бути пов'язана як з проєктом, так і з сценаріями його реалізації (наприклад, коли різні сценарії передбачають відмінний зміст проєктних робіт). Ця група функційних можливостей також включає CRUD-операції над атрибутами елементів оцінювання (включаючи атрибути, які відповідають за HOP). В поточній версії інформаційної технології передбачено представлення ICEO у деревовидному та табличному виглядах, інші варіанти візуалізації ICEO не входять до змісту робіт поточної версії.

4. *Оцінювання трудоємності групуванням за схожістю*. Відповідний підхід до оцінювання трудоємності рекомендований для попереднього оцінювання (див. пункт 2.7.5). Ця функційна можливість включає візуалізацію елементів оцінювання на двовимірній координатній площині, визначення значень атрибутів EIP та EIU, а також розрахунок оптимістичної та песимістичної HOP.

5. *Склад проєктної команди*. Ця група функцій включає CRUD-операції щодо проєктних команд, враховуючи те, що кожна оцінка асоційована із складом проєктної команди. Один і той самий склад проєктної команди може бути асоційований із кільком оцінками (наприклад, оцінки для різних сценаріїв можуть використовувати один і той самий склад проєктної команди). Поряд із статичною частиною склад проєктної команди включає також і варіативну частину. Ця варіативна частина, зокрема, використовується у методі підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту (див. підрозділ 2.11),

а також забезпечує можливість модифікації складу команди для оптимістичної та песимістичної оцінок одного сценарію.

6. *Графік реалізації проєкту* включає можливості визначення ієрархічної структури фаз проєкту, кількість рівнів ієрархії якої залежить від типу оцінювання та специфіки проєкту. Графік реалізації проєкту включає також варіативну частину, що, зокрема, дозволяє застосовувати розроблений метод підтримки прийняття рішень, описаний у підрозділі 2.11.

7. *Розклад реалізації елементів оцінювання,* який формується під час детального оцінювання, базується на методі, описаному в підрозділі 2.10. Функція автоматичної побудови розкладу доповнюється можливістю користувачів робити зміни у побудованому розкладі.

8. *Реалізація методу підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди* (див. підрозділ 2.11). Підкреслимо, що ця функція включає розв’язання послідовності задач цілочисельного програмування та застосування методу аналізу ієрархій для ранжування варіантів, що пропонуються користувачу для прийняття рішення.

9. *Зворотний зв'язок щодо оцінки* передбачений на третьому етапі бізнес-процесу оцінювання (див. підрозділ 3.1) в наступних ситуаціях: перевірці оцінки зовнішніми (відносно команди оцінювання) експертами, затвердження оцінки керівництвом, ознайомлення з оцінкою зацікавлених сторін (наприклад, замовників).

10. *Бібліотека шаблонів* є зібранням компонентів, що призначені для повторного використання. До бібліотеки входять шаблони наступних типів: графіків реалізації проєктів, проєктних команд, фрагментів ICEO. Головним завданням таких шаблонів є підвищення ефективності процесу оцінювання шляхом багаторазового використання вже готових компонентів.

11. *Аутентифікація, авторизація, управління користувачами, профіль користувача* є набором стандартних функцій безпеки інформаційної системи. Передбачається, що аутентифікація здійснюватиметься через єдиний сервіс

аутентифікації організації (англ. «corporate single sign-on»). Такі функції як реєстрація користувача, зміна паролю, двофакторна аутентифікація реалізуються системою безпеки організації (а тому не включені до функційних можливостей інформаційної технології).

12. *Збір event-даних* необхідний для моніторингу проходження бізнес-процесу оцінювання методами процес-майнингу. Власне накопичення та опрацювання відповідних даних здійснюється в модулі RTBPM-E (див. розділ 3).

Важливо підкреслити, що вся інформація щодо проєктів та їх оцінок акумулюється в базі даних інформаційної технології, створюючи передумови для майбутнього покращення методів оцінювання (наприклад, автоматичне калібрування параметрів із застосуванням машинного навчання).

4.5. Методи, реалізовані в інформаційній технології

В інформаційній технології реалізовані наступні методи, розроблені в цій дисертаційній роботі (порядкові номери методів співпадають з їх нумерацією на рис. 4.1):

1. Метод поетапного оцінювання проєктів з розробки ПЗ, що включає попереднє, проміжне та детальне оцінювання (див. підрозділи 2.1-2.9) [4], [5], [16].

2. Метод побудови розкладу реалізації елементів оцінювання (див. підрозділ 2.10) [3].

3. Метод підтримки прийняття рішень щодо графіку реалізації проєкту та складу проєктної команди (див. підрозділ 2.11) [5], [16].

4. Метод побудови схеми слабоструктурованого бізнес-процесу (див. розділ 3) [1], [2], [6], [9], [10], [14].

4.6. Гіпотетичні сценарії реалізації інформаційної технології

Розглянемо два гіпотетичні сценарії реалізації інформаційної технології, ключовою відмінністю між якими є обрані технології програмування, що, з однієї сторони, вливає на вартість та час розробки, а з іншої забезпечує різний

ступінь гнучкості. Підкреслимо, що в цьому та наступних підрозділах поточного розділу йдеться про аналіз, проєктування архітектури та оцінювання програмного забезпечення інформаційної технології за винятком модуля процес-майнингу RTBPM-E, що описаний у розділі 3.

4.6.1. Сценарій 1: реалізація за підходом «pro-code»

Підхід «pro-code» передбачає традиційне програмування, коли переважна більшість вихідного коду системи пишеться інженерами-розробниками з нуля.

Цей сценарій передбачає використання наступних технологій програмування:

1) для розробки веб-інтерфейсу використати одну із таких технологій як Angular, ReactJS, NextJS, VueJS;

2) серверну частину реалізувати, використовуючи мікросервісну архітектуру та Microsoft .NET;

3) модуль підтримки прийняття рішень розробити на мові Python (оскільки бібліотеки, що реалізують математичні алгоритми, зокрема, розв'язування задач оптимізації, мають хорошу підтримку на Python);

4) в якості технології баз даних використати SQL-базу даних, наприклад, Microsoft SQL Server або PostgreSQL;

5) розгортання системи здійснити на одному з «хмарних» сервісів, наприклад, Microsoft Azure або AWS.

Очевидною перевагою цього набору технологій програмування є гнучкість, що дає змогу реалізовувати функційні та нефункційні вимоги високої складності. Головними недоліками є більша тривалість та вища вартість розробки у порівнянні із сценарієм 2.

4.6.2. Сценарій 2: реалізація за підходом «low-code»

Підхід «low-code» передбачає мінімізацію написання вихідного коду інженерами-розробниками; переважна частина функцій система реалізується за допомогою наборів готових компонент та конструкторів з графічним інтерфейсом.

Microsoft Power Platform, будучи є представником технологій «low-code», призначена для зниженням вартості та часу розробки програмного забезпечення. Сильною стороною Power Platform є її повна інтеграція з «екосистемою» Microsoft, зокрема, сервісом аутентифікації, Office 365, SharePoint та Teams. В разі використання Power Platform підхід до реалізації інформаційної технології буде наступний:

1) використати Power App Canvas для реалізації графічного веб-інтерфейсу;

2) в якості бази даних використати «хмарний» сервіс баз даних Dataverse;

3) реалізацію математичних методів, зокрема, методу підтримки прийняття рішень (див. підрозділ 2.11) здійснити зі застосуванням Python за межами Power Platform (наприклад, реалізувати як Azure Functions);

4) для аутентифікації та управління користувачами використати сервіси Microsoft, зокрема, Azure Entra ID (який раніше називався «Azure Active Directory»).

Очевидно, що Microsoft Power Platform поступається у гнучкості технологіям із сценарію 1. Однак, перевагою розробки із застосуванням Microsoft Power Platform є менші обсяги часу та ресурсів (у порівнянні із сценарієм 1).

4.7. Попереднє оцінювання інформаційної технології

Метою попереднього оцінювання ПЗ інформаційної технології є отримання прогнозу часу та ресурсів, необхідних для реалізації за сценаріями 1 та 2, а також, як результат порівняння попередніх оцінок, обрати один із цих сценаріїв як пріоритетний.

4.7.1. Хід попереднього оцінювання

Для попереднього оцінювання ПЗ інформаційної технології була застосована схема, описана у пункті 2.7.8. Після цього було застосовано розроблений метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту (див. підрозділ 2.11). В результаті порівняльного аналізу отриманих аль-

тернативних варіантів оцінок був обраний той з них, який, з точки зору автора дисертації, є кращим прогнозом часу та ресурсів, необхідних для розробки ПЗ інформаційної технології.

Для отримання попередньої оцінки ПЗ інформаційної технології виконані наступні кроки:

Крок 1. Побудова ІСЕО для попереднього оцінювання базується на функційних вимогах із підрозділу 4.4. Цей етап оцінювання не передбачає високого рівня деталізації ІСЕО – одного, а в окремих випадках двох рівнів ієрархії є цілком достатньо. Така «низькодеталізована» ІСЕО має ряд переваг, зокрема, мінімізація робочого часу, необхідного на її підготовку, а також можливість охопити весь зміст проєктних робіт, не надто вдаючись в деталі. Важливим аспектом побудови ІСЕО є визначення атрибутів елементів оцінювання таких як функційні вимоги, нефункційні вимоги, ризики тощо. ІСЕО, побудована для попереднього оцінювання, представлена у табл. Д-9.1 (додаток 9). Варто підкреслити, що ІСЕО побудована в врахуванням того, що вона буде деталізуватися на наступних етапах оцінювання.

Крок 2. Так як кількість елементів оцінювання в ІСЕО, підготованої на кроці 1, є відносно не великою, то для оцінювання трудоємності можна скористатися підходом групування за схожістю (див. пункт 2.7.5). Результати оцінювання трудоємності представлені у табл. 4.1. Більш детальну інформацію про оцінку трудоємності можна отримати з рис. Д-9.1 та табл. Д-9.2 (додаток 9).

Таблиця 4.1

Нормалізована оцінка розробки: попереднє оцінювання

Сценарій реалізації	***Базова НОР,*** *людино-години*	***Оптимістична НОР,*** *людино-години*	***Песимістична НОР,*** *людино-години*
Сценарій реалізації С1	2856.0	2140.9	4050.2
Сценарій реалізації С2	1712.0	1228.6	2518.6

Сценарій реалізації	***Базова НОР,*** *людино-години*	***Оптимістична НОР,*** *людино-години*	***Песимістична НОР,*** *людино-години*
Різниця: $(C1-C2)$	1144.0	912.3	1531.6
$\frac{C1-C2}{C1}\%$	40.06%	42.61%	37.82%

Крок 3. Для попереднього оцінювання сформуємо проєктну команду без диференціації спеціалізацій інженерів-розробників. Розміри цієї команди представлені у табл. 4.2. Більше деталей щодо складу проєктної команди можна знайти у табл. Д-9.3 (додаток 9).

Таблиця 4.2

Попередня оцінка складу проєктної команди

Сценарій реалізації	***Оптимістична оцінка***		***Песимістична оцінка***	
	ЕПЗ проєктної команди	*НЕПЗ інженерів-розробників*	*ЕПЗ проєктної команди*	*НЕПЗ інженерів-розробників*
Сценарій реалізації C1	18.00	10.04	22.00	11.74
Сценарій реалізації C2	13.50	6.14	17.50	8.34
Різниця: $C1-C2$	4.50	3.90	4.50	3.40
$\frac{C1-C2}{C1}\%$	25.00%	38.84%	20.45%	28.96%

Крок 4. Відштовхуючись від результатів, отриманих на кроках 1-3, отримуємо попередні оцінки тривалості проєкту, НСР команди, а також нормалізованої собівартості (див. пункти 2.7.3 та 2.7.4). Основні результати оцінювання представлено у табл. 4.3, більше деталей можна знайти у табл. Д-9.4 (додаток 9).

Таблиця 4.3

Попередні оцінки тривалості проєкту, нормалізованої спроможності розробки та нормалізованої собівартості

	Оптимістична оцінка			***Песимістична оцінка***		
	Тривалість проєкту, місяці	*НСР, людино-години*	*Норм. собівар-тість*	*Тривалість проєкту, місяці*	*НСР, людино-години*	*Норм. собівар-тість*
Сценарій реалізації С1	5.0	2219.4	10878.00	8.0	4152.3	20899.20
Сценарій реалізації С2	5.0	1357.3	8190.00	7.0	2581.0	14994.00
Різниця: С1 – С2	0.0	862.1	2688.00	1.0	1571.3	5905.20
$\frac{С1-С2}{С1}\%$	0.00%	38.84%	24.71%	12.50%	37.84%	28.26%

Крок 5. Для отримання альтернативного варіанту попередньої оцінки застосовано метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту (табл. 4.4 та 4.5, а також табл. Д-9.5 – Д.-9.9 у додатку 9).

Таблиця 4.4

Попередні оцінки тривалості проєкту, нормалізованої спроможності розробки та нормалізованої собівартості, отримані із застосуванням методу підтримки прийняття рішень

Сценарій реалізації	***Оптимістична оцінка***			***Песимістична оцінка***		
	Тривалість проєкту, місяці	*НСР, людино-години*	*Норм. собівар-тість*	*Тривалість проєкту, місяці*	*НСР, людино-години*	*Норм. собівар-тість*
Сценарій реалізації С1	6.0	2567.75	13129.20	8.0	4803.03	23419.20

Сценарій реалізації	***Оптимістична оцінка***			***Песимістична оцінка***		
	Тривалість проєкту, місяці	*НСР, людино-години*	*Норм. собівар-тість*	*Тривалість проєкту, місяці*	*НСР, людино-години*	*Норм. собівар-тість*
Сценарій реалізації С2	4.5	1527.92	8051.40	7.0	2995.71	15317.40
Різниця: С1 – С2	1.5	1039.83	5077.80	1.0	1807.32	8101.80
$\frac{С1-С2}{С1}\%$	25.00%	40.50%	38.68%	12.50%	37.63%	34.59%

Таблиця 4.5

Попередня оцінка складу проєктної команди,

отримана із застосуванням методу підтримки прийняття рішень

Сценарій реалізації	***Оптимістична оцінка***		***Песимістична оцінка***	
	ЕПЗ проєктної команди	*НЕПЗ інженерів-розробників*	*ЕПЗ проєктної команди*	*НЕПЗ інженерів-розробників*
Сценарій реалізації С1	17.50	9.68	23.75	13.58
Сценарій реалізації С2	14.25	7.68	17.50	9.68
Різниця: С1 – С2	3.25	2.00	6.25	3.9
$\frac{С1-С2}{С1}\%$	18.57%	20.66%	26.32%	28.72%

4.7.2. Результати попереднього оцінювання

У попередньому пункті представлені дві альтернативи попередніх оцінок для розробки ПЗ інформаційної технології, отриманих із застосуванням:

1) схеми попереднього оцінювання (див. пункт 2.7.8);

2) методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту (див. підрозділ 2.11).

Як можна бачити із табл. 4.2-4.5, оцінки тривалості реалізації проєкту, а також склад команд, отримані для кожного із сценаріїв реалізації не мають значної різниці, в той час як нормалізована собівартість, отримана із застосуванням методу підтримки прийняття рішень є навіть дещо вищою.

Для подальшого опрацювання для сценаріїв 1 та 2 обрано оцінки, отримані із застосуванням методу підтримки прийняття рішень, з наступної причини: не зважаючи на дещо вищу нормалізовану собівартість, ця оцінка передбачає більше значення НСР проєктної команди, даючи змогу закласти більший «запас міцності», що, в свою чергу, важливо, враховуючи високий рівень невідомого на етапі попереднього оцінювання.

4.7.3. Підсумки попереднього оцінювання

Як зазначено вище, для подальшого опрацювання обрано попередню оцінку, отриману із застосуванням методу підтримки прийняття рішень (табл. 4.4 та 4.5). В цьому пункті здійснено вибір сценарію реалізації інформаційної технології, керуючись наступними критеріями:

1) оцінка тривалості реалізації проєкту;

2) оцінка розміру проєктної команди;

3) оцінка нормалізованої собівартості розробки;

4) технічні ризики;

5) гнучкість технологій реалізації та перспективи розвитку.

Як видно з попереднього пункту, сценарій 1, який передбачає реалізацію системи з використанням технологій, що вимагають написання вихідного коду, має переваги над сценарієм 2, що базується на використанні Microsoft Power

Platform. В той час як за сценарієм 2 можна швидше отримати функціонуюче ПЗ, сценарій 1 забезпечує більше гнучкості в перспективі.

За результатами попереднього оцінювання прийнято наступні рішення:

1. Реалізацію MVP-версії (англ. «minimum viable product») ПЗ інформаційної технології здійснювати за сценарієм 1. Відповідно, наступні етапи оцінювання (проміжне та детальне), а також проєктування архітектури здійснюються згідно із сценарієм 1 (див. підрозділ 4.9). Підкреслимо, що власне реалізація MVP-версії не входить до завдань цієї дисертаційної роботи.

2. Підхід сценарію 2, зокрема, технологію Microsoft Power Platform, використати для реалізації прототипу (POC, англ. «proof of concept»), метою якого є демонстрація ключових функційних можливостей інформаційної технології та тестування методів, розроблених у цій дисертаційні роботі. Власне прототип і реалізований в рамках цієї дисертаційної роботи (див. підрозділ 4.8).

Таким чином, підсумки попереднього оцінювання ПЗ інформаційної технології базуються на оцінці сценарію реалізації 1 («pro-code»), отриманої із застосуванням методу підтримки прийняття рішень. Ключові характеристики підсумкової попередньої оцінки представлено у табл. 4.6. Ця оцінка передається для подальшого опрацювання на етап проміжного оцінювання (див. підрозділ 4.10).

Таблиця 4.6

Підсумки попереднього оцінювання

№	***Характеристика***	***Оптимістична оцінка***	***Песимістична оцінка***
1	Тривалість, місяці	6.0	8.0
2	ЕПЗ проєктної команди	17.50	23.75
3	НЕПЗ інженерів-розробників	9.68	13.58
4	НСР інженерів-розробників, людино-години	2567.75	4803.03
5	НОР змісту проєктних робіт, людино-години	2140.9	4050.2
6	Нормалізована собівартість	13129.20	23419.20

4.8. Прототип інформаційної технології

Оскільки реалізація повноцінної MVP-версії вимагає значних затрат часу та ресурсів (як можна бачити із результатів попереднього оцінювання, див. підрозділ 4.7), для демонстрації роботи розроблених методів реалізовано прототип. З метою зменшення тривалості та трудоємності розробки прототип реалізовано за сценарієм 2, який передбачає застосування Microsoft Power Platform як основної технології реалізації.

Оскільки метою прототипу інформаційної технології є демонстрації лише ключових можливостей, зокрема зроблених методів оцінювання, то наступні функційні вимоги не включені до змісту робіт прототипу:

1) деревовидне представлення ICEO;

2) оцінювання трудоємності групуванням за схожістю;

3) візуалізація розкладу реалізації елементів оцінювання;

4) зворотний зв’язок щодо оцінки;

5) бібліотека шаблонів;

6) безпека (за винятком стандартних можливостей, що надаються Microsoft Power Platform).

Графічний інтерфейс прототипу реалізовано із використанням Model-driven Power App, а для операційної бази даних використано Dataverse. Оскільки Microsoft Power Platform не дає змоги реалізувати метод підтримки прийняття рішень та метод побудови розкладу реалізації елементів оцінювання, то їх реалізовано на мові Python із використанням Azure Functions. Також реалізовано інтеграцію із прототипом модуля RTBPM-E, використовуючи Power Platform API для отримання event-даних. Компоненти реалізованого прототипу та взаємозв’язки між ними представлено на рис. 4.2.

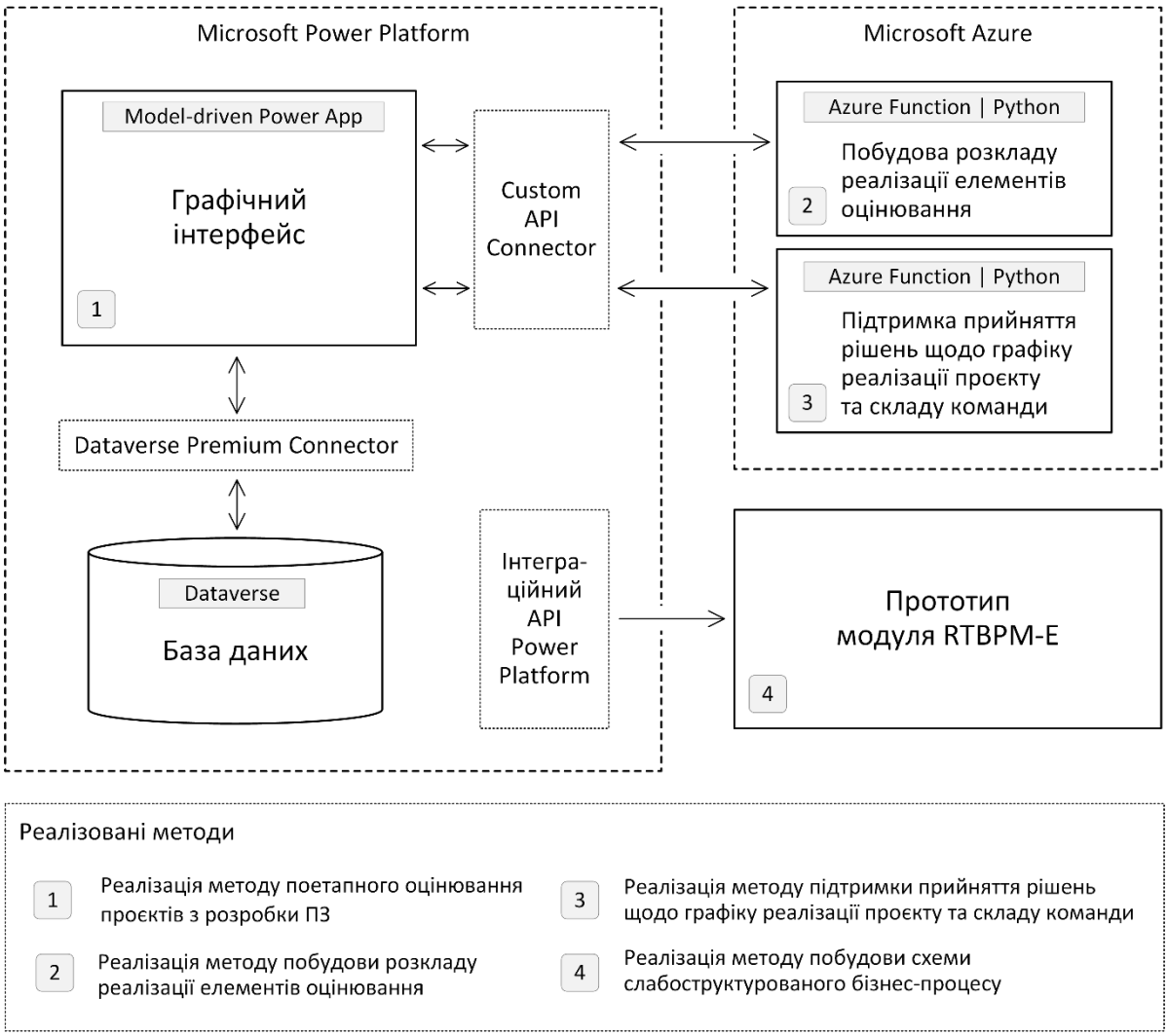


Рис. 4.2. Діаграма компонентів прототипу інформаційної технології

Перевагою реалізованого прототипу є змога тестування ключових функційних можливостей інформаційної технології, мінімізуючи час, затрачений на розробку. Перед початком реалізації MVP-версії інформаційної технології база даних прототипу буде трансформована в операційну базу даних інформаційної технології. Python-реалізації методів оцінювання, що розгорнуті на Azure Functions, будуть трансформовані у відповідні сервіси з мінімальними затратами на їх модифікацію.

4.9. Архітектура MVP-версії інформаційної технології

MVP-версія інформаційної технології є логічним продовженням реалізованого прототипу. За результатами попереднього оцінювання сценарій 1 було

обрано для реалізації MVP-версії. Кінцевою метою MVP-версії є її впровадження для використання на виробничому середовищі організації, яка займається розробкою ПЗ.

Ключовими відмінностями архітектури MVP-версії від архітектури прототипу є технології реалізації графічного інтерфейсу та використання мікросервісної архітектури на серверній стороні, що вимагає дещо більше робочого часу на розробку, але дає значно більше гнучкості та залишає хороші перспективи розвитку цього програмного продукту.

Як можна бачити з діаграми компонентів MVP-версії інформаційної технології (рис. 4.3), дизайн її архітектури передбачає наступні три рівні: графічний веб-інтерфейс, сервіси (кожен з яких реалізує логічно близькі функції та є відносно незалежним від інших) та рівень бази даних. Окремо варто відзначити інтеграцію із зовнішніми сервісами такими як сервіс безпеки організації та модуль RTBPM-E.

Вибір технологій реалізації здійснено за критеріями, аналогічними тим, які були використані для модуля RTBPM-E (табл. 4.7).

Таблиця 4.7

Технології реалізації програмного забезпечення
MVP-версії інформаційної технології

№	*Компонент*	*Рекомендовані технології реалізації*	*Альтернативні технології реалізації*
1	Графічний інтерфейс	ReactJS	Angular, NextJS, VueJS
2	Сервіси (на стороні сервера)	Python Fast API	ASP.NET, Java SpringBoot, NodeJS
3	Реалізація методів 1-3	Python	–
4	Операційна база даних	PostgreSQL	MySQL, MS SQL Server

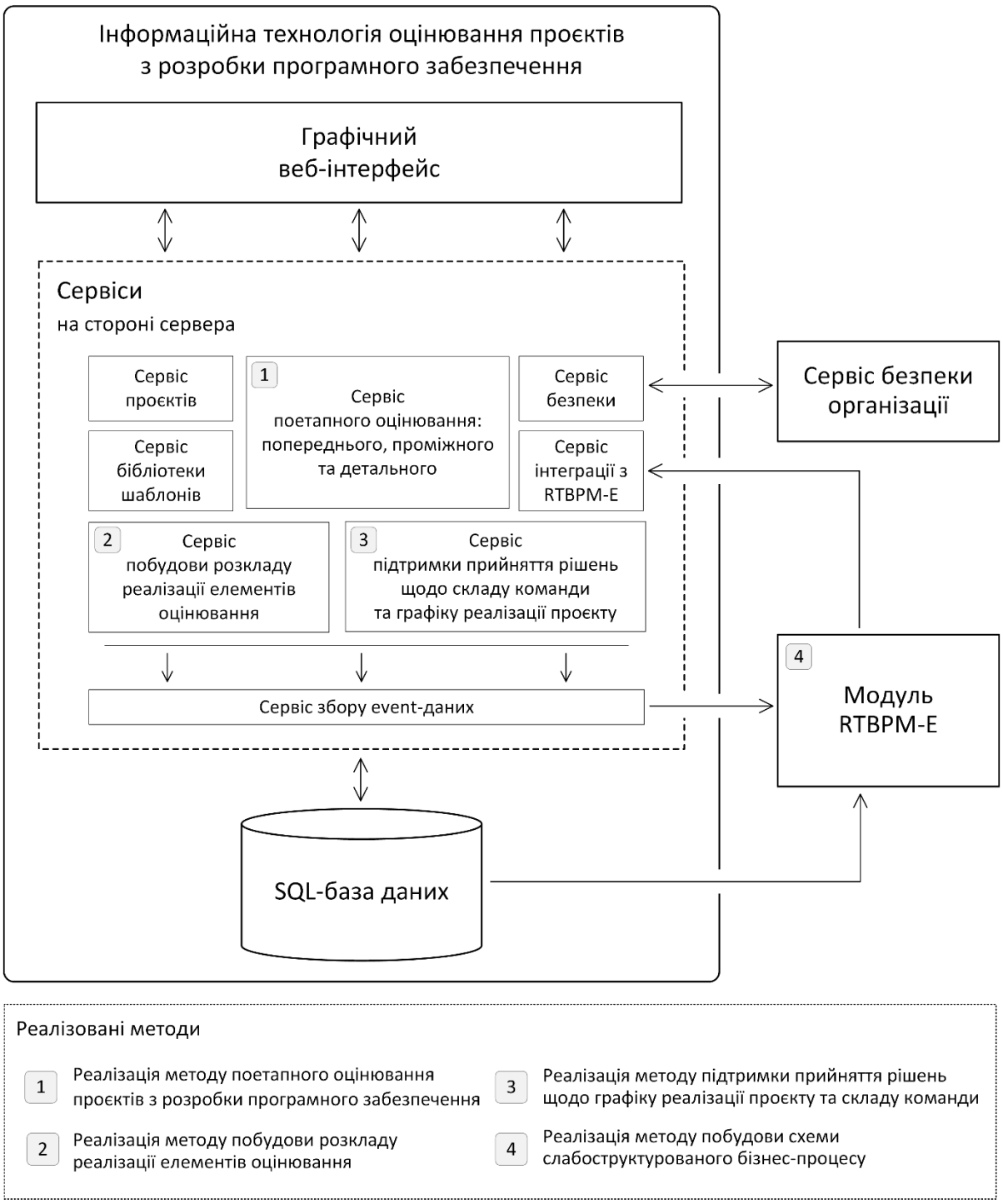


Рис. 4.3. Діаграма компонентів MVP-версії інформаційної технології

Наступний етап оцінювання, проміжне оцінювання, буде проведено для сценарію реалізації 1, відштовхуючись від дизайну архітектури MVP-версії інформаційної технології.

4.10. Проміжне оцінювання

Метою проміжного оцінювання є отримання оцінки для сценарію реалізації 1 (підхід «pro-code») інформаційної технології. Очікується, що результати проміжного оцінювання будуть точнішими та детальнішими за результати попереднього оцінювання, оскільки на цьому етапі більше відомо про інформаційну технологію, зокрема, спроєктовано її архітектуру (див. підрозділ 4.9) та деталізовано ІСЕО (табл. Д-10.1 у додатку 10).

4.10.1. Хід проміжного оцінювання

Проміжне оцінювання інформаційної технології базується на поєднанні схеми проміжного оцінювання (див. пункт 2.8.5) та методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту (див. підрозділ 2.11). Для отримання проміжної оцінки виконано наступні кроки:

Крок 1. Підвищити рівень деталізації ІСЕО, сформованої на етапі попереднього оцінювання (табл. Д-9.1 у додатку 9). Як видно із табл. 4.8, ІСЕО для проміжного оцінювання містить приблизно у 2.5 рази більше елементів ніж ІСЕО для попереднього оцінювання.

Таблиця 4.8

Порівняння ієрархічних структур елементів оцінювання
для попереднього та проміжного оцінювань

Етап оцінювання	***Максимальна кількість рівнів ієрархії***	***Кількість елементів оцінювання***	***Кількість композитних елементів оцінювання***	***Кількість простих елементів оцінювання***
Попереднє оцінювання	2	26	3	23
Проміжне оцінювання	2	65	12	53

Крок 2. Надати оптимістичну та песимістичну НОР для елементів оцінювання. Для надання НОР використано 3-точкову оцінку PERT (табл. Д-10.2 у додатку 10). Як можна бачити із табл. 4.9, сумарна НОР для проміжного оцінювання має «вужчий» інтервал у порівнянні із НОР для попередньої оцінки.

Таблиця 4.9

Порівняння нормалізованих оцінок розробки
для попереднього та проміжного оцінювань

Етап оцінювання	***Оптимістична НОР,*** *людино-години*	***Базова / найбільш ймовірна НОР[1],*** *людино-години*	***Песимістична НОР,*** *людино-години*	***Інтервал НОР***
Попереднє оцінювання	2140.9	2856.0	4050.2	-25,1%…+41,9%
Проміжне оцінювання	2047.0	2445.0	2994.0	-16,3%…+22,5%

Крок 3. Визначити шаблони графіку реалізації проєкту та складу команди. Для проміжного оцінювання використано команду з диференційованими спеціалізаціями, що включає інженерів-розробників: «back end», «front end», та «data science» (табл. Д-10.3 у додатку 10). В свою чергу, шаблон графіку реалізації проєкту передбачає поділ на наступні 6 фаз: початок проєкту, масштабування команди, основна фаза розробки, стабілізація, тестування користувачами, запуск в експлуатацію (табл. Д-10.4 у додатку 10). При цьому фаза 3, основна фаза реалізації проєкту, має змінну тривалість (що дає змогу застосовувати метод підтримки прийняття рішень на кроці 5).

[1] Для попереднього оцінювання використана базова НОР, а для проміжного оцінювання – найбільш ймовірна НОР, що відповідає «most likely» оцінці методу PERT.

Крок 4. Оцінити графік реалізації проєкту та склад проєктної команди, користуючись відповідними співвідношеннями для проміжного оцінювання (див. підрозділ 2.8). Ключові характеристики цієї проміжної оцінки представлено у табл. 4.10. Деталізацію проміжної оцінки надано у табл. Д-10.5 – Д-10.7 (додаток 10).

Таблиця 4.10

Підсумки проміжної оцінки

№	***Характеристика***	***Оптимістична оцінка***	***Песимістична оцінка***
1	Тривалість, спринти	12	16
2	Тривалість, години	912	1216
3	Тривалість, місяці	5.5	7.3
4	Проєктний робочий час команди, людино-години	13262.00	18430.00
5	Проєктний робочий час «нерозробницьких» ролей, людино-години	6156.00	8436.00
6	Проєктний робочий час інженерів-розробників, людино-години	7106.00	9994.00
7	НСР «back end»-розробників, людино-години	813.18	1295.97
8	НСР «front end»-розробників, людино-години	1248.35	1841.17
9	НСР «data science»-розробників, людино-години	212.45	304.33
10	Нормалізована собівартість	10267.60	14060.00

Крок 5. Оцінити графік реалізації проєкту та склад проєктної команди із застосуванням методу підтримки прийняття рішень (див. підрозділ 2.11). Для застосування цього методу використано Python-скрипт, вихідний код якого в повному обсязі надано у додатку 12. Ключові характеристики проміжної оцінки, отриманої із застосуванням методу підтримки прийняття рішень, представлено у табл. 4.11. Деталізацію цієї оцінки надано у табл. Д-10.8 – Д-10.11 (додаток 10).

Таблиця 4.11

Підсумки проміжної оцінки, отриманої із застосуванням методу підтримки прийняття рішень

№	*Характеристика*	*Оптимістична оцінка*	*Песимістична оцінка*
1	Тривалість, спринти	12	15
2	Тривалість, години	912	1140
3	Тривалість, місяці	5.5	6.8
4	Проєктний робочий час команди, людино-години	12027.00	15751.00
5	Проєктний робочий час «нерозробницьких» ролей, людино-години	5586.00	7182.00
6	Проєктний робочий час інженерів-розробників, людино-години	6441.00	8569.00
7	НСР «back end»-розробників, людино-години	791.03	1153.12
8	НСР «front end»-розробників, людино-години	1122.36	1610.15
9	НСР «data science»-розробників, людино-години	192.39	284.55
10	Нормалізована собівартість	9440.15	12299.65

4.10.2. Підсумки проміжного оцінювання

У попередньому пункті отримано два варіанти проміжної оцінки: перший із застосуванням схеми із пункту 2.8.5, а другий, використовуючи метод підтримки прийняття рішень із підрозділу 2.11. Як можна бачити зі табл. 4.10 та 4.11 оцінка тривалості проєкту в обох випадках є приблизно однаковою, однак нормалізована собівартість є дещо нижчою для другого варіанту. Отже, для подальшого опрацювання на етапі детального оцінювання (див. підрозділ 4.11) обираємо проміжну оцінку, отриману із застосуванням методу підтримки прийняття рішень, підсумки якої представлено у табл. 4.11, а деталі – у табл. Д-10.9 – Д-10.11 (додаток 10).

4.11. Детальне оцінювання

Метою детального оцінювання є отримання оцінки, точність та детальність якої дають змогу взяти зобов'язання щодо очікуваних результатів реалізації проєкту. Передбачається, що в разі початку реалізації, ця оцінка буде передана проєктній команді для подальшого опрацювання.

4.11.1. Хід детального оцінювання

Хід детального оцінювання визначений схемою з пункту 2.9.2, яку умовно можна розділити на 2 частини: деталізацію результатів проміжного оцінювання (кроки 1-5) та побудову розкладу реалізації елементів оцінювання (крок 6).

Безперечно, результати проміжного оцінювання, зокрема ICEO, потребують деталізації. Оскільки ця деталізація передбачає використання методів близьких до тих, які застосовувалися на етапі проміжного оцінювання, то з метою скорочення викладок відповідні кроки детального оцінювання пропущено.

Побудову нормалізованого розкладу реалізації елементів оцінювання здійснено із застосуванням методу із підрозділу 2.10. Отримані оптимістичний та песимістичний нормалізовані розклади реалізації елементів оцінювання представлено у додатку 11.

4.11.2. Підсумки детального оцінювання

В якості підсумкової оцінки ПЗ інформаційної технології візьмемо песимістичну оцінку, згідно з якою тривалість реалізації MVP-версії складає 15 спринтів (або 6.8 місяця).

Таким чином, підсумкова оцінка включає наступні складові:

1. Зміст проєктних робіт у вигляді ICEO (табл. Д-10.1 у додатку 10).

2. НОР (песимістична) змісту проєктних робіт (табл. Д-10.2 та табл. Д-11.1 у додатках 10 та 11 відповідно).

3. Оцінка (песимістична) складу проєктної команди (табл. Д-10.9 у додатку 10).

4. Оцінки компонентів структури робочого часу та НСР (табл. Д-10.11 у додатку 10).

5. Нормалізований розклад реалізації елементів оцінювання (табл. Д-11.3 у додатку 11).

Вважатимемо, що отримана оцінка може бути використана для взяття зобов'язань щодо очікуваних результатів та термінів реалізації MVP-версії інформаційної технології. Згідно з життєвим циклом оцінювання (див. підрозділ 2.1) ця оцінка передається команді, що здійснюватиме реалізацію проєкту. Команда, в свою чергу, деталізуватиме оцінку до рівня, необхідного для етапу реалізації.

4.12. Перспективи розвитку інформаційної технології як програмного продукту

Вище в цьому розділі представлено архітектуру POC- та MVP-версій ПЗ інформаційної технології оцінювання проєктів з розробки ПЗ. При цьому MVP-версія включає лише основні функційні можливості, необхідні для початку її експлуатації у виробничих умовах.

Вбачаються наступні кроки розвитку інформаційної технології після реалізації MVP:

1. Реалізація функційних можливостей для підтримки початкового оцінювання (етап, що передує попередньому оцінюванню, див. підрозділ 2.1 та рис. 2.1). Зокрема, після накопичення достатнього обсягу даних у базі даних інформаційної технології, стає можливим застосування методів оцінювання за аналогією.

2. Розробка методів та функційних можливостей для оцінювання на етапі реалізації проєкту, включаючи інтеграцію з системами управління проєктними задачами, наприклад: Microsoft Azure DevOps, Atlassian Jira тощо.

3. Покращення методів оцінювання за рахунок машинного навчання та ШІ, що опираються на акумульовані дані.

4. Інтеграція генеративного ШІ. При цьому, найбільше ймовірно, генеративний ШІ матиме двоякий вплив: з однієї сторони, він вплине на процеси розробки ПЗ, що потребуватиме відповідного відображення у методах оцінювання, а з іншої, власне оцінювання може базуватися на застосуванні генеративного ШІ.

5. Реалізація SaaS-продукту (англ. «system as a service»), що буде доступним для організацій-користувачів за ліцензійною підпискою.

6. Інтеграція існуючих наборів даних, призначених для оцінювання проєктів з розробки ПЗ [64], [65]. Використання вже існуючих наборів даних відкриє можливості для використання машинного навчання та ШІ. Така інтеграція потребуватиме вирішення ряду задач, зокрема: визначення якості даних, аналіз ліцензій, за якими поширюються ці дані, а також їх регулярне обновлення.

Варто підкреслити, що реалізація цих функцій вимагає достатнього рівня гнучкості, що забезпечується обраним сценарієм реалізації 1. При цьому використання Microsoft Power Platform, як передбачається сценарієм 2, для реалізації описаних вище перспективних можливостей інформаційної технології приховує значні технічні ризики.

Висновки до розділу 4

В цьому розділі представлено інформаційну технологію оцінювання проєктів з розробки ПЗ.

Продемонстровано застосування методів, розроблених у цій дисертаційній роботі, до оцінювання ПЗ інформаційної технології. Оцінювання здійснювалося у три етапи. Спершу, відштовхуючись від функційних вимог, було зроблено попередню оцінку для двох потенційних сценаріїв реалізації інформаційної технології. За результатами попереднього оцінювання, а також враховуючи перспективи розвитку інформаційної технології, було прийнято рішення реалізовувати ПЗ інформаційної технології за сценарієм «pro-code». Для цього сценарію реалізації були проведені проміжне та детальне оцінювання. Під час

поетапного оцінювання було продемонстровано, як поглиблюється розуміння вимог та деталізується архітектура інформаційної технології.

Однак, прототип інформаційної технології, ключовим завданням якого є демонстрація можливостей розроблених методів оцінювання, розроблено із застосуванням Microsoft Power Platform, що дало змогу скоротити час розробки.

Окреслено перспективи розвитку інформаційної технології як програмного продукту, ключовими з яких є розширення підтримуваних методів оцінювання як на етап раннього знайомства з ідеєю проєкту (початкове оцінювання), так і на етап реалізації проєкту. З точки зору майбутньої монетизації інформаційної технології важливою є задача реалізація SaaS-рішення, яке доступно індивідуальним та корпоративним користувачам за ліцензійною підпискою. Також варто підкреслити, що у найближчому майбутньому сфера розробки ПЗ може зазнати суттєвих змін завдяки широкому використанню генеративного ШІ, що, безумовно, необхідно враховувати з точки зору перспектив розвитку інформаційної технології.

ЗАГАЛЬНІ ВИСНОВКИ ДО РОБОТИ

В цій дисертаційні роботі вирішено актуальну наукову задачу з розробки методів оцінювання проєктів з розробки ПЗ та інформаційної технології, що забезпечує застосування розроблених методів на практиці.

У першому розділі роботи конкретизовано поняття оцінки проєкту з розробки ПЗ, надано перелік складових цієї оцінки, а також сформульовано визначення точності та надійності оцінки. Існуючі методи оцінювання класифіковано за чотирьома категоріями, кожна з яких відповідає етапу їх еволюції, які, в свою чергу, відображають основні тенденції у розробці ПЗ того періоду: «класичні» методи (50-80-ті роки, процедурне програмування та водоспадна модель життєвого циклу), «класичні» методи другого покоління (друга половина 80-х та початок 2000-х років, об'єктно-орієнтоване програмування та спіральна модель життєвого циклу), agile-методи оцінювання (починаючи з кінця 90-х років і до сьогодні) та методи із застосуванням машинного навчання та ШІ (з першої половини 2000-х та до сьогодні).

За результатами проведеного аналізу існуючих методів оцінювання, а також, відштовхуючись від практичного виробничого досвіду автора дисертації в оцінюванні проєктів з розробки ПЗ, ідентифіковано перелік проблем, ключовою з яких є відсутність методів оцінювання та відповідних програмних засобів, які б послідовно охоплювали всі етапи життєвого циклу розробки ПЗ, починаючи від зародження ідеї та завершуючи впровадженням в експлуатацію. Наслідком цієї проблеми, зокрема, є так звані «одноразові оцінки», недостатність зворотного зв'язку щодо оцінки від команди, що здійснює реалізацію проєкту, до експертів, які надавали оцінку, відсутність накопичених даних про минулі проєкти (включаючи як оцінки, так і фактично затрачені на розробку ресурси).

Для вирішення сформульованих проблем розроблено схему поетапного оцінювання, що комплементарна етапам життєвого циклу проєкту з розробки ПЗ. Розроблена схема є відображенням ідеї «конусу невизначеності», коли точність та рівень деталізації оцінки поступово зростає по мірі поглиблення розуміння

вимог та архітектури проєкту. Розроблено теоретичні основи методів оцінювання, ключовими аспектами яких є структура робочого часу інженера-розробника, ієрархічна структура елементів оцінювання, підхід «нормалізації» до оцінювання трудоємності, система рівнянь балансу робочого часу проєктної команди.

Відштовхуючись від розроблених теоретичних основ, вперше розроблено метод поетапного оцінювання, що включає попереднє, проміжне та детальне оцінювання, які охоплюють етапи аналізу та проєктування, що передують початку реалізації проєкту. Розроблено метод побудови розкладу реалізації елементів оцінювання, що є подальшим розвитком методу List Scheduling. Розроблений метод базується на відповідності нормалізованої оцінки розробки проєктних задач нормалізованій спроможності розробки проєктної команди. Цей метод найбільш доцільно застосовувати на етапі детального оцінювання, коли до оцінки проєкту необхідно включити, в якій послідовності реалізовуватимуться проєктні задачі. Розроблено метод підтримки прийняття рішень, що є комплементарним до методу поетапного оцінювання та дає змогу зробити підготовку оцінки більш ефектною, пропонуючи варіанти графіку реалізації проєкту та складу команди за результатами розв’язання задач цілочисельного програмування із наступним ранжуванням альтернатив, застосовуючи метод аналізу ієрархій.

Якщо організація системно займається розробкою ПЗ (для власних потреб або на замовлення), то оцінювання цих проєктів стає бізнес-процесом. Типовою помилкою, що зустрічається на практиці, є підхід до бізнес-процесу оцінювання як до структурованого бізнес-процесу, який передбачає чітко фіксовану послідовність кроків та добре прогнозовані результати. В дисертаційні роботі обґрунтовано, що при застосуванні у виробничому середовищі організації, оцінювання є слабоструктурованим бізнес-процесом, який надає певний ступінь гнучкості, що створює умови для застосування знань, досвіду та творчих підходів учасниками команди оцінювання, що, в свою чергу, сприяє отриманню унікальних конкурентоспроможних результатів. Розроблено концептуальну схему бізнес-процесу оцінювання, що описує його основні активності, приблизний порядок їх

виконання, а також окреслює ключові сфери відповідальності учасників. Для побудови актуальної схеми слабоструктурованого бізнес-процесу оцінювання розроблено метод, який є подальшим розвитком методу процес-майнингу, відомого під назвою Fuzzy Miner. Для програмної реалізації цього методу спроєктовано модуль RTBPM-E, для якого передбачена інтеграція з основною частиною інформаційної технології.

Необхідність розробки інформаційної технології для оцінювання проєктів з розробки ПЗ зумовлена кількома причинами. Перш за все, це можливість реалізації методів підтримки прийняття рішень та побудови розкладу реалізації задач проєкту. Іншою, не менш вагомою, причиною є необхідність накопичення даних про минулі проєкти та їх оцінки (для чого добре підходить SQL-база даних). В рамках дисертаційної роботи проаналізовано вимоги до інформаційної технології та спроєктовано її архітектуру. Продемонстровано застосування попереднього, проміжного та детального оцінювань до оцінювання ПЗ цієї інформаційної технології. Оскільки, згідно отриманих оцінок, розробка MVP-версії інформаційної технології, готової до впровадження у виробничу експлуатацію, вимагає обсягів часу та ресурсів, що виходять за рамки цієї дисертаційної роботи, було реалізовано прототип інформаційної технології, що демонструє основні функційні можливості та роботу розроблених методів. Варто підкреслити, що MVP-версія інформаційної технології є лише початковою версією програмного продукту, серед головних, на думку автора дисертації, перспектив розвитку якого є розробка та реалізація методів оцінювання для етапу реалізації проєкту, застосування можливостей генеративного ШІ, а також розробка SaaS версії, що буде доступна користувачам за ліцензійною підпискою.

Підсумовуючи результати, отримані під час виконання дисертаційної роботи, можна стверджувати, що всі наукові та технічні завдання дисертаційної роботи виконані в повній мірі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] A. Ye. Batyuk and V. V. Voityshyn, “Process Mining: Applied Discipline and Software Implementations,” *KPI Science News*, no. 5, pp. 22–36, Nov. 2018, doi: 10.20535/1810-0546.2018.5.146178.

[2] A. Batyuk and V. Voityshyn, “Distributed software system with web interface for automated business process discovery,” *Bulletin of Lviv Polytechnic National University: Information Systems and Networks*, vol. 5, pp. 70–77, Jun. 2019, doi: 10.23939/sisn2019.01.070.

[3] В. Теслюк, А. Батюк, and В. Войтишин, “Метод побудови нормалізованого розкладу реалізації задач проєкту з розробки програмного забезпечення для scrum-команди без диференціації спеціалізацій,” *Український журнал інформаційних технологій*, vol. 6, no. 2, pp. 11–19, 2024, doi: 10.23939/ujit2024.02.011.

[4] V. Teslyuk, A. Batyuk, and V. Voityshyn, “Method of Software Development Project Duration Estimation for Scrum Teams with Differentiated Specializations,” *Systems*, vol. 123, no. 10, pp. 1–19, Aug. 2022, doi: 10.3390/systems10040123.

[5] V. Teslyuk, A. Batyuk, and V. Voityshyn, “Preliminary Estimation for Software Development Projects Empowered with a Method of Recommending Optimal Duration and Team Composition,” *Appl. Syst. Innov.*, vol. 7, no. 3, pp. 1–21, Apr. 2024, doi: 10.3390/asi7030034.

[6] A. Batyuk and V. Voityshyn, “Software Architecture Design of the Information Technology for Real-Time Business Process Monitoring,” *ECONTECHMOD*, vol. 7, no. 3, pp. 13–22, 2018.

[7] A. Batyuk and V. Voityshyn, “Apache storm based on topology for real-time processing of streaming data from social networks,” in *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP’2016)*, Lviv, Ukraine, Aug. 2016, pp. 345–349. doi: 10.1109/DSMP.2016.7583573.

[8] A. Batyuk and V. Voityshyn, “Business Processes Monitoring by Means of Real-Time Visual Dashboards,” in *2017 6th International Academic Conference on Information, Communication, Society 2017 (ICS’2017)*, Slavske, Lviv, Ukraine, May 2017, pp. 204–205.

[9] A. Batyuk, V. Voityshyn, and V. Verhun, “Software Architecture Design of the Real- Time Processes Monitoring Platform,” in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP’2018)*, Lviv, Aug. 2018, pp. 98–101. doi: 10.1109/DSMP.2018.8478589.

[10] A. Batyuk and V. Voityshyn, “Streaming Process Discovery for Lambda Architecture-Based Process Monitoring Platform,” in *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT’2018)*, Lviv, Sep. 2018, pp. 298–301. doi: 10.1109/STC-CSIT.2018.8526592.

[11] A. Batyuk and V. Voityshyn, “Real-Time Process Monitoring Platform: Technical Implementation,” in *The 7th International Academic Conference on Information, Communication, Society 2018 (ICS’2018)*, Chynadiyovo, Lviv, Ukraine, May 2018, pp. 275–276.

[12] A. Batyuk and V. Voityshyn, “Real-Time Process Monitoring Platform Based on Streaming Process Discovery Techniques,” in *2018 XIV International Scientific Conference on Intellectual Systems of Decision-Making and Problems of Computational Intelligence (ISDMCI’2018)*, Zaliznyj Port, Kherson, Ukraine, May 2018, pp. 7–8.

[13] A. Batyuk and V. Voityshyn, “Cloud-based Architecture of the Business Process Monitoring Information Technology,” in *The 8th International Academic Conference on Information, Communication, Society 2019 (ICS’2019)*, Chynadiyovo, Lviv, Ukraine, May 2019, pp. 17–18.

[14] A. Batyuk and V. Voityshyn, “Streaming Process Discovery Method for Semi-Structured Business Processes,” in *2020 IEEE Third International Conference*

on Data Stream Mining & Processing (DSMP'2020), Lviv, Ukraine, Aug. 2020, pp. 444–448. doi: 10.1109/DSMP47368.2020.9204201.

[15] A. Batyuk and V. Voityshyn, "Process mining-based information technology for operational support of software projects estimation," in *XVI International Scientific Conference on Intellectual Systems of Decision-Making and Problems of Computational Intelligence (ISDMCI'2020)*, Zaliznyj Port, Kherson, Ukraine, 2020, pp. 9–11.

[16] V. Teslyuk, A. Batyuk, and V. Voityshyn, "Method of Recommending a Scrum Team Composition for Intermediate Estimation of Software Development Projects," in *2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT'2022)*, Lviv, Ukraine, Nov. 2022, pp. 373–376. doi: 10.1109/CSIT56902.2022.10000432.

[17] В. Рач and О. Пилипенко, "Кількісні методи ресурсно-часового оцінювання в гнучкій методології розробки програмного забезпечення," *Управління проектами та розвиток виробництва*, vol. 1, no. 61, pp. 62–71, 2017.

[18] Т. О. Говорущенко and О. В. Поморова, "Метод оцінки достатності інформації для визначення складності та якості програмного забезпечення на основі порівняльного аналізу онтологій," *Радіоелектронні і комп'ютерні системи*, vol. 6, no. 80, pp. 59–68, 2016.

[19] International Institute of Business Analysis, *Babok v3: a guide to business analysis body of knowledge*, Version 3. Toronto: IIBA, 2015.

[20] S. Grimstad, M. Jørgensen, and K. Moløkken-Østvold, "Software effort estimation terminology: The tower of Babel," *Information and Software Technology*, vol. 48, no. 4, pp. 302–310, Apr. 2006, doi: 10.1016/j.infsof.2005.04.004.

[21] *ISO 5725-1:2023 Accuracy (trueness and precision) of measurement methods and results — Part 1: General principles and definitions*, ISO 5725-1:2023,

Switzerland., 2023. Accessed: Mar. 30, 2025. [Online]. Available: https://www.iso.org/standard/69418.html

[22] Bureau of Naval Weapons, United States, Special Projects Office, “Program Evaluation Research Task PERT Summary Report: Phase 1,” Special Projects Office, Bureau of Naval Weapons, Department of the Navy, Washington, D.C., Jul. 1958.

[23] Bureau of Naval Weapons, United States, Special Projects Office, “Program Evaluation Research Task PERT Summary Report: Phase 2,” Special Projects Office, Bureau of Naval Weapons, Department of the Navy, Washington, D.C., Sep. 1958.

[24] B. W. Boehm, *Software Engineering Economics*. NJ, USA: Prentice-Hall: Saddle River, 1981.

[25] J. E. Kelley and M. R. Walker, “Critical-Path Planning and Scheduling,” in *Eastern Joint IRE-AIEE-ACM Computer Conference*, Association for Computing Machinery, NY, USA, 1959, pp. 160–173. doi: 10.1145/1460299.1460318.

[26] A. J. Albrecht, “Measuring Application Development Productivity,” in *Proceedings of IBM Applications Development Symposium*, IBM Corporation, White Plains, NY, USA, 1979, p. Monterey.

[27] A. J. Albrecht and J. E. Gaffney, “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation,” *IIEEE Trans. Software Eng.*, vol. SE-9, no. 6, pp. 639–648, Nov. 1983, doi: 10.1109/TSE.1983.235271.

[28] B. W. Boehm *et al.*, *Software Cost Estimation with COCOMO II*. NJ, USA: Prentice-Hall: Saddle River, 2000.

[29] G. Karner, “Resource Estimation for Objectory Projects.” Objective Systems SF AB, 1993. [Online]. Available:

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.604.7842&rep=rep1&type=pdf

[30] S. W. Munialo and G. M. Muketha, “A Review of Agile Software Effort Estimation Methods,” *IJCATR*, vol. 5, no. 9, pp. 612–618, Sep. 2016, doi: 10.7753/IJCATR0509.1009.

[31] M. Vyas, A. Bohra, D. C. S. Lamba, and A. Vyas, “A Review on Software Cost and Effort Estimation Techniques for Agile Development Process,” *Int J Recent Res Aspects*, vol. 5, no. 1, pp. 1–5, 2018.

[32] A. Trendowicz, J. Münch, and R. Jeffery, “State of the Practice in Software Effort Estimation: A Survey and Literature Review,” in *Software Engineering Techniques*, vol. 4980, Z. Huzar, R. Koci, B. Meyer, B. Walter, and J. Zendulka, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 232–245. doi: 10.1007/978-3-642-22386-0_18.

[33] R. D. Stutzke, *Estimating software-intensive systems: projects, products, and processes*. in SEI series in software engineering. Upper Saddle River, NJ: Addison Wesley, 2005.

[34] M. Azzeh, A. Bou Nassif, and I. B. Attili, “Predicting software effort from use case points: A systematic review,” *Science of Computer Programming*, vol. 204, p. 102596, Apr. 2021, doi: 10.1016/j.scico.2020.102596.

[35] B. Boehm, C. Abts, and S. Chulani, “Software development cost estimation approaches — A survey,” *Annals of Software Engineering*, vol. 10, no. 1, pp. 177–205, Nov. 2000, doi: 10.1023/A:1018991717352.

[36] A. Saeed, W. H. Butt, F. Kazmi, and M. Arif, “Survey of Software Development Effort Estimation Techniques,” in *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, in ICSCA 2018. New York, NY, USA: Association for Computing Machinery, Feb. 2018, pp. 82–86. doi: 10.1145/3185089.3185140.

[37] A. Trendowicz and R. Jeffery, *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-03629-8.

[38] D. Trietsch and K. R. Baker, "PERT 21: Fitting PERT/CPM for use in the 21st century," *International Journal of Project Management*, vol. 30, no. 4, pp. 490–502, May 2012, doi: 10.1016/j.ijproman.2011.09.004.

[39] A. A. B. Pritsker, "GERT: Graphical Evaluation and Review Technique." RM-4973-NASA. National Aeronautics and Space Administration under Contract No. NASr-21., 1966.

[40] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IIEEE Trans. Software Eng.*, vol. SE-4, no. 4, pp. 345–361, Jul. 1978, doi: 10.1109/TSE.1978.231521.

[41] L. H. Putnam and W. Myers, *Measures for Excellence: Reliable Software on Time, Within Budget*. Yourdon Press, 1992.

[42] B. W. Boehm, "Software Engineering Economics," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 1, pp. 4–21, 1984, doi: 10.1109/TSE.1984.5010193.

[43] J. W. Grenning, "Planning Poker or How to Avoid Analysis Paralysis While Release Planning." 2002. [Online]. Available: https://wingman-sw.com/articles/planning-poker

[44] *ISO/IEC 20926:2009 Software and systems engineering — Software measurement — IFPUG functional size measurement method 2009*, ISO/IEC 20926:2009, 2009. Accessed: Apr. 06, 2025. [Online]. Available: https://www.iso.org/standard/51717.html

[45] C. Tichenor, "A new software metric to complement function points: The software non-functional assessment process (SNAP)," *CrossTalk Journal*, vol. 26, pp. 21–26, 2013.

[46] “Project Estimation using Use Case Metrics Tutorial in Enterprise Architect | Sparx Systems.” Accessed: Nov. 23, 2023. [Online]. Available: https://sparxsystems.com/resources/tutorials/use-case-metrics.html

[47] A. V. Hrechko and A. O. Melashchenko, “Application of the Functional Size Measurement Method on the Example of the National Register of Ukraine,” *Cybern Syst Anal*, vol. 55, no. 2, pp. 298–305, Mar. 2019, doi: 10.1007/s10559-019-00134-7.

[48] R. D. Banker, R. J. Kauffman, and R. Kumar, “An Empirical test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment,” *Journal of Management Information Systems*, vol. 8, no. 3, pp. 127–150, Dec. 1991, doi: 10.1080/07421222.1991.11517933.

[49] R. Valerdi and B. W. Boehm, “COSYSMO: A Systems Engineering Cost Model,” *Génie logiciel*, 2010, Accessed: Feb. 26, 2025. [Online]. Available: https://dspace.mit.edu/handle/1721.1/83521

[50] K. Beck, *Extreme Programming Eplained: Embrace Change*. Reading, MA: Addison-Wesley, 2000.

[51] “Manifesto for Agile Software Development.” Accessed: Jun. 23, 2024. [Online]. Available: https://agilemanifesto.org/

[52] R. K. Mallidi and M. Sharma, “Study on Agile Story Point Estimation Techniques and Challenges,” *IJCA*, vol. 174, no. 13, pp. 9–14, Jan. 2021, doi: 10.5120/ijca2021921014.

[53] M. Poženel, L. Fürst, D. Vavpotič, and T. Hovelja, “Agile Effort Estimation: Comparing the Accuracy and Efficiency of Planning Poker, Bucket System, and Affinity Estimation Methods,” *Int. J. Soft. Eng. Knowl. Eng.*, vol. 33, no. 11n12, pp. 1923–1950, Dec. 2023, doi: 10.1142/S021819402350064X.

[54] E. Coelho and A. Basu, “Effort Estimation in Agile Software Development using Story Points,” *IJAIS*, vol. 3, no. 7, pp. 7–10, Aug. 2012, doi: 10.5120/ijais12-450574.

[55] S. Bhalerao and M. Ingle, “Incorporating Vital Factors in Agile Estimation through Algorithmic Method,” *IJCSA*, vol. 6, pp. 85–97, 2009.

[56] P. Kokol, “The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature,” *Information*, vol. 15, no. 6, p. 354, Jun. 2024, doi: 10.3390/info15060354.

[57] J. Keung, “Software Development Cost Estimation Using Analogy: A Review,” in *Software Development Cost Estimation Using Analogy: A Review*, Gold Coast, QLD, Australia, 2009, pp. 327–336. doi: 10.1109/ASWEC.2009.32.

[58] A. Idri, F. A. Amazal, and A. Abran, “Analogy-based software development effort estimation: A systematic mapping and review,” *Information and Software Technology*, vol. 58, pp. 206–230, Feb. 2015, doi: 10.1016/j.infsof.2014.07.013.

[59] P. Phannachitta, “Robust comparison of similarity measures in analogy based software effort estimation,” in *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, Malabe: IEEE, Dec. 2017, pp. 1–7. doi: 10.1109/SKIMA.2017.8294126.

[60] M. Auch, M. Weber, P. Mandl, and C. Wolff, “Similarity-based analyses on software applications: A systematic literature review,” *Journal of Systems and Software*, vol. 168, p. 110669, Oct. 2020, doi: 10.1016/j.jss.2020.110669.

[61] S. Hameed, Y. Elsheikh, and M. Azzeh, “An optimized case-based software project effort estimation using genetic algorithm,” *Information and Software Technology*, vol. 153, p. 107088, Jan. 2023, doi: 10.1016/j.infsof.2022.107088.

[62] E. I. Mustafa and R. Osman, “A random forest model for early-stage software effort estimation for the SEERA dataset,” *Information and Software Technology*, vol. 169, p. 107413, May 2024, doi: 10.1016/j.infsof.2024.107413.

[63] P. Phannachitta, “On an optimal analogy-based software effort estimation,” *Information and Software Technology*, vol. 125, p. 106330, Sep. 2020, doi: 10.1016/j.infsof.2020.106330.

[64] M. Rahman, T. Goncalves, and H. Sarwar, “Review of Existing Datasets Used for Software Effort Estimation,” *IJACSA*, vol. 14, no. 7, 2023, doi: 10.14569/IJACSA.2023.01407100.

[65] M. F. Bosu and S. G. MacDonell, “Experience: Quality Benchmarking of Datasets Used in Software Effort Estimation,” 2020, doi: 10.48550/ARXIV.2012.10836.

[66] A. Jadhav and S. K. Shandilya, “Reliable machine learning models for estimating effective software development efforts: A comparative analysis,” *Journal of Engineering Research*, vol. 11, no. 4, pp. 362–376, Dec. 2023, doi: 10.1016/j.jer.2023.100150.

[67] N. Alsadi, S. A. Gadsden, and J. Yawney, “Intelligent estimation: A review of theory, applications, and recent advances,” *Digital Signal Processing*, vol. 135, p. 103966, Apr. 2023, doi: 10.1016/j.dsp.2023.103966.

[68] A. G. Priya Varshini, K. Anitha Kumari, D. Janani, and S. Soundariya, “Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation,” *J. Phys.: Conf. Ser.*, vol. 1767, no. 1, p. 012019, Feb. 2021, doi: 10.1088/1742-6596/1767/1/012019.

[69] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, “Software Effort Estimation Accuracy Prediction of Machine Learning Techniques: A Systematic Performance Evaluation,” *Journal of Software: Practice and Experience*, vol. 52, no. 1, pp. 39–65, 2021, doi: 10.1002/spe.3009.

[70] M. Turic, S. Celar, S. Dragicevic, and L. Vickovic, “Advanced Bayesian Network for Task Effort Estimation in Agile Software Development,” *Applied Sciences*, vol. 13, no. 16, p. 9465, Aug. 2023, doi: 10.3390/app13169465.

[71] A. Jadhav, S. K. Shandilya, I. Izonin, and M. Gregus, “Effective Software Effort Estimation Leveraging Machine Learning for Digital Transformation,” *IEEE Access*, vol. 11, pp. 83523–83536, 2023, doi: 10.1109/ACCESS.2023.3293432.

[72] A. Jadhav, S. K. Shandilya, I. Izonin, and R. Muzyka, “Multi-Step Dynamic Ensemble Selection to Estimate Software Effort,” *Applied Artificial Intelligence*, vol. 38, no. 1, p. 2351718, Dec. 2024, doi: 10.1080/08839514.2024.2351718.

[73] F. Habibi, O. T. Birgani, H. Koppelaar, and S. N. Radenović, “Using fuzzy logic to improve the project time and cost estimation based on Project Evaluation and Review Technique (PERT),” *Journal of Project Management*, vol. 3, no. 4, pp. 183–196, 2018, doi: 10.5267/J.JPM.2018.4.002.

[74] J. R. Dorp, “A dependent project evaluation and review technique: A Bayesian network approach,” *European Journal of Operational Research*, vol. 280, no. 2, pp. 689–706, 2020, doi: 10.1016/j.ejor.2019.07.051.

[75] O. Aljumaili, “The Automation of Critical Path Method using Machine Learning: A Conceptual Study,” *Int. j. eng. manag. res.*, vol. 11, no. 3, Jun. 2021, doi: 10.31033/ijemr.11.3.38.

[76] S. Goyal and A. Parashar, “Machine Learning Application to Improve COCOMO Model using Neural Networks,” *IJITCS*, vol. 10, no. 3, pp. 35–51, Mar. 2018, doi: 10.5815/ijitcs.2018.03.05.

[77] R. K. Sachan *et al.*, “Optimizing Basic COCOMO Model Using Simplified Genetic Algorithm,” *Procedia Computer Science*, vol. 89, pp. 492–498, 2016, doi: 10.1016/j.procs.2016.06.107.

[78] S. M. Satapathy, B. P. Acharya, and S. K. Rath, “Early stage software effort estimation using random forest technique based on use case points,” *The Institution of Engineering and Technology: Software*, vol. 10, no. 1, pp. 10–17, 2016, doi: 10.1049/iet-sen.2014.0122.

[79] J. Sauvola, S. Tarkoma, M. Klemettinen, J. Riekki, and D. Doermann, “Future of software development with generative AI,” *Autom Softw Eng*, vol. 31, no. 1, p. 26, May 2024, doi: 10.1007/s10515-024-00426-z.

[80] O. Karyy, L. Halkiv, and A. Tsapulych, “Development of the IT-Sphere of Ukraine: Factors and Directions of Activation,” *SEMI*, vol. 5, no. 1, pp. 42–55, Jun. 2021, doi: 10.23939/semi2021.01.042.

[81] С. Кравченко, “Засоби оцінки вартості програмного забезпечення,” *Вісник ЖДТУ*, vol. 1, pp. 112–117, 2014.

[82] Ю. Рябокінь, “Оцінка вартості програмного забезпечення,” *Вісник Київського національного університету технологій та дизайну. Серія : Технічні науки*, vol. 1, pp. 117–124, 2015.

[83] Л. Лозовська and В. Дудник, “Сучасні підходи до вартісної оцінки програмних продуктів,” *Європейський вектор економічного розвитку. Економічні науки*, vol. 2, pp. 131–139, 2014.

[84] І. Кожевніков, “Емпіричні моделі оцінки вартості розробки програмного забезпечення,” *Економічний простір*, vol. 83, pp. 195–208, 2014.

[85] Д. Баценко, “Метод калібрування моделі COCOMO шляхом редукції основного рівняння,” *Інженерія програмного забезпечення*, vol. 1, no. 9, pp. 16–24, 2011.

[86] V. Levykin, M. Ievlanov, O. Neumyvakina, I. Levykin, and A. Nakonechnyi, “Estimation of IT-project efforts for information system creation in the conditions of re-use of its functions,” *EEJET*, vol. 2, no. 2 (128), pp. 6–19, Apr. 2024, doi: 10.15587/1729-4061.2024.301227.

[87] *ISO/IEC 19761:2011 Software engineering — COSMIC: a functional size measurement method*, ISO/IEC 19761:2011, Mar. 2011. Accessed: Mar. 01, 2025. [Online]. Available: https://www.iso.org/standard/54849.html

[88] V. Litvinov and A. Moskaliuk, “Modification of the PERT method for project time evaluation taking into account unexpected delays,” *EEJET*, vol. 4, no. 3 (94), pp. 6–13, Aug. 2018, doi: 10.15587/1729-4061.2018.140752.

[89] R. Bihun and I. Karpov, “Modeling and Optimization of Task Scheduling in Multi-Team IT Projects,” *Vìsn. Nac. unìv. “L'vìv. polìteh.”, Ser. Ìnf. sist. merežì*, vol. 16, pp. 104–111, Dec. 2024, doi: 10.23939/sisn2024.16.104.

[90] S. Prykhodko, I. Shutko, and A. Prykhodko, “Early LOC Estimation of Web Apps Created Using Yii Framework by Nonlinear Regression Models,” *WSEAS TRANSACTIONS ON COMPUTERS*, vol. 20, pp. 321–328, Nov. 2021, doi: 10.37394/23205.2021.20.35.

[91] S. Prykhodko, N. Prykhodko, and K. Knyrik, “Estimating the Efforts of Mobile Application Development in the Planning Phase Using Nonlinear Regression Analysis,” *Applied Computer Systems*, vol. 25, no. 2, pp. 172–179, Dec. 2020, doi: 10.2478/acss-2020-0019.

[92] S. Prykhodko, I. Shutko, and A. Prykhodko, “Early size estimation of web apps created using codeigniter framework by nonlinear regression models,” *reks*, no. 3, pp. 84–94, Oct. 2022, doi: 10.32620/reks.2022.3.06.

[93] S. B. Prykhodko, A. V. Pukhalevych, K. S. Prykhodko, and L. M. Makarova, “Nonlinear Regression Models for Estimating the Duration of Software Development in Java for PC Based on the 2021 ISBSG Data,” *RIC*, no. 3, p. 144, Oct. 2022, doi: 10.15588/1607-3274-2022-3-14.

[94] Л. Хрущ, *Економіка програмного забезпечення*. Івано-Франківськ: Прикарпатський національний університет імені Василя Стефаника, 2018. Accessed: Mar. 23, 2025. [Online]. Available: http://lib.pnu.edu.ua:8080/bitstream/123456789/11952/1/1%D0%95%D0%9F%D0%97_%D0%A5%D1%80%D1%83%D1%89.pdf

[95] М. О. Сидоров, П. Ю. Родіонов, and О. І. Марченко, *Економіка ІТ-індустрії та підприємництво: навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою «Інженерія програмного забезпечення інформаційних систем» спеціальності 121 Інженерія програмного забезпечення*. Київ: КПІ ім. Ігоря Сікорського, 2024. Accessed: Mar. 29, 2025. [Online]. Available:

https://ela.kpi.ua/server/api/core/bitstreams/5843debf-51cb-4a67-8ee3-75c1f4424cf9/content

[96] M. Hammer and J. Champy, *Reengineering the corporation: a manifesto for business revolution*, 1st ed. New York, NY: HarperBusiness, 1993.

[97] T. H. Davenport, *Process Innovation: Reengineering Work Through Information Technology*. Boston, MA: Harvard Business Review Press, 1993.

[98] L. I. Chernobaj and O. I. Duma, "Business-process of the enterprise: general description and economical essence," *Lviv Polytechnic National University. Series: Management and entrepreneurship in Ukraine: stages of rasing and development problems*, vol. 769, pp. 125–131, 2013.

[99] *Business Process Model and Notation (BPMN)*, formal/2011-01-03, Jan. 2011. Accessed: May 05, 2024. [Online]. Available: http://www.omg.org/spec/BPMN/2.0

[100] M. Weske, "Concepts, Languages, Architectures," in *Business Process Management*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-73522-9.

[101] A. Burattin, *Process Mining Techniques in Business Environments: Theoretical Aspects, Algorithms, Techniques and Open Challenges in Process Mining*, vol. 207. in Lecture Notes in Business Information Processing, vol. 207. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-17482-2.

[102] W. M. P. van der Aalst, *Process mining: data science in action, 2nd ed*. Berlin: Springer, 2016. doi: 10.1007/978-3-662-49851-4.

[103] R. Agrawal, D. Gunopulos, and F. Leymann, "Mining process models from workflow logs," in *Advances in Database Technology — EDBT'98*, vol. 1377, H.-J. Schek, G. Alonso, F. Saltor, and I. Ramos, Eds., in Lecture Notes in Computer Science, vol. 1377. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 467–483. doi: 10.1007/BFb0101003.

[104] P. Harmon, *Business process change: a guide for business managers and BPM and six sigma professionals*, 2nd ed. Amsterdam ; Boston: Elsevier/Morgan Kaufmann Publishers, 2007.

[105] S. Kemsley, "The Changing Nature of Work: From Structured to Unstructured, from Controlled to Social," in *Business Process Management*, vol. 6896, S. Rinderle-Ma, F. Toumani, and K. Wolf, Eds., in Lecture Notes in Computer Science, vol. 6896. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 2–2. doi: 10.1007/978-3-642-23059-2_2.

[106] O. Marjanovic and R. Freeze, "Knowledge Intensive Business Processes: Theoretical Foundations and Research Challenges," in *2011 44th Hawaii International Conference on System Sciences*, Kauai, HI: IEEE, Jan. 2011, pp. 1–10. doi: 10.1109/HICSS.2011.271.

[107] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-30409-5.

[108] C. Di Ciccio, A. Marrella, and A. Russo, "Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches," *J Data Semant*, vol. 4, no. 1, pp. 29–57, 2015, doi: 10.1007/s13740-014-0038-4.

[109] W. Van Der Aalst *et al.*, "Process Mining Manifesto," in *Business Process Management Workshops*, vol. 99, F. Daniel, K. Barkaoui, and S. Dustdar, Eds., in Lecture Notes in Business Information Processing, vol. 99. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 169–194. doi: 10.1007/978-3-642-28108-2_19.

[110] I. Ailenei, A. Rozinat, A. Eckert, and W. M. P. Van Der Aalst, "Definition and Validation of Process Mining Use Cases," in *Business Process Management Workshops*, vol. 99, F. Daniel, K. Barkaoui, and S. Dustdar, Eds., in Lecture Notes in Business Information Processing, vol. 99. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 75–86. doi: 10.1007/978-3-642-28108-2_7.

[111] W. M. P. Van Der Aalst and A. J. M. M. Weijters, “Process mining: a research agenda,” *Computers in Industry*, vol. 53, no. 3, pp. 231–244, Apr. 2004, doi: 10.1016/j.compind.2003.10.001.

[112] “Market Guide for Process Mining,” Gartner. Accessed: Jun. 30, 2018. [Online]. Available: https://www.gartner.com/en/documents/3870291

[113] M. Weske, *Business Process Management*, 1st ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-73522-9.

[114] W. Van Der Aalst, T. Weijters, and L. Maruster, “Workflow mining: discovering process models from event logs,” *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004, doi: 10.1109/TKDE.2004.47.

[115] A. J. M. M. Weijters and W. M. P. Van Der Aalst, “Rediscovering workflow models from event-based data using little thumb,” *ICA*, vol. 10, no. 2, pp. 151–162, May 2003, doi: 10.3233/ICA-2003-10205.

[116] A. J. M. M. Weijters and J. T. S. Ribeiro, “Flexible Heuristics Miner (FHM),” in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Paris, France: IEEE, Apr. 2011, pp. 310–317. doi: 10.1109/CIDM.2011.5949453.

[117] C. W. Günther and W. M. P. Van Der Aalst, “Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics,” in *Business Process Management*, vol. 4714, G. Alonso, P. Dadam, and M. Rosemann, Eds., in Lecture Notes in Computer Science, vol. 4714. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 328–343. doi: 10.1007/978-3-540-75183-0_24.

[118] C. W. Günther, “Process Mining in Flexible Environments,” Ph.D dissertation, Eindhoven University of Technology, Eindhoven, 2009. doi: https://doi.org/10.6100/IR644335.

[119] Ch. W. Günther and A. Rozinat, “Disco: Discover Your Processes,” in *Proceedings of the Demonstration Track of the 10th International Conference*

on Business Process Management (BPM 2012), vol. 940, Tallinn, Estonia, 2012, pp. 40–44.

[120] F. Veit, J. Geyer-Klingeberg, J. Madrzak, M. Haug, and J. Thomson, "The proactive insights engine: process mining meets machine learning and artificial intelligence," presented at the 15th Int. Conf. Business Process Management (BPM'2017). BPM Demo Track and BPM Dissertation Award, Barcelona, Spain, 2017.

[121] J. M. Gorey, "Estimate Types," *AACE Bulletin*, vol. 1, no. 1, pp. 4–17, 1958.

[122] S. McConnell, *Software Estimation: Demystifying the Black Art*. Washington, USA: Microsoft Press, 2006.

[123] S. McConnell, *Software project survival guide: how to be sure your first important project isn't your last*. Redmond, Wash: Microsoft Press, 1998.

[124] Project Management Institute, *Project Management Institute, A Guide To The Project Management Body Of Knowledge (PMBOK-Guide) - Sixth version*. in PMBOK® Guide. Project Management Institute, 2017.

[125] L. Weinstein and J. A. Adam, *Guesstimation: Solving the World's Problems on the Back of a Cocktail Napkin*. in EBL-Schweitzer. Princeton: Princeton University Press, 2008. doi: 10.1515/9781400824441.

[126] F. M. Webster, "The WBS," *PM Network*, vol. 8, no. 12, pp. 40–46, 1994.

[127] R. E. Luby, D. Peel, and W. Swahl, "Component-based work breakdown structure (CBWBS)," *Project Management Journal*, vol. 26, no. 4, pp. 38–43, 1995.

[128] P. Sudarmaningtyas and R. Mohamed, "A Review Article on Software Effort Estimation in Agile Methodology," *JST*, vol. 29, no. 2, pp. 837–861, 2021, doi: 10.47836/PJST.29.2.08.

[129] B. W. Tuckman, "Developmental sequence in small groups," *Psychological Bulletin*, vol. 63, no. 6, pp. 384–399, 1965, doi: 10.1037/h0022100.

[130] H. L. Gantt, “A Graphical Daily Balance in Manufacture,” *Transactions of the American Society of Mechanical Engineers*, vol. 24, pp. 1322–1331, Jan. 1903, doi: 10.1115/1.4060667.

[131] J. Geraldi and T. Lechter, “Gantt charts revisited: A critical analysis of its roots and implications to the management of projects today,” *International Journal of Managing Projects in Business*, vol. 5, no. 4, pp. 578–594, Sep. 2012, doi: 10.1108/17538371211268889.

[132] Project Management Institute, Ed., *A guide to the project management body of knowledge (PMBOK® guide) and the standard for project management*, Seventh Edition. Pennsylvania: Project Management Institute, Inc, 2021.

[133] J. M. J. Schutten, “List scheduling revisited,” *Operations Research Letters*, vol. 18, no. 4, pp. 167–170, Feb. 1996, doi: 10.1016/0167-6377(95)00057-7.

[134] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, “Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey,” in *Annals of Discrete Mathematics*, vol. 5, Elsevier, 1979, pp. 287–326. doi: 10.1016/S0167-5060(08)70356-X.

[135] R. L. Graham, “Bounds on Multiprocessing Timing Anomalies,” *SIAM J. Appl. Math.*, vol. 17, no. 2, pp. 416–429, Mar. 1969, doi: 10.1137/0117039.

[136] J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan, “Scheduling subject to resource constraints: classification and complexity,” *Discrete Applied Mathematics*, vol. 5, no. 1, pp. 11–24, Jan. 1983, doi: 10.1016/0166-218X(83)90012-4.

[137] P. Brucker, *Scheduling Algorithms*, 5. 5th ed. 2007. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-69516-5.

[138] S. Hartmann and D. Briskorn, “An updated survey of variants and extensions of the resource-constrained project scheduling problem,” *European Journal of Operational Research*, vol. 297, no. 1, pp. 1–14, Feb. 2022, doi: 10.1016/j.ejor.2021.05.004.

[139] W. Herroelen, B. De Reyck, and E. Demeulemeester, “Resource-constrained project scheduling: A survey of recent developments,” *Computers & Operations Research*, vol. 25, no. 4, pp. 279–302, Apr. 1998, doi: 10.1016/S0305-0548(97)00055-5.

[140] A. Ciupe, S. Meza, and B. Orza, “Heuristic Optimization for the Resource Constrained Project Scheduling Problem: a Systematic Mapping,” presented at the 2016 Federated Conference on Computer Science and Information Systems, Oct. 2016, pp. 619–626. doi: 10.15439/2016F389.

[141] R. Kolisch and S. Hartmann, “Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis,” in *Project Scheduling*, vol. 14, J. Węglarz, Ed., in International Series in Operations Research & Management Science, vol. 14. , Boston, MA: Springer US, 1999, pp. 147–178. doi: 10.1007/978-1-4615-5533-9_7.

[142] O. H. Türkakın, D. Arditi, and E. Manisalı, “Comparison of Heuristic Priority Rules in the Solution of the Resource-Constrained Project Scheduling Problem,” *Sustainability*, vol. 13, no. 17, p. 9956, Sep. 2021, doi: 10.3390/su13179956.

[143] A. H. Land and A. G. Doig, “An Automatic Method of Solving Discrete Programming Problems,” *Econometrica*, vol. 28, no. 3, p. 497, Jul. 1960, doi: 10.2307/1910129.

[144] T. L. Saaty, “How to make a decision: The analytic hierarchy process,” *European Journal of Operational Research*, vol. 48, no. 1, pp. 9–26, Sep. 1990, doi: 10.1016/0377-2217(90)90057-I.

[145] M. L. Bynum *et al.*, *Pyomo — Optimization Modeling in Python*, vol. 67. in Springer Optimization and Its Applications, vol. 67. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-68928-5.

[146] W. E. Hart, J.-P. Watson, and D. L. Woodruff, “Pyomo: modeling and solving mathematical programs in Python,” *Math. Prog. Comp.*, vol. 3, no. 3, pp. 219–260, Sep. 2011, doi: 10.1007/s12532-011-0026-8.

[147] *Unified Modeling Language (UML)*, formal/2017-12-05, Dec. 2017. Accessed: May 11, 2024. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/

[148] *Case Management Model and Notation (CMMN)*, formal/2016-12-01, Dec. 2016. Accessed: May 11, 2024. [Online]. Available: https://www.omg.org/spec/CMMN/1.1/

[149] R. P. J. C. Bose, W. M. P. Van Der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling Concept Drift in Process Mining," in *Active Flow and Combustion Control 2018*, vol. 141, R. King, Ed., in Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 141. , Cham: Springer International Publishing, 2011, pp. 391–405. doi: 10.1007/978-3-642-21640-4_30.

[150] "Fuzzy Miner." Accessed: Jan. 29, 2019. [Online]. Available: http://www.processmining.org/online/fuzzyminer

[151] N. Marz and J. Warren, *Big data: principles and best practices of scalable real-time data systems*. Shelter Island, NY: Manning, 2015.

[152] M. Kleppmann, *Designing data-intensive applications: the big ideas behind reliable, scalable, and maintainable systems*, First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2017.

[153] B. F. Van Dongen, A. K. A. De Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. Van Der Aalst, "The ProM Framework: A New Era in Process Mining Tool Support," in *Applications and Theory of Petri Nets 2005*, vol. 3536, G. Ciardo and P. Darondeau, Eds., in Lecture Notes in Computer Science, vol. 3536. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 444–454. doi: 10.1007/11494744_25.

[154] A. Kumar, A. Mishra, and S. Kumar, *Architecting a Modern Data Warehouse for Large Enterprises: Build Multi-cloud Modern Distributed Data Warehouses with Azure and AWS*. Berkeley, CA: Apress, 2024. doi: 10.1007/979-8-8688-0029-0.

ДОДАТКИ

Додаток 1. Огляд методів оцінювання проєктів з розробки програмного забезпечення

Таблиця Д-1.1

Огляд методів оцінювання проєктів з розробки програмного забезпечення

№	*Назва методу англійською мовою*	*Абревіатура англійською мовою*	*Назва методу українською мовою*	*Категорія*	*Етапи життєвого циклу проєкту з розробки ПЗ*	*Одиниця вимірювання трудоємності*	*Використання історичних даних*	*Застосування експертних оцінок*
1	Project/program evaluation and review technique	PERT	–	«Класичні» методи	Аналіз та проєктування, реалізація	Людино-години	Ні	Так
2	Critical path method	CPM	Метод критичного шляху	«Класичні» методи	Аналіз та проєктування, реалізація	Людино-години	Ні	Так
3	Graphical evaluation and review technique	GERT	–	«Класичні» методи	Аналіз та проєктування, реалізація	Людино-години	Ні	Так
4	Software lifecycle management	SLIM	–	«Класичні» методи	Аналіз та проєктування	Кількість ліній коду (SLOC)	Так	Ні
5	Function points	–	Метод функційних точок	«Класичні» методи	Аналіз та проєктування	Функційні точки	Так	Ні
6	Constructive cost model	COCOMO	–	«Класичні» методи	Аналіз та проєктування	Кількість ліній коду (SLOC)	Так	Ні

№	*Назва методу англійською мовою*	*Абревіатура англійською мовою*	*Назва методу українською мовою*	*Категорія*	*Етапи життєвого циклу проєкту з розробки ПЗ*	*Одиниця вимірювання трудоємності*	*Використання історичних даних*	*Застосування експертних оцінок*
7	Wideband Delphi	–	–	«Класичні» методи	Аналіз та проєктування, реалізація	Як буде узгоджено експертами	Ні	Так
8	Function point analysis (IFPUG)	FPA	–	«Класичні» методи 2-ого покоління	Аналіз та проєктування, реалізація	Функційні точки	Так	Ні
9	Use case point analysis	UCP	Метод прецедентних точок	«Класичні» методи 2-ого покоління	Аналіз та проєктування	Прецедентні точки	Так	Ні
10	Software non-functional assessment process	SNAP	–	«Класичні» методи 2-ого покоління	Аналіз та проєктування	SNAP-точки	Так	Ні
11	Common software measurement international consortium	COSMIC	–	«Класичні» методи 2-ого покоління	Аналіз та проєктування	Функційні точки COSMIC	Так	Ні
12	Constructive cost model II	COCOMO II	–	«Класичні» методи 2-ого покоління	Аналіз та проєктування	Кількість ліній коду (SLOC), функційні точки, application-точки	Так	Експертна оцінка може застосовуватися в разі відсутності історичних даних
13	Planning poker	–	–	Agile-методи	Реалізація	Сторі-поінти	Ні	Так

№	*Назва методу англійською мовою*	*Абревіатура англійською мовою*	*Назва методу українською мовою*	*Категорія*	*Етапи життєвого циклу проєкту з розробки ПЗ*	*Одиниця вимірювання трудоємності*	*Використання історичних даних*	*Застосування експертних оцінок*
14	T-shirt sizing	–	–	Agile-методи	Реалізація	Розміри одягу: XS, S, M, L, XL, 2XL, 3XL	Ні	Так
15	Affinity grouping	–	Метод групування за схожістю	Agile-методи	Реалізація	Сторі-поінти	Ні	Так
16	Bucket system	–	–	Agile-методи	Реалізація	Сторі-поінти	Ні	Так
17	Random distribution	–	–	Agile-методи	Реалізація	Сторі-поінти	Ні	Так
18	Dot voting	–	–	Agile-методи	Реалізація	Сторі-поінти	Ні	Так
19	Large, small, uncertain	–	–	Agile-методи	Реалізація	Large, small, uncertain	Ні	Так
20	Constructive agile estimation algorithm	CAEA	–	Agile-методи	Аналіз та проєктування, реалізація	Сторі-поінти	Так	Так
21	Estimation by analogy[1]	–	Оцінювання за аналогією	Методи із застосуванням машинного навчання та ШІ	Аналіз та проєктування	Залежить від наявних історичних даних	Так	Так, в разі відсутності достатнього обсягу історичних даних

[1] Метод «estimation by analogy» віднесений до групи «методи із застосуванням машинного навчання на ШІ», оскільки для визначення «схожості» проєктів може використовуватися метрика (мається на увазі метрика, що використовується у метричних просторах, якими оперує математична наука), побудова якої є математичною задачею, що може вирішуватися із застосуванням машинного навчання.

№	*Назва методу англійською мовою*	*Абревіатура англійською мовою*	*Назва методу українською мовою*	*Категорія*	*Етапи життєвого циклу проєкту з розробки ПЗ*	*Одиниця вимірювання трудоємності*	*Використання історичних даних*	*Застосування експертних оцінок*
22	Machine learning based estimation methods utilizing models trained on the past project data	–	Методи оцінювання, що базуються на застосуванні моделей, «навчених» на наборах даних минулих проєктів	Методи із застосуванням машинного навчання та ШІ	Аналіз та проєктування, реалізація	Залежно від конкретного методу	Так	Залежно від конкретного методу
23	Estimation methods empowered with machine learning and AI	–	Методи оцінювання, покращені за рахунок використання машинного навчання на ШІ	Методи із застосуванням машинного навчання та ШІ	Аналіз та проєктування, реалізація	Залежно від конкретного методу	Так	Залежно від конкретного методу
24	Generative AI based estimation[1]	–	Методи оцінювання із застосуванням генеративного ШІ	Методи із застосуванням машинного навчання та ШІ	Невідомо	Невідомо	Невідомо	Невідомо
25	Multi-stage estimation method[2]	–	Метод поетапного оцінювання	Близький до «класичних» методів оцінювання другого покоління та agile-методів	Аналіз та проєктування	Людино-години (нормалізована оцінка розробки)	Ні	Так

[1] На час написання цієї дисертаційної роботи група методів оцінювання, що базуються на застосування генеративного ШІ, розглядається як перспективний напрямок, який у найближчому майбутньому, з високою ймовірністю, призведе до значних змін у сфері розробки ПЗ.

[2] Метод оцінювання, розроблений в рамках цієї дисертаційної роботи (див. розділ 2).

Додаток 2. Властивості бізнес-процесів залежно від ступеня структурованості

Таблиця Д-2.1

Властивості бізнес-процесів залежно від ступеня структурованості

№	Критерій	Категорія бізнес-процесів		
		Структуровані	*Слабоструктуровані*	*Неструктуровані*
1	Послідовність кроків	Послідовність кроків чітко визначена	Послідовність кроків визначена (але не є чітко фіксованою), допустимі відхилення та нестандартні ситуації	Послідовність кроків процесу можна визначити лише на високому рівні, на детальному рівні послідовність виконання процесу може суттєво змінюватися
2	Розгалуження	Кількість розгалужень відносно невелика	Велика кількість розгалужень, багато залежностей між кроками процесу	Структура розгалужень та залежність між кроками процесу є досить складною
3	Підхід до опису	Наперед визначена детальна послідовність кроків процесу (наприклад, BPNM-модель)	Явно визначена модель процесу (наприклад, у вигляді BPNM- чи CMMN-моделі), бізнес-правила, опис нестандартних ситуацій	Бізнес-правила, «засвоєні уроки» (англ. «lessons learnt»), рекомендації, підказки
4	Рівень компетенції учасників	Компетенція, достатня для виконання чітко поставлених рутинних завдань	Високий рівень компетентності (англ. «knowledgeable worker»)	Експертний рівень
5	Спосіб взаємодії між учасниками	Ієрархічний	Регламентована співпраця	Інтенсивна співпраця
6	Залучення людей в автоматизований процес	Автоматизація з мінімальною участю людини	Участь людей необхідна в ключових точках (наприклад, для прийняття рішень, що вимагають спеціальних знань)	Автоматизація всіх ланок процесу є технічно складно або практично неможливою

№	Критерій	Категорія бізнес-процесів		
		Структуровані	*Слабоструктуровані*	*Неструктуровані*
7	Можливість автоматизації засобами BPM-систем	Класичний BPM підхід з наперед визначеною моделлю процесу (наприклад, з використанням BPMN)	Класичний BPM-підхід та адаптивний кейс-менеджмент (англ. «adaptive case management»)	Автоматизація засобами BPM-систем можлива лише для окремих ланок процесу
8	Моніторинг	Вимірювання, пов'язані із контролем таких показників як час, вартість, якість	Вимірювання, пов'язані із контролем часу, вартості, якості, а також гнучкості (можливості опрацьовувати нестандартні ситуації)	Вимірювання, пов'язані із ефективністю, вираженою у формі цілей
9	Підхід до отримання конкурентної переваги	Продуктивність процесу, стандартизація з метою мінімізації варіацій процесу	Ефективність процесу, застосування компетенції (знань, навиків, досвіду) учасників процесу	Експертні знання, результати командної роботи експертів

Додаток 3. Етапи оцінювання проєктів з розробки програмного забезпечення

Таблиця Д-3.1

Етапи оцінювання проєктів з розробки програмного забезпечення

№	Характеристика	Етапи оцінювання					
		Етап 0	*Етап 1*	*Етап 2*	*Етап 3*	*Етап 4*	*Етап 5*
1	*Назва етапу оцінювання*	*Початкове оцінювання*	*Попереднє оцінювання*	*Проміжне оцінювання*	*Детальне оцінювання*	*Оцінювання проєктною командою та моніторинг реалізації*	*Завершення збору зворотного зв'язку*
2	*Назва етапу оцінювання англійською мовою*	Introductory estimation	Preliminary estimation	Intermediate estimation	Precise estimation	Project team estimation and implementation monitoring	Finalizing feedback collection
3	*Етап життєвого циклу*	Первинне знайомство з ідеєю проєкту	Розвиток ідеї проєкту, початок аналізу та проєктування	Аналіз та проєктування	Завершення аналізу та проєктування	Реалізація проєкту	Після завершення реалізації проєкту
4	*Цілі оцінювання*	Отримати приблизне розуміння «порядку» обсягу ресурсів та часу, необхідних для реалізації проєкту	Надати наближену оцінку ресурсам та часу, необхідних для реалізації проєкту	Надати оцінку, точнішу за ту, яка отримана на етапі попереднього оцінювання	Підготувати оцінку, достатньо точну для взяття зобов'язань щодо очікуваних результатів проєкту та передачі проєкту на етап реалізації	Оцінювання задач проєктною командою, порівняння оцінок із фактично затраченими часом та ресурсами, уточнення оцінки задач, які залиши-	Завершення збору та аналіз зворотного зв'язку щодо оцінки, отриманого за результатами реалізації проєкту

№	Характеристика	Етапи оцінювання					
		Етап 0	*Етап 1*	*Етап 2*	*Етап 3*	*Етап 4*	*Етап 5*
						лися до завершення проєкту	
5	*Основні результати оцінювання*	Порівняння обсягу робіт та складності із вже реалізованими проєктами, орієнтовні тривалість проєкту та склад команди	Зміст проєктних робіт у вигляді ІСЕО, склад проєктної команди, оцінка тривалості проєкту	Зміст проєктних робіт у вигляді ІСЕО, склад проєктної команди, графік реалізації проєкту та оцінка тривалості	Зміст проєктних робіт у вигляді ІСЕО, склад проєктної команди, графік реалізації проєкту та оцінка тривалості, розклад реалізації елементів оцінювання	Оцінка проєктних задач командою реалізації проєкту, фактично затрачений робочий час на реалізацію проєктних задач та його порівняння з оцінкою	Порівняння оцінки із фактично затраченим часом та ресурсами, висновки щодо точності оцінки
6	*Сценарії реалізації проєкту*	Визначаються та оцінюються кілька гіпотетичних сценаріїв реалізації проєкту	Кількість сценарії скорочується (у порівняння із попереднім етапом), може оцінюватися більше як один сценарій	Рекомендується обрати один із сценаріїв для оцінювання	Обирається один сценарій для оцінювання	Аналізуються фактичні та ймовірні відхилення від обраного сценарію реалізації проєкту	Аналізуються фактичні відхилення від сценарію реалізації проєкту, обраного під час оцінювання
7	*Ієрархічна структура елементів оцінювання*	Зазвичай, відсутня	Передбачає мінімальну деталізацію	Детальніша у порівнянні з попереднім оцінюванням	Досить детальна, щоб надати оцінку для взяття зобов'язань щодо реалізації проєкту та формування «беклогу» проєкту	Деталізована до рівня задач реалізації, представлена у вигляді «беклогу» проєкту	Фактично реалізовані задачі проєкту, які можна порівняти з ІСЕО, що була перед початком реалізації

№	Характеристика	Етапи оцінювання					
		Етап 0	*Етап 1*	*Етап 2*	*Етап 3*	*Етап 4*	*Етап 5*
					для етапу реалізації		
8	*Структура робочого часу розробника*	Не застосовується	У явному вигляді не застосовується, враховується у вигляді КРЧР	Враховується у системі рівнянь балансу робочого часу	Враховується у системі рівнянь балансу робочого часу	Моніторинг фактично затраченого робочого часу	–
9	*Графік реалізації проєкту*	Не застосовується	У явному вигляді не застосовується, враховується у вигляді КРЧР	Розбиття на фази реалізації проєкту враховуються у системі рівнянь балансу робочого часу	Розбиття на фази реалізації проєкту враховуються у системі рівнянь балансу робочого часу, надається розклад реалізації елементів оцінювання, можлива прив’язка фаз проєкту до календарних дат	Більш детальне (у порівнянні з попередній етапом) планування, що передбачає формування розкладу реалізації задач проєкту за спринтами, здійснюється прив’язка спринтів до календарних дат	–
10	*Склад проєктної команди*	Приблизний розмір команди без уточнення ролей та спеціалізацій	Визначені командні ролі без диференціації спеціалізацій інженерів-розробників	Визначені командні ролі та спеціалізації інженерів-розробників	Визначені командні ролі та спеціалізації інженерів-розробників	Визначені командні ролі та спеціалізації інженерів-розробників, призначено конкретних учасників команди	–
11	*Система рівнянь*	Не застосовується	Спрощена версія, що явно не враховує структуру	Повна версія, що враховує структуру робочого	Повна версія, що враховує структуру робочого	Повна версія, що враховує структуру робочого часу	–

№	Характеристика	Етапи оцінювання					
		Етап 0	*Етап 1*	*Етап 2*	*Етап 3*	*Етап 4*	*Етап 5*
	балансу робочого часу		робочого часу інженерів-розробників та етапів реалізації проєкту; спрощення досягається за рахунок застосування КРЧР	часу інженерів-розробників та графік реалізації проєкту	часу інженерів-розробників та графік реалізації проєкту	інженерів-розробників та графік реалізації проєкту, а також фактично затрачений робочий час	
12	*Вимоги до проєкту*	Дуже поверхнево сформульовані вимоги	Поверхневе розуміння вимог	Частково проведений аналіз вимог	Вимоги проаналізовано до рівня, достатнього для початку реалізації проєкту	Вимоги деталізовані до задач реалізації проєкту	–
13	*Архітектура*	Відсутня	Високорівнева архітектура для кожного із оцінюваних сценаріїв	Більш детальний (порівняно з попереднім етапом) дизайн архітектури	Дизайн архітектури, готовий до передачі на етап реалізації проєкту	Дизайн архітектури деталізовано до задач реалізації	–
14	*Хто здійснює оцінювання*	Команда оцінювання	Команда оцінювання	Команда оцінювання	Команда оцінювання, можливе залучення команди реалізації для верифікації оцінки та передачі її на етап реалізації проєкту	Команда реалізації проєкту	Команда реалізації та команда оцінювання

№	Характеристика	Етапи оцінювання					
		Етап 0	*Етап 1*	*Етап 2*	*Етап 3*	*Етап 4*	*Етап 5*
15	*Очікувана точність оцінки*[1]	Не можливо визначити точність	Досить низька: -50% ... +100%	Низька: -25% ... +50%	Достатня: -5% ... +10%	Висока, постійно уточнювана по мірі реалізації проєкту	Порівняння оцінки із фактично затраченими ресурсами та часом
16	*Використання для взяття зобов'язань щодо реалізації*	Категорично не рекомендовано	Строго не рекомендовано	Не рекомендовано	Може використовуватися для взяття зобов’язань щодо реалізації проєкту	Може використовуватися для взяття зобов’язань щодо реалізації проєкту	–

[1] Вказані інтервали відображають очікувану точність оцінки, однак метод поетапного оцінювання не може гарантувати цієї точності на етапі оцінювання.

Додаток 4. Основні терміни методу поетапного оцінювання проєктів з розробки програмного забезпечення

Таблиця Д-4.1

Основні терміни методу поетапного оцінювання проєктів з розробки програмного забезпечення

№	*Термін українською мовою*	*Абревіатура українською мовою*	*Термін англ. мовою*	*Абревіатура англ. мовою*	*Символічне позначення*	*Визначення терміну*
1	Елемент оцінювання	–	Estimable item	–	x	*Елементом оцінювання* називатимемо частину змісту проєктних робіт, для якої може бути оцінена трудоємність (тобто вказана нормалізована оцінка розробки, НОР), здійснена декомпозиція на дочірні елементи, проведено аналіз припущень, залежностей та ризиків. Елемент оцінювання володіє набором атрибутів; значення атрибуту може бути отримано за допомогою квадратних дужок.
2	Простий елемент оцінювання	–	Leaf estimable item	LEI	x	Елемент оцінювання називатимемо *простим*, якщо він не має дочірніх елементів.
3	Композитний елемент оцінювання	–	Composite estimable item	CEI	x	Елемент оцінювання називатимемо *композитним*, якщо він має дочірні елементи.
4	Ієрархічна структура елементів оцінювання	ICEO	Estimable items breakdown structure	EIBS	X	*Ієрархічною структурою елементів оцінювання* називатимемо дерево (згідно визначення з теорії графів) з вершинами, які є елементами оцінювання, ребрами, що пов’язують батьківські та дочірні

№	Термін українською мовою	Абревіатура українською мовою	Термін англ. мовою	Абревіатура англ. мовою	Символічне позначення	Визначення терміну
						елементи оцінювання, та кореневим елементом, який представляє зміст робіт всього проєкту.
5	Структура робочого часу інженера-розробника	–	Structure of software developer working time	–	–	*Структура робочого часу інженера-розробника* – це схема розподілу робочого часу інженера-розробника, що включає як виконання проєктних завдань, так і «неробочий» час, коли інженер-розробник фактично не працює над задачами проєкту (наприклад, час відпустки чи лікарняних).
6	Проєктний робочий час	–	Project working time	PWT	W	*Проєкний робочий час* є повним робочим часом, який затрачає учасник команди, будучи залученим до реалізації проєкту.
7	Повний робочий час розробки	–	Full development working time	FDWT	M	*Повний робочий час розробки* – це частина проєктного робочого часу, що затрачається інженером-розробником на реалізацію задач із змісту проєктних робіт, включаючи також такі супровідні активності розробки як написання модульних тестів, виправлення дефектів, взаємодію з іншими учасниками команди тощо.
8	Робочий час розробки	–	Development working time	DWT	D	*Робочий час розробки* є частиною повного робочого часу розробки, що затрачається безпосередньо на реалізацію проєктних задач (як правило, на написання коду), не включаючи супровідні задачі такі як: написання модульних тестів, виправлення дефектів, взаємодія з іншими учасниками команди тощо.
9	Робочий час супровідних	–	Supplementary development	–	A	*Робочий час супровідних активностей розробки* – це частина повного робочого часу розробки, що затрачається інженерами-розробниками на виконання

№	Термін українською мовою	Абревіатура українською мовою	Термін англ. мовою	Абревіатура англ. мовою	Символічне позначення	Визначення терміну
	активностей розробки		activities working time			завдань, що супроводжують основну розробку. До супровідних активностей розробки, наприклад, належить: написання модульних тестів, виправлення дефектів, взаємодія з іншими учасниками команди.
10	Робочий час загальних проєктних активностей	–	General project activities working time	–	*G*	*Робочий час загальних проєктних активностей* – це частина проєктного робочого часу, що затрачається інженером-розробником на завдання, не пов'язані безпосередньо з розробкою. Приклади загальних проєктних активностей: щоденні наради, планування спринтів, презентація результатів роботи, налаштування робочого середовища на початку проєкту, стабілізаційні задачі перед релізом.
11	Проєктний «неробочий» час	–	Project non-working time	–	*N*	*Проєктний «неробочий» час* – це частина проєктного робочого часу, коли інженер-розробник не виконує жодних проєктних задач, наприклад: відпустки, лікарняні, а також час простій, якщо у розробника немає задач чи виконання задач неможливе з певних причин.
12	Проєктний час простою	–	Project idle time	–	*I*	*Проєктний час простою* – це частина проєктного «неробочого часу», коли інженер-розробник, будучи присутнім на проєкті, фактично не виконує проєктних задач. Для прикладу, така ситуація може виникнути в разі недостатності проєктних задач або коли реалізація задач заблокована залежностями від інших ще нереалізованих задач.
13	Проєктний час відсутності	–	Project leaves time	–	*O*	*Проєктний час відсутності* – це частина проєктного «неробочого часу», коли інженер-розробник

№	Термін українською мовою	Абревіатура українською мовою	Термін англ. мовою	Абревіатура англ. мовою	Символічне позначення	Визначення терміну
						відсутній, наприклад: плановані відпустки, незаплановані вихідні, лікарняні тощо.
14	Резерв проєктного робочого часу	–	Project working time contingency reserve	–	R	*Резерв проєктного робочого часу* є частиною проєктного робочого часу, що в разі «матеріалізації» ризиків може бути затрачена на кожну із трьох інших компонент структури робочого часу: розробку, загальні проєктні активності чи «неробочий» час.
15	Інженер-розробник середнього рівня компетентності	ІРСРК	Software developer of the average competency level	SDACL	–	*Інженером-розробником середнього рівня компетентності* називатимемо інженера-розробника, який володіє достатнім рівнем компетенцій (тобто має достатній рівень знань, навичок та досвіду), щоб працювати у проєктній команді, виконуючи задачі певного типу із належним рівнем якості без необхідності постійного нагляду зі сторони більш кваліфікованих колег; при цьому такий інженер-розробник все ще має простір для підвищення рівня володіння компетентностями з точки зору підвищення ефективності та збільшення складності виконуваних завдань.
16	Нормалізований робочий час розробки	НРЧР	Normalized development working time	NDWT	U	*Нормалізованим робочим часом розробки* називатимемо робочий час розробки, затрачений ІРСРК, який працює на повне навантаження і не залучений до роботи на інших проєктах, на реалізацію певного обсягу проєктних задач.
17	Коефіцієнт продуктивності	–	Productivity coefficient	–	ρ	*Коефіцієнтом продуктивності* інженера-розробника називатимемо відношення НРЧР, затраченого ІРСРК на реалізацію певного обсягу проєктних робіт, до робочого часу розробки, затраченого іншим

№	Термін українською мовою	Абревіатура українською мовою	Термін англ. мовою	Абревіатура англ. мовою	Символічне позначення	Визначення терміну
						інженером-розробником на реалізацію тих самих проєктних робіт.
18	Коефіцієнт продуктивності за рівнем компетентності	КПРК	Competency level productivity coefficient	CLPC	α	*Коефіцієнтом продуктивності за рівнем компетентності* інженера-розробника називатимемо відношення НРЧР, затраченого ІРСРК на реалізацію певного обсягу проєктних робіт, до робочого часу розробки, затраченого іншим інженером-розробником на реалізацію тих самих проєктних робіт за умови, що інженер-розробник залучений до проєкту на повне навантаження.
19	Коефіцієнт продуктивності за ступенем залученості	КПСЗ	Involvement productivity coefficient	IPC	η	*Коефіцієнтом продуктивності за ступенем залученості* інженера розробника називатимемо відношення робочого часу розробки, затраченого цим інженером-розробником на реалізацію певного обсягу проєктних робіт, до робочого часу розробки, затраченого цим інженером-розробником на реалізацію тих самих проєктних робіт за умови, що інженер-розробник залучений до проєкту на повне навантаження.
20	Нормалізована спроможність розробки	НСР	Normalized development capacity	NDC	C	*Нормалізованою спроможністю розробки* називатимемо максимально можливий обсяг НРЧР, який інженер-розробник (або команда інженерів-розробників) може затратити на реалізацію проєктних задач.
21	Нормалізована оцінка розробки	НОР	Normalized development estimate	NDE	E	*Нормалізованою оцінкою розробки* певного обсягу проєктних робіт є прогноз НРЧР, необхідного для реалізації цього обсягу робіт.

№	***Термін українською мовою***	***Абревіатура українською мовою***	***Термін англ. мовою***	***Абревіатура англ. мовою***	***Символічне позначення***	***Визначення терміну***
22	Нормалізований еквівалент повної зайнятості	НЕПЗ	Normalized development full-time equivalent	ND-FTE	ψ	*Нормалізованим еквівалентом повної зайнятості розробника* називатимемо добуток коефіцієнту продуктивності на еквівалент повної зайнятості цього розробника.
23	Графік реалізації проєкту	–	Project implementation schedule	–	H	*Графік реалізації проєкту* – це часовий проміжок з ієрархічною структурою фаз, упродовж якого здійснюється реалізація проєкту.
24	Шаблон графіку реалізації проєкту	–	Project implementation schedule template	–	–	*Шаблоном графіку реалізації проєкту* називатимемо ієрархічну структуру фаз проєкту та значень параметрів, асоційованих з цими фазами (наприклад, тривалості фаз), у якій закладено певний «простір» для варіації значень цих параметрів (наприклад, можливість зміни в певних межах тривалості фази основної розробки), що дає змогу побудувати різні варіанти графіків реалізації проєкту на основі одного і того ж шаблону.
25	Склад проєктної команди	–	Project team composition	–	T	*Склад проєктної команди* – це множина ролей учасників команди та асоційованих з ними параметрів, наприклад, ЕПЗ. Проєктна команда включає ролі двох типів: ролі інженерів-розробників та «нерозробницькі» ролі (проєктні менеджери, архітектори, бізнес-аналітики, дизайнери, інженери-тестувальники тощо).
26	Шаблон складу проєктної команди	–	Project team composition template	–	–	*Шаблоном складу проєктної команди* називатимемо сукупність множини ролей учасників команди та пов’язані з кожною із ролей значення ЕПЗ, а також множини правил, що визначають залежності між цими значеннями ЕПЗ (варто підкреслити, що

№	Термін українською мовою	Абревіатура українською мовою	Термін англ. мовою	Абревіатура англ. мовою	Символічне позначення	Визначення терміну
						значення ЕПЗ для однієї і тієї ж ролі можуть відрізнятися залежно від фази реалізації проєкту).
27	Команда інженерів-розробників без диференціації спеціалізацій	–	Development team with non-differentiated specializations	–	–	*Команда інженерів-розробників без диференціації спеціалізацій* передбачає, що кожен з інженерів-розробників може виконувати (тобто володіє відповідною компетентністю) будь-яку із задач із змісту проєктних робіт.
28	Команда інженерів-розробників з диференційованими спеціалізаціями	–	Development team with differentiated specializations	–	–	*Команда інженерів-розробників з диференційованими спеціалізаціями* передбачає, що кожен з інженерів-розробників може виконувати (тобто володіє відповідною компетентністю) задачі із розробки, які належать до його спеціалізації; при цьому кожен з інженерів-розробників володіє лише однією спеціалізацією.
29	Команда інженерів-розробників із змішаними спеціалізаціями	–	Development team with mixed specializations	–	–	*Команда інженерів-розробників із змішаними спеціалізаціями* передбачає, що до її складу входять інженери-розробники, які можуть виконувати (тобто володіють відповідними компетентностями) задачі із розробки, які належать до більше як однієї спеціалізації; іншими словами, до такої команди входять інженери-розробники, що володіють більш як однією спеціалізацією.
30	Система рівнянь балансу робочого часу	–	System of working-time balance equations	–	–	*Система рівнянь балансу робочого часу* проєктної команди визначає взаємозв'язок між такими складовими оцінки проєкту з розробки ПЗ як структура робочого часу інженерів розробників, склад проєктної команди, графік реалізації проєкту, а також НОР змісту проєктних робіт.

№	Термін українською мовою	Абревіатура українською мовою	Термін англ. мовою	Абревіатура англ. мовою	Символічне позначення	Визначення терміну
31	Коефіцієнт робочого часу розробки	КРЧР	Development working time coefficient	DWTC	ξ	*Коефіцієнт робочого часу розробки* – це відношення проєктного робочого часу інженерів-розробників до робочого часу розробки. Коефіцієнт робочого часу розробки використовується під час попереднього оцінювання для спрощення структури робочого часу інженерів-розробників та системи рівнянь балансу робочого часу.
32	Альтернативний коефіцієнт робочого часу розробки	–	Alternative development working time coefficient	–	κ	*Коефіцієнт робочого часу розробки* – це відношення проєктного робочого часу інженерів-розробників до НОР. Альтернативний коефіцієнт робочого часу розробки використовується під час попереднього оцінювання для спрощення структури робочого часу інженерів-розробників та системи рівнянь балансу робочого часу.
33	«Розмір» простого елементу оцінювання	–	Leaf estimable item point	EIP	$x[\mathrm{EIP}]$	*«Розмір» простого елементу оцінювання* – це додатне число, що представляє його відносну трудоємність у порівнянні з іншими простими елементами оцінювання, що належать до тієї ж ІСЕО.
34	Невизначеність простого елементу оцінювання	–	Leaf estimable item uncertainty	EIU	$x[\mathrm{EIU}]$	*Невизначеність простого елементу оцінювання* – це безрозмірне додатне число, що позначає відносний рівень невідомого (у порівнянні з іншими простими елементами оцінювання, що належать до тієї ж ІСЕО), пов’язаного з цим елементом оцінювання.
35	Коефіцієнт супровідних активностей розробки	КСАР	Supplementary development activities coefficient	SDAC	d	*Коефіцієнт супровідних активностей розробки* – це відношення робочого часу, затраченого інженером-розробником на супровідні активності розробки, до робочого часу розробки.

№	Термін українською мовою	Абревіатура українською мовою	Термін англ. мовою	Абревіатура англ. мовою	Символічне позначення	Визначення терміну
36	Коефіцієнт загальних проєктних активностей	КЗПА	General project activities coefficient	GPAC	g	*Коефіцієнт загальних проєктних активностей* – це відношення робочого часу, що затрачає інженер-розробник на загальні проєктні активності, до проєктного робочого часу, від якого віднято проєктний час відсутності.
37	Коефіцієнт резерву проєктного робочого часу	КРПРЧ	Coefficient of project working time contingency	CPWTC	r	*Коефіцієнт резерву проєктного робочого часу* – це відношення резерву проєктного робочого часу інженера-розробника до його проєктного робочого часу.
38	Розклад реалізації елементів оцінювання	–	Estimable items implementation schedule	–	J	*Розклад реалізації елементів оцінювання* – це матриця розподілу повного робочого часу розробки, що затрачається на кожній з фаз (спринтів) на реалізацію елементів оцінювання.
39	Нормалізований розклад реалізації елементів оцінювання	–	Normalized estimable items implementation schedule	–	J_N	*Нормалізований розклад реалізації елементів оцінювання* – це матриця розподілу НРЧР, що затрачається на кожній з фаз (спринтів) на реалізацію елементів оцінювання.

Додаток 5. Атрибути елементів оцінювання

Таблиця Д-5.1

Атрибути елементів оцінювання

№	*Назва атрибуту*	*Назва атрибуту англ. мовою*	*Символічне позначення*	*Кратність*	*Опис*
1	Ідентифікатор	Identifier	ID	1	Унікальний ідентифікатор, необхідний для посилання на цей елемент оцінювання.
2	Заголовок	Title	TITLE	1	Заголовок, який стисло відображає суть елементу оцінювання.
3	Опис	Description	DESC	0...1	Детальний опис елементу оцінювання.
4	Батьківський елемент	Parent	PARENT	0...1	Батьківський елемент оцінювання; батьківський елемент є у всіх елементів оцінювання, крім кореневого елемента ICEO який представляє зміст робіт всього проєкту.
5	Дочірні елементи	Children	CHILD	0...∞	Список дочірніх елементів оцінювання; якщо список дочірніх елементів порожній, то елемент оцінювання називається простим; у протилежному випадку елемент оцінювання називають композитним.
6	Припущення	Assumptions	ASMP	0...∞	Список припущень відносно елементу оцінювання, зроблених з метою зменшення рівня невизначеності.
7	Ризики	Risks	RISK	0...∞	Список ризиків, пов’язаних з елементом оцінювання; для кожного ризику рекомендується вказувати ймовірність та ступінь впливу.
8	Запитання	Questions	Q	0...∞	Список запитань (та, можливо, відповідей), пов’язаних з елементом оцінювання.
9	Залежності	Dependencies	DEPN	0...∞	Список залежностей від інших елементів оцінювання або зовнішніх факторів.

№	Назва атрибуту	Назва атрибуту англ. мовою	Символічне позначення	Крат-ність	Опис
10	Зміст робіт	In scope	SCOPE	0...∞	Опис того, що входить до змісту робіт елементу оцінювання.
11	Що не входить до змісту робіт	Out of scope	OOS	0...∞	Явна специфікація робіт, які не входять до змісту робіт елементу оцінювання.
12	Функційна вимога	Functional requirement	FR	0...∞	Функційна вимога, пов’язана з елементом оцінювання.
13	Нефункційна вимога	Non-functional requirement	NFR	0...∞	Нефункційна вимога, пов’язана з елементом оцінювання.
14	Обмеження	Constraint	CONST	0...∞	Пов’язане з елементом оцінювання прийняте раніше рішення, яке не можливо змінити.
15	Визначення завершеності	Definition of Done	DOD	0...1	Вимоги, виконання яких означає, що реалізацію елементу оцінювання завершено.
16	Критерії прийняття	Acceptance Criteria	AC	0...∞	Перелік критеріїв, виконання яких необхідно для того, щоб реалізація елементу оцінювання була прийнята замовником.
17	Нормалізована оцінка розробки	Normalized development estimate	NDE	0...1	НОР, пов’язана з елементом оцінювання. НОР може бути представлена як скалярним значенням (для команд без диференціації спеціалізацій), так і вектором (у випадку команд з диференційованими чи змішаними спеціалізаціями). Варто зауважити, що НОР може бути представлена у вигляді інтервалу від оптимістичного до песимістичного значень.
18	Відносний розмір	Estimable item point	EIP	0...1	Відносний «розмір» елементу оцінювання, виражений у «точкових» одиницях (див. пункт 2.7.5).
19	Рівень невизначеності	Estimable item uncertainty	EIU	0...1	Відносний рівень невизначеності, пов’язаний з елементом оцінювання (див. пункт 2.7.5).
20	Дизайн архітектури	Architecture considerations	ARCH	0...∞	Аспекти дизайну архітектури, пов’язані з елементом оцінювання.
21	Технологія	Technology	TECH	0...∞	Технології реалізації елементу оцінювання.
22	Компонент	Component	COMP	0...∞	Компонент системи, з яким пов’язаний елемент оцінювання.

№	*Назва атрибуту*	*Назва атрибуту англ. мовою*	*Символічне позначення*	*Крат-ність*	*Опис*
23	Аналогічні задачі	Similar tasks	SIM	0...∞	Аналогічні елементи оцінювання.
24	Реліз проєкту	Release	RELEASE	0...∞	В якій версії проєкту буде реалізований елемент оцінювання.
25	Користувач	User	USER	0...∞	Користувач системи, з яким пов’язаний елемент оцінювання.
26	Спеціалізації інженерів-розробників	Development specializations	SPEC	0...∞	Спеціалізації інженерів-розробників, які будуть займатися реалізацією елементу оцінювання.

Додаток 6. Концептуальна схема слабоструктурованого бізнес-процесу оцінювання проєктів з розробки програмного забезпечення

Таблиця Д-6.1

Концептуальна схема слабоструктурованого бізнес-процесу оцінювання проєктів з розробки програмного забезпечення

№	*Етап бізнес-процесу*	*Активність бізнес-процесу*	*Опис активності бізнес-процесу*	*Основні виконавці*	*Реалізація у інформаційній технології*
1	1. Аналіз та проєктування	1.1. Інформація про проєкт	Для оцінювання важливо зібрати та проаналізувати, зокрема, наступну інформацію про потенційний проєкт: а) інформація про клієнта та його бізнес; б) постановка проблеми; в) сфера бізнесу та різновид програмної системи; г) аналогічні проєкти.	Команда оцінювання	Інформаційна технологія дає можливість введення цих даних через графічний інтерфейс із наступним збереженням їх в базі даних.
2	1. Аналіз та проєктування	1.2. Збір та аналіз вимог	Ця активність передбачає, в першу чергу, формулювання функційних, а також нефункційних вимог до проєкту. Такі вимоги беруться за основу для побудови ІСЕО, яка використовується під час оцінювання.	Бізнес-аналітики	В інформаційній технології не передбачені функції для безпосереднього збору та аналізу вимог. При цьому варто підкреслити, що інформаційна технологія включає роботу з ІСЕО, яка будується на основі зібраних вимог до проєкту.
3	1. Аналіз та проєктування	1.3. Проєктування архітектури	Для оцінювання важливо розуміти, як буде реалізовуватися проєкт, для чого, зокрема, необхідно: а) зібрати та проаналізувати архітектурно значущі вимоги;	Технічні експерти	Не планується використання інформаційної технології для проєктування архітектури, однак передбачена можливість завантаження

№	Етап бізнес-процесу	Активність бізнес-процесу	Опис активності бізнес-процесу	Основні виконавці	Реалізація у інформаційній технології
			б) візуалізувати архітектуру, розробивши архітектурні діаграми; в) обрати технології реалізації проєкту.		архітектурних документів у вигляді файлів.
4	1. Аналіз та проєктування	1.4. Ідентифікація сценаріїв реалізації проєкту	Може бути визначено декілька потенційних сценаріїв реалізації проєкту, для кожного з яких проводиться оцінювання. Сценарії, для прикладу, можуть базуватися на різних технологіях реалізації або включати відмінні набори функційних можливостей.	Команда оцінювання	В інформаційній технології передбачені можливості для роботи із сценаріями реалізації проєкту.
5	2. Оцінювання	2.1. Ієрархічна структура елементів оцінювання	Побудова ICEO зводиться до визначення елементів оцінювання та їх атрибутів (див. підрозділ 2.2 та додаток 5).	Бізнес-аналітики	В інформаційній технології передбачена функціональність для роботи із ICEO.
6	2. Оцінювання	2.2. Оцінювання трудоємності	Оцінювання трудоємності має за мету надання НОР для елементів оцінювання, що входять до ICEO.	Технічні експерти	В інформаційній технології передбачена функціональність для оцінювання трудоємності.
7	2. Оцінювання	2.3. Склад проєктної команди	Ця активність включає вибір шаблону складу проєктної команди та визначення складу проєктної команди згідно обраного шаблону (див. підрозділ 2.6). Варто підкреслити, що шаблон складу проєктної команди пов’язаний із шаблоном графіку реалізації проєкту.	Менеджери	В інформаційній технології передбачена функціональність для роботи зі складом проєктної команди, зокрема, реалізовано метод підтримки прийняття рішень, описаний у підрозділі 2.11.
8	2. Оцінювання	2.4. Оцінювання графіку реалізації проєкту	Ця активність передбачає вибір шаблону графіку реалізації проєкту та визначення графіку реалізації проєкту	Менеджери	В інформаційній технології передбачена функціональність для оцінювання графіку реалізації проєк-

№	*Етап бізнес-процесу*	*Активність бізнес-процесу*	*Опис активності бізнес-процесу*	*Основні виконавці*	*Реалізація у інформаційній технології*
			згідно цього шаблону. При цьому шаблон графіку реалізації проєкту пов'язаний із шаблоном складу проєктної команди.		ту. Зокрема, до цього застосовний метод підтримки прийняття рішень щодо складу проєктної команди та графіку реалізації проєкту (див. підрозділ 2.11).
9	2. Оцінювання	2.5. Оцінювання ризиків	Під час аналізу ризиків рекомендується оцінити їх ймовірність та потенційний вплив на перебіг реалізації проєкту. Відзначимо, що розробка методів та засобів оцінювання ризиків не входить до завдань цієї дисертаційної роботи.	Команда оцінювання	В інформаційній технології ризики визначаються як атрибути елементів оцінювання.
10	2. Оцінювання	2.6. Оцінювання інших ресурсів	До таких ресурсів відносяться, наприклад, апаратне забезпечення, ліцензії на програмне забезпечення, хмарні сервіси тощо.	Команда оцінювання	Інформаційна технологія не передбачає функцій для оцінювання такого типу ресурсів. При цьому є можливість завантаження файлів, що містять деталі відповідних оцінок.
11	2. Оцінювання	2.7. Оцінювання бюджету	Оцінювання бюджету є однією із ключових завдань бізнес-процесу оцінювання. Підкреслимо, що розробка методів та засобів оцінювання бюджету проєкту не входить до завдань цієї дисертаційної роботи.	Менеджери	Інформаційна технологія не передбачає функцій для оцінювання бюджету. При цьому передбачена можливість завантаження результатів оцінювання бюджету у вигляді файлів.
12	3. Комунікація та зворотний зв'язок	3.1. Порівняння оцінок для різних сценаріїв реалізації проєкту	Порівняння оцінок для різних сценаріїв здійснюється на ранніх етапах оцінювання. Завданням такого порівняння є вибір перспективніших сценаріїв реалізації. В ідеальному	Команда оцінювання	Інформаційна технологія лише базові функції для порівняння оцінок для різних сценаріїв. При цьому не передбачено застосування методів

№	Етап бізнес-процесу	Активність бізнес-процесу	Опис активності бізнес-процесу	Основні виконавці	Реалізація у інформаційній технології
			випадку на етапі детального оцінювання повинен залишитися один із альтернативних сценаріїв.		підтримки прийняття рішень для порівняння.
13	3. Комунікація та зворотний зв'язок	3.2. Перевірка оцінки зовнішніми експертами	Метою цієї активності є отримання зворотного зв'язку щодо оцінки від зовнішніх експертів (які не входять до команди оцінювання).	Зовнішні (відносно команди оцінювання) експерти	Інформаційна технологія передбачає функції перегляду оцінки та надання зворотного зв'язку зовнішніми експертами.
14	3. Комунікація та зворотний зв'язок	3.3. Затвердження оцінки	У випадку проєкту певного обсягу необхідне затвердження оцінки керівництвом.	Керівництво	Інформаційна технологія передбачає ієрархічне затвердження оцінки керівництвом.
15	3. Комунікація та зворотний зв'язок	3.4. Комунікація оцінки зацікавленим сторонам	Комунікація оцінки зацікавленим сторонам відбувається у довільній формі, залежно від конкретного випадку, наприклад, у формі презентації чи через електронну пошту.	Менеджери	В інформаційній технології не передбачені функції комунікації оцінки зацікавленим сторонам.
16	3. Комунікація та зворотний зв'язок	3.5. Опрацювання зворотного зв'язку	Опрацювання зворотного зв'язку, отриманого в результаті ознайомлення з оцінкою зацікавлених сторін, здійснюється всією командою оцінювання. За необхідності робляться зміни в оцінці відповідно до отриманого зворотного зв'язку.	Команда оцінювання	В інформаційній технології передбачені функції опрацювання зворотного зв'язку щодо оцінки.

Додаток 7. Характеристики методу побудови схеми слабоструктурованого бізнес-процесу

Таблиця Д-7.1

Характеристики методу побудови схеми слабоструктурованого бізнес-процесу

№	*Назва характеристики англ. мовою*	*Опис характеристики*	*Кратність*	*Лог-метрика*	*Агрегатна*	*Розроблена в цій дисертації*
1	Event Class Frequency Significance Characteristic	Визначення відносної важливості класу подій на основі частоти появи відповідних подій у event-даних.	Унарна	Лог-характеристика	Ні	Ні
2	Routing Significance Characteristic	Визначення важливості класу подій з точки зору структури схеми бізнес-процесу на основі вхідних та вихідних залежностей. Згідно логіки цієї характеристики вищу важливість мають ті класи подій, у яких гілки схеми бізнес-процесу сходяться або розгалужуються.	Унарна	Похідна характеристика	Ні	Ні
3	Aggregate Unary Characteristic	Агрегація значення унарних лог-характеристик, враховуючи вагу кожної з них.	Унарна	Лог-характеристика	Так	Ні
4	Drift Unary Log Characteristic	Агрегація значення унарних лог-характеристик, враховуючи еволюцію схеми бізнес-процесу.	Унарна	Лог-характеристика	Так	Так
5	Frequency Binary Significance Characteristic	Визначення відносної важливості переходу між двома класами подій на основі того, як часто один з них слідує за іншим.	Бінарна	Лог-характеристика	Ні	Ні
6	Distance Significance Characteristic	Визначення важливості переходу між двома класами подій на основі різниці між важливістю переходу між цими двома класами подій та важливістю цих класів подій.	Бінарна	Похідна характеристика	Ні	Ні

№	*Назва характеристики англ. мовою*	*Опис характеристики*	*Кратність*	*Лог-метрика*	*Агрегатна*	*Розроблена в цій дисертації*
7	Proximity Correlation Characteristic	Визначає «близькість» двох класів подій на основі часу їх виникнення: цим менша різниця в часі виникнення подій, тим «ближчими» вважаються відповідні класи подій.	Бінарна	Лог-характе-ристика	Ні	Ні
8	Originator Correlation Characteristic	Визначає «близькість» двох класів подій на основі того, чи виконувалися вони одним і тим ж виконавцем.	Бінарна	Лог-характе-ристика	Ні	Ні
9	Endpoint Correlation Characteristic	Визначає «близькість» двох класів подій на основі «схожості» їх імен.	Бінарна	Лог-метрика	Ні	Ні
10	Data Type Correlation Characteristic	Визначає «близькість» двох класів подій на основі «схожості» типів атрибутів даних, пов’язаних з відповідними подіями.	Бінарна	Лог-характе-ристика	Ні	Ні
11	Data Value Correlation Characteristic	Визначає «близькість» двох класів подій на основі «схожості» значень атрибутів даних, пов’язаних з відповідними подіями.	Бінарна	Лог-характе-ристика	Ні	Ні
12	Aggregate Binary Characteristic	Агрегація значення бінарних лог-характерис-тик, враховуючи вагу кожної з них.	Бінарна	Лог-характе-ристика	Так	Ні
13	Drift Binary Log Characteristic	Агрегація значень бінарних лог-характеристик, враховуючи еволюцію схеми бізнес-процесу.	Бінарна	Лог-характе-ристика	Так	Так

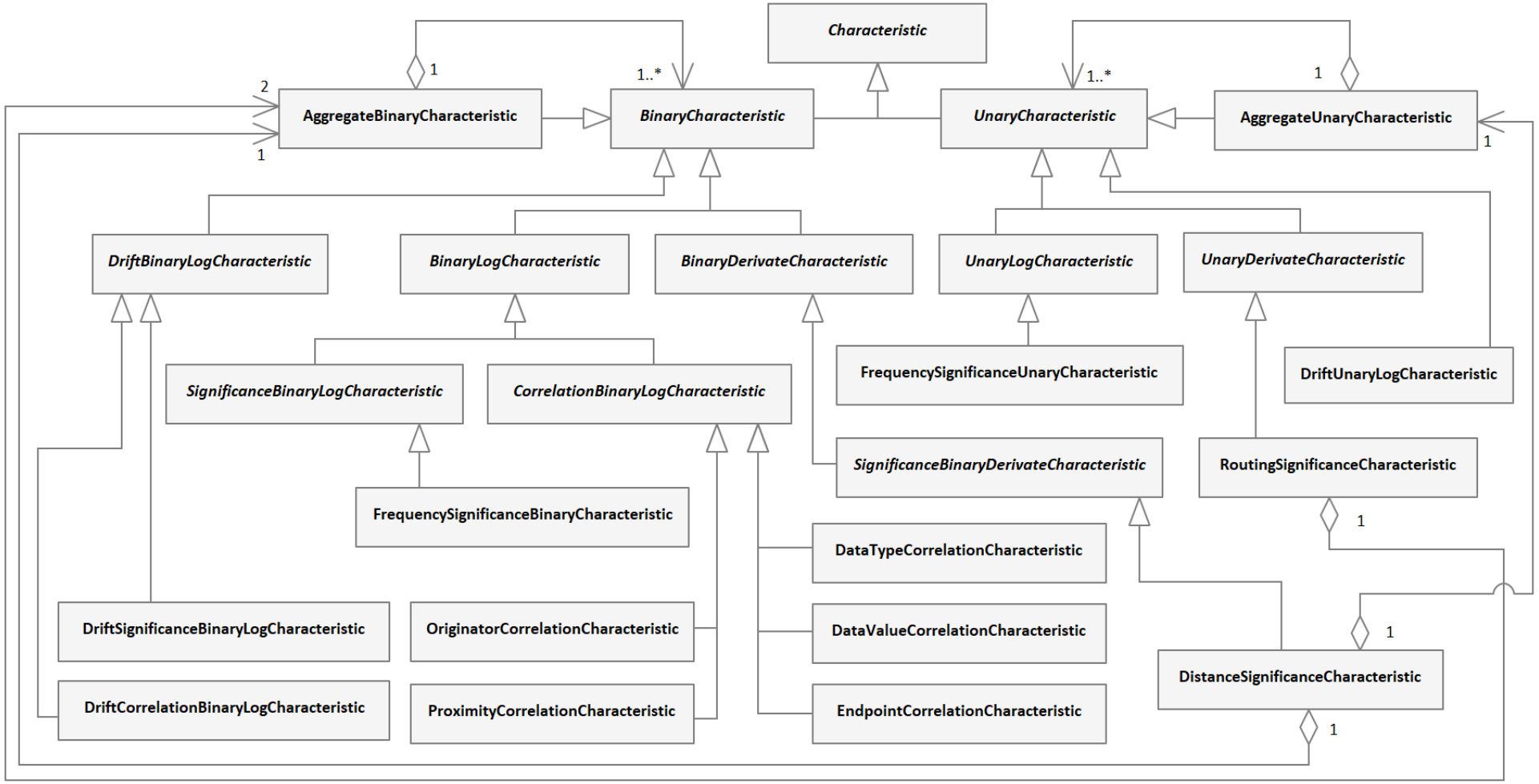


Рис. Д-7.1. UML-діаграма класів характеристик методу побудови схеми слабоструктурованого бізнес-процесу[1]

[1] На UML-діаграмі класів відображено лише класи та відношення між ними, при цьому поля та методи класів приховані з метою лаконічності.

Додаток 8. Структура бази даних реалізації методу побудови схеми слабоструктурованого бізнес-процесу

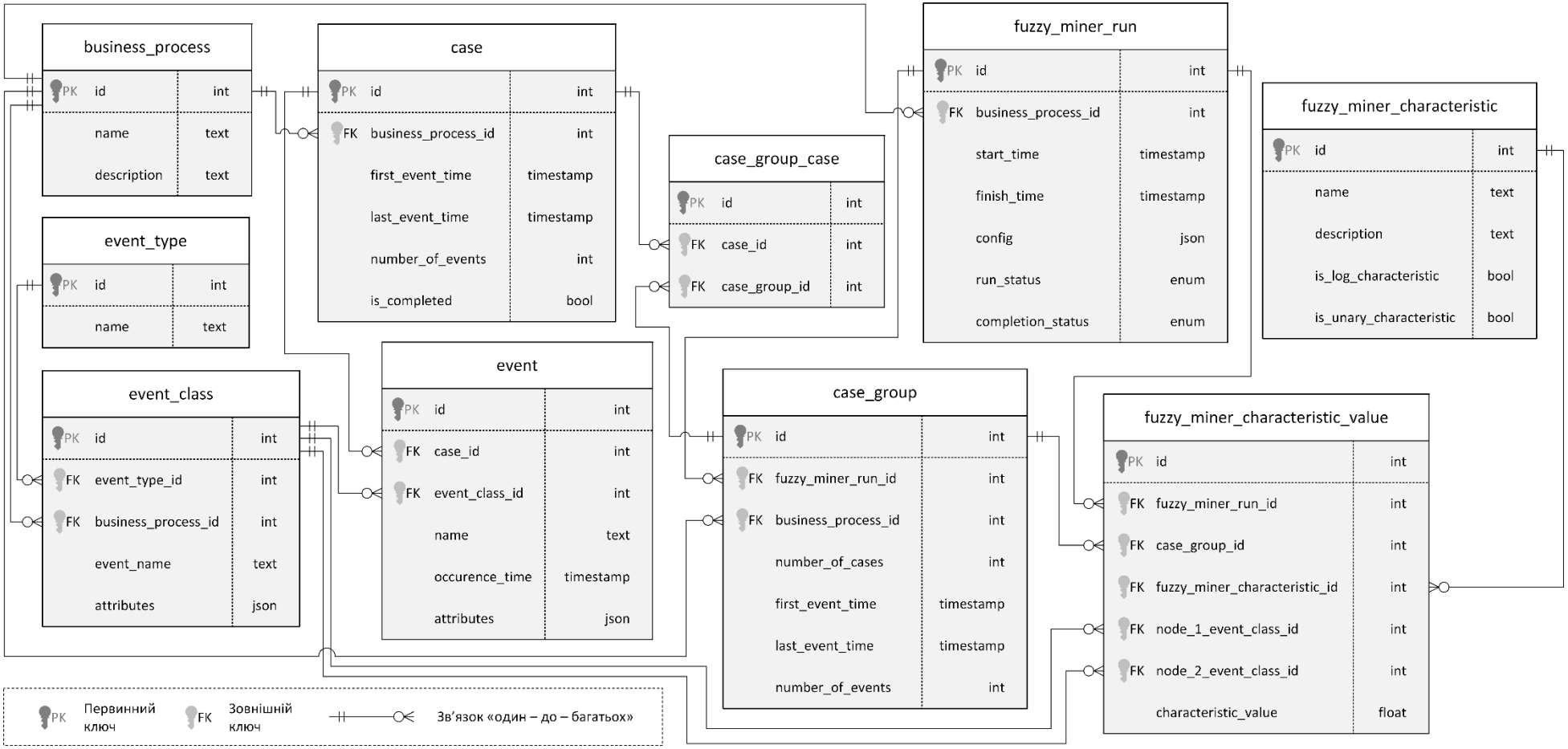


Рис. Д-8.1. Структура даних реалізації методу побудови схеми слабоструктурованого бізнес-процесу

Таблиця Д-8.1

Структура даних реалізації методу побудови схеми слабоструктурованого бізнес-процесу

№	*Назва таблиці*	*Опис таблиці*
1	business_process	Різновид бізнес-процесу, з яким пов’язані екземпляри бізнес-процесу (таблиця case)
2	event_type	Тип події: початок, завершення, основна активність
3	event_class	Клас події
4	case	Екземпляр бізнес-процесу
5	event	Подія в екземплярі бізнес-процесу
6	case_group	Група екземплярів бізнес-процесу
7	case_group_case	Відношення «багато-до-багатьох» між таблицями trace та trace_group
8	fuzzy_miner_run	Запуск потокової або пакетної версії розробленого методу побудови схеми бізнес-процесу
9	fuzzy_miner_characteristic	Довідник характеристик Fuzzy Miner (див. додаток 7)
10	fuzzy_miner_ characteristic_value	Значення характеристик Fuzzy Miner

Додаток 9. Попереднє оцінювання програмного забезпечення інформаційної технології

Таблиця Д-9.1

Ієрархічна структура елементів оцінювання та їх атрибути для попереднього оцінювання[1]

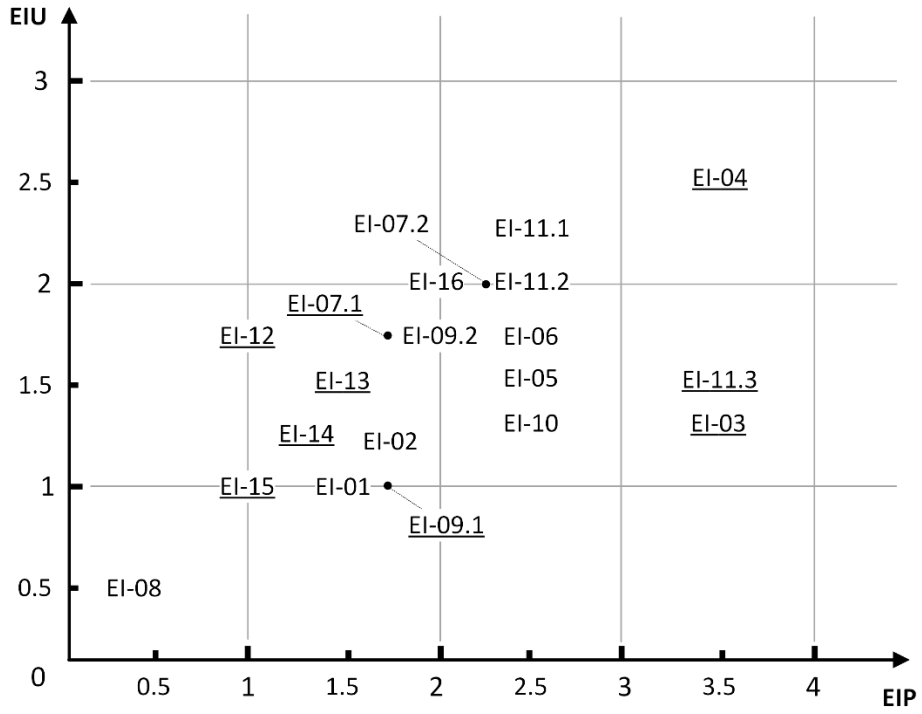
№	*Іденти-фікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
1	EI-01	–	Проєкти та сценарії їх реалізації	[FR] CRUD-операції над проєктами. [FR] CRUD-операції над сценаріями реалізації проєктів; один проєкт може мати кілька потенційних сценаріїв реалізації. [NFR] Наповнення бази даних проєктів з метою застосування машинного навчання у майбутніх версіях.
2	EI-02	–	Оцінки проєктів	[FR] CRUD-операції над оцінками сценаріїв. [FR] Збереження різних версій оцінки. [FR] Порівняння оцінок.
3	EI-03	–	Побудова ICEO	[FR] CRUD-операції над елементами оцінювання. [FR] CRUD-операції над атрибутами елементів оцінювання. [FR] Представлення ICEO у табличному та деревовидному виглядах. [FR] Вибір фрагментів ICEO з бібліотеки шаблонів. [RISK] Представлення ICEO у деревовидному вигляді засобами технології Microsoft Power Platform. [OOS] Представлення ICEO у формах, відмінних від табличної та деревовидної.
4	EI-04	–	Оцінювання трудоємності групування за спорідне-ністю	[FR] Візуалізація елементів оцінювання на двовимірній координатній площині та визначення значень атрибутів EIP та EIU. [FR] Розрахунок оптимістичної та песимістичної HOP.

[1] Символічні позначення типів атрибутів елементів оцінювання описані у додатку 5.

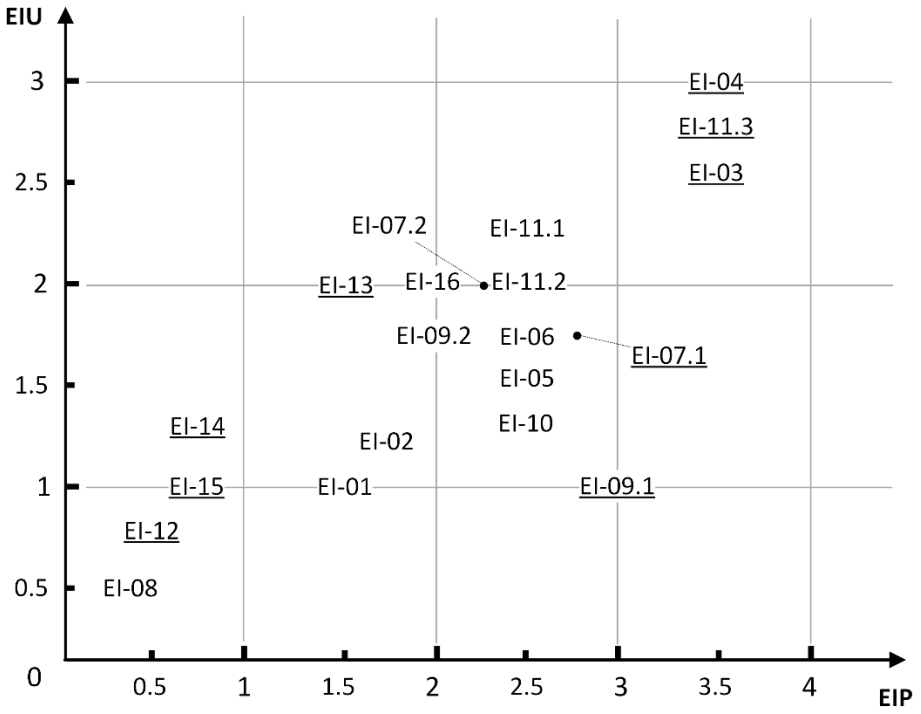
№	*Ідентифікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
				[RISK] Реалізація координатної площини з можливістю переміщення елементів оцінювання засобами технології Microsoft Power Platform.
5	EI-05	–	Склад проєктної команди	[FR] CRUD-операції щодо ролей часників команди. [FR] Варіативна частина складу проєктної команди та правила зміни цієї варіативної частини. [FR] Вибір шаблону складу проєктної команди.
6	EI-06	–	Графік реалізації проєкту	[FR] Визначення ієрархічної структури фаз проєкту, кількість рівнів ієрархії якої залежить від типу оцінювання та специфіки проєкту. [FR] Варіативна частина графіку реалізації проєкту.
7	EI-07	–	Розклад реалізації елементів оцінювання	
8	EI-07.1	EI-07	Метод побудови розкладу реалізації елементів оцінювання	[FR] Автоматична побудова розкладу реалізації елементів оцінювання із застосуванням методу, описаного в підрозділі 2.10. [TECH] Використати Python в якості технології реалізації. [RISK] Реалізація виклику Python-функції засобами технології Microsoft Power Platform.
9	EI-07.2	EI-07	Графічний інтерфейс розкладу реалізації елементів оцінювання	[FR] Визначення атрибутів елементів оцінювання, необхідних для побудови розкладу їх реалізації. [FR] Можливість змінювати вже побудований розклад реалізації елементів оцінювання. [FR] Візуалізація розкладу реалізації елементів оцінювання у вигляді діаграми Ганта. [FR] Розклад реалізації елементів оцінювання зберігається в базі даних інформаційної технології. [RISK] Реалізація діаграми Ганта засобами технологій ReactJS та Microsoft Power Platform.
10	EI-08	–	Визначення КРЧР для попереднього оцінювання	[FR] Значення КРЧР визначається за результатами експертної оцінки.

№	*Ідентифікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
				[OOS] Інструмент підтримки прийняття рішень, що пропонує значення КРЧР, враховуючи специфіку проєкту.
11	EI-09	–	Підтримка прийняття рішень щодо графіку реалізації проєкту та складу команди	
12	EI-09.1	EI-09	Метод підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди	[FR] Реалізації методу підтримки прийняття рішень, описаного у підрозділі 2.11. [TECH] Використати Python в якості технології реалізації. [RISK] Реалізація виклику Python-функції засобами технології Microsoft Power Platform.
13	EI-09.2	EI-09	Графічний інтерфейс підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди	[FR] Результати застосування методу підтримки прийняття рішень зберігаються в базі даних інформаційної технології.
14	EI-10	–	Зворотний зв'язок щодо оцінки	[FR] Зворотний зв’язок від експертів, що здійснюють перевірку оцінки. [FR] Затвердження оцінки керівництвом. [FR] Збір зворотного зв’язку від зацікавлених сторін (наприклад, замовників).
15	EI-11	–	Бібліотека шаблонів	[DESC] Зібрання компонентів, що призначені для повторного використання.
16	EI-11.1	EI-11	Шаблони графіку реалізації проєкту	[FR] CRUD-операції щодо шаблонів графіку реалізації проєкту. [FR] Зберігання графіку реалізації проєкту у бібліотеці шаблонів.
17	EI-11.2	EI-11	Шаблони складу проєктної команди	[FR] CRUD-операції щодо шаблонів складу проєктної команди. [FR] Зберігання складу проєктної команди у бібліотеку шаблонів.
18	EI-11.3	EI-11	Шаблони ICEO	[FR] CRUD-операції щодо шаблонів фрагментів ICEO.

№	Ідентифікатор	Батьківський елемент оцінювання	Заголовок елементу оцінювання	Атрибути елементу оцінювання
				[FR] Зберігання фрагментів ICEO у бібліотеці шаблонів. [RISK] Представлення ICEO у деревовидному вигляді засобами технології Microsoft Power Platform.
19	EI-12	–	Безпека	[FR] Аутентифікація. [FR] Авторизація. [FR] Управління користувачами [FR] Профіль користувача. [ARCH] Аутентифікація здійснюється через сервіс аутентифікації організації. Реєстрація користувача, зміна паролю, двофакторна аутентифікація реалізуються системою безпеки організації.
20	EI-13	–	Збір аудит-логів	[ARCH] Аудит-логи надсилаються в режимі, наближеному до реального часу, у модуль RTBPM-E і не зберігаються в базі даних інформаційної технології. [NFR] Зібрані аудит-логи повинні давати змогу здійснювати моніторинг проходження бізнес-процесу оцінювання засобами процес-майнингу (функції модуля RTBPM-E).
21	EI-14	–	База даних інформаційної технології	[DESC] База даних необхідна для накопичення даних щодо проєктів та їх оцінок, що дасть змогу в перспективі, накопичивши достатню кількість даних, засовувати методи машинного навчання для підвищення ефективності методів оцінювання.
22	EI-15	–	Налаштування базової структури проєкту	
23	EI-16	–	Невідома частина змісту проєктних робіт	[DESC] Частина змісту проєктних робіт, невідома на етапі попереднього оцінювання.



(а) – сценарій реалізації 1:
реалізація за підходом «pro-code»



(б) – сценарій реалізації 2:
реалізація за підходом «low-code»

Рис. Д-9.1. Попередня оцінка трудоємності групуванням за схожістю[1]

[1] На рис. Д-9.1 для підкреслених елементів оцінювання відповідні оцінки для двох сценаріїв відрізняються, а для непідкреслених елементів відповідні оцінки співпадають.

Таблиця Д-9.2

Нормалізована оцінка розробки: попереднє оцінювання[1]

№	Елемент оцінювання	Сценарій реалізації 1					Сценарій реалізації 2				
		EIP	Базова НОР, людино-години	EIU	Оптимістична НОР, людино-години	Песимістична НОР, людино-години	EIP	Базова НОР, людино-години	EIU	Оптимістична НОР, людино-години	Песимістична НОР, людино-години
1	EI-01	1.5	105.0	1	89.3	131.3	1.5	60.0	1	51.0	75.0
2	EI-02	1.75	122.5	1.2	100.5	159.3	1.75	70.0	1.2	57.4	91.0
3	EI-03	3.5	245.0	1.25	199.1	321.6	3.5	140.0	2.5	87.5	227.5
4	EI-04	3.5	245.0	2.5	153.2	398.2	3.5	140.0	3	77.0	245.0
5	EI-05	2.5	175.0	1.5	135.7	240.7	2.5	100.0	1.5	77.5	137.5
6	EI-06	2.5	175.0	1.75	129.1	251.6	2.5	100.0	1.75	73.8	143.8
7	EI-07.1	1.5	105.0	1.75	77.5	151.0	2.75	110.0	1.75	81.2	158.2
8	EI-07.2	2.25	157.5	2	110.3	236.3	2.25	90.0	2	63.0	135.0
9	EI-08	0.3	21.0	0.5	19.5	23.7	0.3	12.0	0.5	11.1	13.5
10	EI-09.1	1.75	122.5	1	104.2	153.2	3	120.0	1	102.0	150.0
11	EI-09.2	2	140.0	1.75	103.3	201.3	2	80.0	1.75	59.0	115.0
12	EI-10	2.5	175.0	1.75	129.1	251.6	2.5	100.0	1.75	73.8	143.8
13	EI-11.1	2.5	175.0	2.25	116.0	273.5	2.5	100.0	2.25	66.3	156.3
14	EI-11.2	2.5	175.0	2	122.5	262.5	2.5	100.0	2	70.0	150.0
15	EI-11.3	3.5	245.0	1.5	189.9	336.9	3.5	140.0	2.75	82.3	236.3
16	EI-12	1	70.0	1.75	51.7	100.7	0.5	20.0	0.75	17.8	23.8

[1] Для отримання оптимістичної та песимістичної НОР використані наступні значення параметрів: $p = 70$, $u_1 = 0.15$, $u_2 = 0.25$ (див. пункт 2.7.5).

№	Елемент оцінювання	Сценарій реалізації 1					Сценарій реалізації 2				
		EIP	Базова НОР, людино-години	EIU	Оптиміс-тична НОР, людино-години	Песиміс-тична НОР, людино-години	EIP	Базова НОР, людино-години	EIU	Оптиміс-тична НОР, людино-години	Песиміс-тична НОР, людино-години
17	EI-13	1.5	105.0	1.5	81.4	144.4	1.5	60.0	2	42.0	90.0
18	EI-14	1.25	87.5	1.25	71.1	114.9	1.25	50.0	0.75	44.4	59.4
19	EI-15	1	70.0	1	59.5	87.5	1	40.0	0.75	35.5	47.5
20	EI-16	2	140.0	2	98.0	210.0	2	80.0	2	56.0	120.0
Сума		**–**	***2856.0***	**–**	***2140.9***	***4050.2***	**–**	***1712.0***	**–**	***1228.6***	***2518.6***

Таблиця Д-9.3

Попередня оцінка складу проєктної команди

№	*Категорія ролі*[1]	*Роль проєктної команди*	*Нормалізована собівартість*[2]	*КПРК*	*КПСЗ*	*Коефіцієнт продуктивності*	*Крок зміни ЕПЗ*	*Сценарій реалізації 1*				*Сценарій реалізації 2*			
								Оптимістичний ЕПЗ	*Оптимістичний НЕПЗ*	*Песимістичний ЕПЗ*	*Песимістичний НЕПЗ*	*Оптимістичний ЕПЗ*	*Оптимістичний НЕПЗ*	*Песимістичний ЕПЗ*	*Песимістичний НЕПЗ*
1	D	Старший інженер-розробник	1.00	1.2	0.9	1.08	0.50	0.5	0.54	0.5	0.54	0.5	0.54	0.5	0.54
2	D	Старший інженер-розробник	0.90	1.2	1	1.2	0.50	2.0	2.40	2.0	2.40	1.0	1.20	2.0	2.40
3	D	Інженер-розробник	0.70	1	1	1	1.00	5.0	5.00	6.0	6.00	3.0	3.00	4.0	4.00
4	D	Молодший інженер-розробник	0.50	0.7	1	0.7	1.00	3.0	2.10	4.0	2.80	2.0	1.40	2.0	1.40
5	ND	Проєктний менеджер	0.95	–	–	–	0.50	1.0	–	1.0	–	1.0	–	1.0	–
6	ND	Технічний лідер[3]	1.00	–	–	–	0.50	0.5	–	0.5	–	0.5	–	0.5	–
7	ND	Бізнес-аналітик	0.80	–	–	–	0.50	1.0	–	2.0	–	1.0	–	2.0	–
8	ND	Дизайнер	0.80	–	–	–	0.25	1.0	–	1.0	–	1.0	–	1.0	–
9	ND	DevOps-інженер	0.80	–	–	–	0.50	1.0	–	1.0	–	0.5	–	0.5	–
10	ND	Інженер-тестувальник	0.60	–	–	–	0.50	3.0	–	4.0	–	3.0	–	4.0	–
Сума		–	–	–	–	–	–	***18.0***	***10.04***	***22.0***	***11.74***	***13.5***	***6.14***	***17.5***	***8.34***

[1] Пояснення позначень категорій проєктних ролей: D – роль інженера-розробника, ND – «нерозробницька» роль (див. підрозділ 2.6).

[2] Нормалізована собівартість – це відносна собівартість 1 людино-години певного учасника команди (див. пункт 2.11.1).

[3] Роль технічного лідера виконує старший інженер-розробник 1 з таким розподілом робочого часу: 50% – обов'язки технічного лідера, 50% – обов'язки старшого інженера-розробника.

Таблиця Д-9.4

Попередні оцінки тривалості проєкту, нормалізованої спроможності розробки,
робочого часу проєктної команди та нормалізованої собівартості

№	Характеристика	Сценарій реалізації 1		Сценарій реалізації 2		Різниця (сценарій 1 – сценарій 2)	
		Оптимістична оцінка	*Песимістична оцінка*	*Оптимістична оцінка*	*Песимістична оцінка*	*Оптимістична оцінка*	*Песимістична оцінка*
1	НОР, людино-години	2140.9	4050.2	1228.6	2518.6	912.30 (+42.61%)	1531.60 (+37.82%)
2	КРЧР	3.8	3.8	3.8	3.8	–	–
3	ЕПЗ розробників	10.5	12.5	6.5	8.5	4.00 (+38.10%)	4.00 (+32.00%)
4	Нормалізований ЕПЗ розробників	10.04	11.74	6.14	8.34	3.90 (+38.84%)	3.40 (+28.96%)
5	ЕПЗ проєктної команди	18.0	22.0	13.5	17.5	4.50 (+25.00%)	4.50 (+20.45%)
6	Оцінка тривалості у годинах	810.4	1311.0	760.4	1147.6	50.00 (+6.17%)	163.40 (+12.46%)
7	Кількість робочих днів у місяці	21	21	21	21	–	–
8	Кількість робочих годин у місяці	168	168	168	168	–	–
9	Оцінка тривалості у місяцях	4.82	7.80	4.53	6.83	0.30 (+6.17%)	0.97 (+12.46%)
10	Округлена оцінка тривалості у місяцях	5.0	8.0	5.0	7.0	0.00 (0.00%)	1.00 (+12.50%)
11	Округлена оцінка тривалості у годинах	840.0	1344.0	840.0	1176.0	0.00 (0.00%)	168.00 (+12.50%)
12	НСР, людино-години	2219.4	4152.3	1357.3	2581.0	862.11 (+38.84%)	1571.24 (+37.84%)

№	Характеристика	Сценарій реалізації 1		Сценарій реалізації 2		Різниця (сценарій 1 – сценарій 2)	
		Оптимістична оцінка	*Песимістична оцінка*	*Оптимістична оцінка*	*Песимістична оцінка*	*Оптимістична оцінка*	*Песимістична оцінка*
13	Нормалізований час простою, людино-години	78.5	102.1	128.7	62.4	–	–
14	Проєктний робочий час розробників, людино-години	8820.0	16800.0	5460.0	9996.0	3360.00 (+38.10%)	6804.00 (+40.50%)
15	Проєктний робочий час всієї команди, людино-години	15120.0	29568.0	11340.0	20580.0	3780.00 (+25.00%)	8988.00 (+30.40%)
16	Нормалізована собівартість	10878.00	20899.20	8190.00	14994.00	2688.00 (+24.71%)	5905.20 (+28.26%)

Таблиця Д-9.5

Обмеження для задач оптимізації методу підтримки прийняття рішень щодо складу команди та тривалості проєкту

№	*Обмеження*	*Пояснення*
1	$3\phi_4 \le \phi_1 + \phi_2 + \phi_3$	Обмеження визначають співвідношення між ЕПЗ молодших інженерів-розробників та ЕПЗ інженерів-розробників, рівень компетентності яких є вищим за молодшого інженера-розробника.
2	$\phi_1 + \phi_2 + \phi_3 \le 4\phi_4$	
3	$2\phi_1 + 2\phi_2 \le \phi_3$	Обмеження визначають співвідношення між ЕПЗ інженерів-розробників та ЕПЗ старших інженерів-розробників.
4	$\phi_3 \le 4\phi_1 + 4\phi_2$	
5	$\phi_1 + \phi_2 + \phi_3 + \phi_4 + \phi_6 + \phi_7 + \phi_8 + \phi_9 + \phi_{10} \le 15\phi_5$	На 1 ЕПЗ проєктного менеджера повинно припадати не більше як 15 ЕПЗ всіх інших проєктних ролей.
6	$\phi_1 + \phi_2 + \phi_3 + \phi_4 \le 10\phi_7$	На 1 ЕПЗ бізнес-аналітика повинно припадати не більше як 10 ЕПЗ інженерів-розробників.
7	$\phi_1 + \phi_2 + \phi_3 + \phi_4 \le 4\phi_{10}$	На 1 ЕПЗ інженера-тестувальника повинно припадати не більше як 4 ЕПЗ інженерів-розробників.
8	$0.5\phi_2 + 0.5\phi_3 + 0.5\phi_4 \le 7\phi_6$	На 0.5 ЕПЗ технічного лідера повинно припадати не більше як 7 ЕПЗ інженерів-розробників.
9	$\phi_1 + \phi_2 + \phi_3 + \phi_4 \le 12\phi_8$	На 1 ЕПЗ дизайнера повинно припадати не більше як 12 ЕПЗ інженерів-розробників.
10	$0.5\phi_1 + 0.5\phi_2 + 0.5\phi_3 + 0.5\phi_4 \le 10\phi_9$	На 0.5 ЕПЗ DevOps-інженера не повинен припадати більше як 10 ЕПЗ інженерів-розробників.
11	$\phi_1 \le \phi_6$	Старший інженер-розробник повинен виконувати роль технічного лідера на 0.5.
12	$\phi_6 \le \phi_1$	
13	$\phi_1 \ge 0.5$	В команді повинен бути хоча б 1 інженер-розробник, що виконує на 0.5 навантаження роль технічного лідера.
14	$\phi_6 \ge 0.5$	
15	$\phi_5 \ge 1$	ЕПЗ проєктних менеджерів не повинно бути меншим за 1.

№	*Обмеження*	*Пояснення*
16	$\phi_7 \geq 1$	ЕПЗ бізнес-аналітиків не повинно бути меншим за 1.
17	$\phi_8 \geq 0.25$	ЕПЗ дизайнерів не повинно бути меншим за 0.25.
18	$\phi_9 \geq 0.5$	ЕПЗ DevOps-інженерів не повинно бути меншим за 0.5.
19	$\phi_{10} \geq 1$	ЕПЗ інженерів-тестувальників не повинно бути меншим за 1.
20	$\phi_1 + \phi_2 + \phi_3 + \phi_4 + \phi_5 + \phi_6 + \phi_7 + \phi_8 + \phi_9 + \phi_{10} \leq 25$	Розмір проєктної команди не повинен перевищувати 25 ЕПЗ.

Таблиця Д-9.6

Критерії порівняння альтернатив для методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту

	Нормалізована собівартість	***Тривалість***	***Розмір команди***	***Нормалізований час простою***
Нормалізована собівартість	1	3	3	7
Тривалість	$\frac{1}{3}$	1	$\frac{1}{7}$	7
Розмір команди	$\frac{1}{3}$	7	1	5
Нормалізований час простою	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{5}$	1

Таблиця Д-9.7.

Альтернативи попередніх оцінок для сценарію реалізації 1,

отримані із застосуванням методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту[1,2]

№	***Тривалість, місяці***	***Оптимістична оцінка***					***Песимістична оцінка***				
		Норм. собівартість	*ЕПЗ команди*	*НСР команди*	*Норм. час простою*	*Ранг*	*Норм. собівартість*	*ЕПЗ команди*	*НСР команди*	*Норм. час простою*	*Ранг*
1	4	–	–	–	–	–	–	–	–	–	–
2	4.5	12833.1	23.25	2582.34	441.44	0.060	–	–	–	–	–
3	5	13671.0	22.25	2648.21	507.31	0.052	–	–	–	–	–
4	5.5	14160.3	21	2767.14	626.24	0.047	–	–	–	–	–
5	6	**13129.2**	**17.5**	**2567.75**	**426.85**	**0.059**	–	–	–	–	–
6	6.5	13731.9	17	2609.31	468.41	0.052	–	–	–	–	–
7	7	13700.4	15.75	2686.23	545.33	0.054	–	–	–	–	–
8	7.5	14112.0	15.25	2679.16	538.26	0.051	–	–	–	–	–
9	8	14313.6	14.25	2716.29	575.39	0.049	**23419.2**	**23.75**	**4803.03**	**752.83**	**0.098**
10	8.5	15208.2	14.25	2886.06	745.16	0.041	24240.3	23.25	4877.75	827.55	0.082
11	9	16102.8	14.25	3055.83	914.93	0.035	24985.8	22.75	4925.94	875.74	0.065

[1] Обрані альтернативи для оптимістичної та песимістичної оцінок виділені півжирним.

[2] Для реалізації методу підтримки прийняття рішень було використано Python 3.10.7, FICO Xpress 9.4.1, Pyomo 6.7.3, Ahpy 2.0.

№	Трива-лість, місяці	Оптимістична оцінка					Песимістична оцінка				
		Норм. собівар-тість	ЕПЗ команди	НСР команди	Норм. час простою	Ранг	Норм. собівар-тість	ЕПЗ команди	НСР команди	Норм. час простою	Ранг
12	9.5	16997.4	14.25	3225.60	1084.70	0.031	25974.9	22.25	5031.60	981.40	0.049
13	10	17892.0	14.25	3395.37	1254.47	0.028	25746.0	21	5031.16	980.96	0.054
14	10.5	18786.6	14.25	3565.14	1424.24	0.025	27033.3	21	5282.72	1232.52	0.038
15	11	19681.2	14.25	3734.91	1594.01	0.023	28320.6	21	5534.27	1484.07	0.030
16	11.5	20575.8	14.25	3904.67	1763.77	0.021	25164.3	17.5	4921.52	871.32	0.066
17	12	15120.0	10	2567.75	426.85	0.051	25351.2	17	4817.18	766.98	0.067
18	12.5	15750.0	10	2674.74	533.84	0.047	26407.5	17	5017.89	967.69	0.050
19	13	16380.0	10	2781.73	640.83	0.043	25443.6	15.75	4988.72	938.52	0.069
20	13.5	17010.0	10	2888.72	747.82	0.040	26422.2	15.75	5180.59	1130.39	0.055
21	14	17640.0	10	2995.71	854.81	0.038	26342.4	15.25	5001.09	950.89	0.062
22	14.5	18270.0	10	3102.69	961.79	0.035	27283.2	15.25	5179.71	1129.51	0.052
23	15	18900.0	10	3209.68	1068.78	0.034	26838.0	14.25	5093.05	1042.85	0.061
24	15.5	16926.0	8.5	2631.41	490.51	0.045	27732.6	14.25	5262.82	1212.62	0.054
25	16	17472.0	8.5	2716.29	575.39	0.043	28627.2	14.25	5432.59	1382.39	0.050

Таблиця Д-9.8.

Альтернативи попередніх оцінок для сценарію реалізації 2,

отримані із застосуванням методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту[1,2]

№	Тривалість, місяці	Оптимістична оцінка					Песимістична оцінка				
		Норм. собівартість	*ЕПЗ команди*	*НСР команди*	*Норм. час простою*	*Ранг*	*Норм. собівартість*	*ЕПЗ команди*	*НСР команди*	*Норм. час простою*	*Ранг*
1	4	7828.8	15.75	1534.99	306.39	0.058	–	–	–	–	–
2	4.5	**8051.4**	**14.25**	**1527.92**	**299.32**	**0.057**	–	–	–	–	–
3	5	8946.0	14.25	1697.68	469.08	0.049	14637.0	23.75	3001.89	483.29	0.061
4	5.5	9840.6	14.25	1867.45	638.85	0.042	15453.9	22.75	3058.93	540.33	0.054
5	6	10735.2	14.25	2037.22	808.62	0.036	16405.2	22.25	3177.85	659.25	0.048
6	6.5	11629.8	14.25	2206.99	978.39	0.031	16734.9	21	3270.25	751.65	0.043
7	7	8820.0	10	1497.85	269.25	0.055	**15317.4**	**17.5**	**2995.71**	**477.11**	**0.058**
8	7.5	9450.0	10	1604.84	376.24	0.049	15844.5	17	3010.74	492.14	0.051
9	8	10080.0	10	1711.83	483.23	0.044	15657.6	15.75	3069.98	551.38	0.057
10	8.5	10710.0	10	1818.82	590.22	0.039	16636.2	15.75	3261.85	743.25	0.049
11	9	9828.0	8.5	1527.92	299.32	0.053	16934.4	15.25	3214.99	696.39	0.046

[1] Обрані альтернативи для оптимістичної та песимістичної оцінок виділені півжирним.

[2] Для реалізації методу підтримки прийняття рішень було використано Python 3.10.7, FICO Xpress 9.4.1, Pyomo 6.7.3, Ahpy 2.0.

№	Тривалість, місяці	Оптимістична оцінка					Песимістична оцінка				
		Норм. собівартість	ЕПЗ команди	НСР команди	Норм. час простою	Ранг	Норм. собівартість	ЕПЗ команди	НСР команди	Норм. час простою	Ранг
12	9.5	10374.0	8.5	1612.80	384.20	0.050	16997.4	14.25	3225.60	707.00	0.047
13	10	10920.0	8.5	1697.68	469.08	0.046	17892.0	14.25	3395.37	876.77	0.042
14	10.5	11466.0	8.5	1782.57	553.97	0.043	18786.6	14.25	3565.14	1046.54	0.037
15	11	12012.0	8.5	1867.45	638.85	0.040	19681.2	14.25	3734.91	1216.31	0.033
16	11.5	12558.0	8.5	1952.34	723.74	0.038	20575.8	14.25	3904.67	1386.07	0.030
17	12	13104.0	8.5	2037.22	808.62	0.036	21470.4	14.25	4074.44	1555.84	0.028
18	12.5	13650.0	8.5	2122.11	893.51	0.034	22365.0	14.25	4244.21	1725.61	0.026
19	13	14196.0	8.5	2206.99	978.39	0.033	23259.6	14.25	4413.98	1895.38	0.024
20	13.5	14742.0	8.5	2291.87	1063.27	0.031	24154.2	14.25	4583.75	2065.15	0.024
21	14	15288.0	8.5	2376.76	1148.16	0.030	17640.0	10	2995.71	477.11	0.056
22	14.5	15834.0	8.5	2461.64	1233.04	0.029	18270.0	10	3102.69	584.09	0.051
23	15	16380.0	8.5	2546.53	1317.93	0.028	18900.0	10	3209.68	691.08	0.048
24	15.5	16926.0	8.5	2631.41	1402.81	0.027	19530.0	10	3316.67	798.07	0.045
25	16	17472.0	8.5	2716.29	1487.69	0.027	20160.0	10	3423.66	905.06	0.044

Таблиця Д-9.9

Варіанти складу проєктної команди,

що відповідають альтернативам попередніх оцінок, обраним у табл. Д-9.7 та Д-9.8

№	*Категорія ролі*[1]	*Роль проєктної команди*	*Нормалізована собівартість*[2]	*КПРК*	*КПСЗ*	*Коефіцієнт продуктивності*	*Крок зміни ЕПЗ*	*Сценарій реалізації 1*				*Сценарій реалізації 2*			
								Оптимістичний ЕПЗ	*Оптимістичний НЕПЗ*	*Песимістичний ЕПЗ*	*Песимістичний НЕПЗ*	*Оптимістичний ЕПЗ*	*Оптимістичний НЕПЗ*	*Песимістичний ЕПЗ*	*Песимістичний НЕПЗ*
1	D	Старший інженер-розробник	1.00	1.2	0.9	1.08	0.50	1	1.08	1	1.08	1	1.08	1	1.08
2	D	Старший інженер-розробник	0.90	1.2	1	1.2	0.50	1	1.20	2	2.40	1	1.20	1	1.20
3	D	Інженер-розробник	0.70	1	1	1	1.00	6	6.00	8	8.00	4	4.00	6	6.00
4	D	Молодший інженер-розробник	0.50	0.7	1	0.7	1.00	2	1.40	3	2.10	2	1.40	2	1.40
5	ND	Проєктний менеджер	0.95	–	–	–	0.50	1.5	–	1.5	–	1	–	1.5	–
6	ND	Технічний лідер[3]	1.00	–	–	–	0.50	1	–	1	–	1	–	1	–
7	ND	Бізнес-аналітик	0.80	–	–	–	0.50	1	–	1.5	–	1	–	1	–
8	ND	Дизайнер	0.80	–	–	–	0.25	1	–	1.25	–	0.75	–	1	–
9	ND	DevOps-інженер	0.80	–	–	–	0.50	0.5	–	1	–	0.5	–	0.5	–
10	ND	Інженер-тестувальник	0.60	–	–	–	0.50	2.5	–	3.5	–	2	–	2.5	–
Сума		**–**		–	–	–	–	***17.50***	***9.68***	***23.75***	***13.58***	***14.25***	***7.68***	***17.50***	***9.68***

[1] Пояснення позначень категорій проєктних ролей: D – роль інженера-розробника, ND – «нерозробницька» роль (див. підпункт 2.6).

[2] Нормалізована собівартість – це відносна собівартість 1 людино-години певного учасника команди (див. пункт 2.11.1).

[3] Роль технічного лідера виконує старший інженер-розробник 1 з таким розподілом робочого часу: 50% – обов'язки технічного лідера, 50% – обов'язки старшого інженера-розробника.

Додаток 10. Проміжне оцінювання програмного забезпечення інформаційної технології

Таблиця Д-10.1

Ієрархічна структура елементів оцінювання та їх атрибути для проміжного оцінювання[1]

№	*Ідентифікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
1	EI-01	–	Проєкти та сценарії їх реалізації	[FR] CRUD-операції над проєктами. [FR] CRUD-операції над сценаріями реалізації проєктів; один проєкт може мати кілька потенційних сценаріїв реалізації. [NFR] Наповнення бази даних проєктів з метою застосування машинного навчання у майбутніх версіях.
2	EI-01.01	EI-01	Проєкти	[FR] Перегляд переліку проєктів у табличному вигляді. [FR] Перегляд інформації про обраний проєкт. [FR] Додавання нового проєкту. [FR] Редагування інформації про існуючий проєкт. [FR] Позначення існуючого проєкту «неактивним» (аналог видалення без реального видалення запису з бази даних). [FR] Інформація про проєкт включає до 15 полів даних, а також можливість завантаження файлів.
3	EI-01.02	EI-01	Сценарії реалізації проєкту	[FR] Перегляд переліку сценаріїв реалізації проєкту у табличному вигляді. [FR] Перегляд інформації про обраний сценарій реалізації проєкту. [FR] Додавання нового сценарію реалізації проєкту. [FR] Редагування інформації про існуючий сценарій реалізації проєкту. [FR] Позначення існуючого сценарію реалізації проєкту «неактивним» (аналог видалення без реального видалення запису з бази даних).

[1] Символічні позначення типів атрибутів елементів оцінювання описані у додатку 5.

№	*Ідентифікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
				[FR] Деревовидна структура сценаріїв реалізації проєкту, коли один сценарій може мати кілька «дочірніх» сценаріїв.
4	EI-02	–	Оцінки проєктів	[FR] CRUD-операції над оцінками сценаріїв. [FR] Збереження різних версій оцінки. [FR] Порівняння оцінок.
5	EI-02.01	EI-02	CRUD-операції над оцінками сценаріїв	[FR] Перегляд переліку оцінок у табличному вигляді. [FR] Перегляд інформації про обрану оцінку. [FR] Додавання нової оцінки. [FR] Редагування інформації про існуючу оцінку. [FR] Життєвий цикл оцінки (стани оцінки та переходи між ними).
6	EI-02.02	EI-02	Версії оцінки	[FR] Збереження поточної версії оцінки (що, наприклад, може бути використано для порівняння різних версій оцінок). [FR] Поетапність оцінок, коли одна оцінка є продовженням іншої.
7	EI-02.03	EI-02	Порівняння оцінок	[FR] Порівняння основних даних для двох чи більше оцінок. [OOS] Детальне порівняння розрахунків, пов’язаних з оцінками, не входить у цю задачу.
8	EI-03	–	Побудова ICEO	[FR] CRUD-операції над елементами оцінювання. [FR] CRUD-операції над атрибутами елементів оцінювання. [FR] Представлення ICEO у табличному та деревовидному виглядах. [FR] Вибір фрагментів ICEO з бібліотеки шаблонів. [OOS] Представлення ICEO у формах, відмінних від табличної та деревовидної.
9	EI-03.01	EI-03	CRUD-операції над елементами оцінювання	[FR] Додавання, редагування та видалення елементів оцінювання.
10	EI-03.02	EI-03	CRUD-операції над атрибутами елементів оцінювання	[FR] Додавання, редагування та видалення атрибутів елементів оцінювання.

№	Ідентифікатор	Батьківський елемент оцінювання	Заголовок елементу оцінювання	Атрибути елементу оцінювання
				[FR] Типи атрибутів елементів оцінювання обираються із статичного списку.
11	EI-03.03	EI-03	Представлення ICEO у табличному вигляді	[FR] Таблиця із переліком елементів оцінювання. [FR] Додавання, редагування та видалення елементів оцінювання. [FR] Групування, фільтрування та сортування елементів оцінювання. [FR] Експортування таблиці з елементами оцінювання у Excel-документ.
12	EI-03.04	EI-03	Представлення ICEO у деревовидному вигляді	[FR] Дерево елементів оцінювання. [FR] Додавання, редагування та видалення елементів оцінювання. [FR] Зміна батьківського елементу оцінювання перетягуванням.
13	EI-03.05	EI-03	Вибір фрагментів ICEO з бібліотеки шаблонів.	[FR] Фрагмент із бібліотеки шаблонів копіюється у ICEO; при зміни скопійованого фрагменту, початкова версія у бібліотеці шаблонів не змінюється.
14	EI-04	–	Оцінювання трудоємності групування за спорідненістю	[FR] Візуалізація елементів оцінювання на двовимірній координатній площині та визначення значень атрибутів EIP та EIU. [FR] Розрахунок оптимістичної та песимістичної HOP.
15	EI-04.01	EI-04	Двовимірна координатна площина	[FR] Візуалізація двовимірної координатної площини, на горизонтальній осі якої задано трудоємність (EIP), а на вертикальній – рівень невизначеності (EIU).
16	EI-04.02	EI-04	Позиціонування елементів оцінювання на координатній площині	[FR] Визначення позицій елементів оцінювання на координатній площині перетягуванням мишкою або введенням значень координат з клавіатури.
17	EI-04.03	EI-04	Розрахунок HOP	[FR] Розрахунок HOP відповідно до позицій елементів оцінювання на координатній площині (тобто за значеннями EIP та EIU). [FR] Задання параметрів, необхідних для перетворення EIP та EIU у HOP.
18	EI-05	–	Склад проєктної команди	[FR] CRUD-операції щодо ролей часників команди. [FR] Варіативна частина складу проєктної команди та правила зміни цієї варіативної частини.

№	Ідентифікатор	Батьківський елемент оцінювання	Заголовок елементу оцінювання	Атрибути елементу оцінювання
				[OOS] Вибір шаблону складу проєктної команди входить до задачі EI-11.02.
19	EI-05.01	EI-05	Ролі учасників команди	[FR] Табличне представлення переліку ролей проєктної команди. [FR] Додавання, редагування та видалення ролей часників проєктної команди та їх параметрів. [FR] Підтримка трьох типів команд залежно від диференціації спеціалізацій інженерів-розробників: без диференціації спеціалізацій, з диференційованими спеціалізаціями, із змішаними спеціалізаціями.
20	EI-05.02	EI-05	Варіативна частина складу проєктної команди	[DESC] Варіативна частина складу проєктної команди буде трансформуватися у обмеження задача цілочисельного програмування, що розв’язуються під час застосування методу підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди.
21	EI-06	–	Графік реалізації проєкту	[FR] Визначення ієрархічної структури фаз проєкту, кількість рівнів ієрархії якої залежить від типу оцінювання та специфіки проєкту. [FR] Варіативна частина графіку реалізації проєкту. [OOS] Вибір шаблону складу проєктної команди входить до задачі EI-11.01.
22	EI-06.01	EI-06	Ієрархічна структура фаз графіку реалізації проєкту	[FR] Визначення ієрархічної структури фаз проєкту, кількість рівнів ієрархії якої залежить від типу оцінювання та специфіки проєкту. [FR] Визначення параметрів фаз реалізації проєкту, наприклад: кількість спринтів, КЕПА тощо.
23	EI-06.02	EI-06	Варіативна частина графіку реалізації проєкту	[DESC] Варіативна частина графіку реалізації проєкту полягає у змінній тривалості фаз проєкту. [DESC] Варіативна частина графіку реалізації проєкту використовується методом підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди.
24	EI-07	–	Розклад реалізації елементів оцінювання	

№	*Ідентифікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
25	EI-07.01	EI-07	Метод побудови розкладу реалізації елементів оцінювання	[FR] Автоматична побудова розкладу реалізації елементів оцінювання із застосуванням методу, описаного у підрозділі 2.10. [TECH] Використати Python в якості технології реалізації.
26	EI-07.02	EI-07	Графічний інтерфейс розкладу реалізації елементів оцінювання	[FR] Візуалізація розкладу реалізації елементів оцінювання у вигляді діаграми Ганта.
27	EI-07.03	EI-07	Корегування вже побудованого розкладу реалізації елементів оцінювання	[FR] Можливість змінювати вже побудований розклад реалізації елементів оцінювання.
28	EI-08	–	Визначення КРЧР для попереднього оцінювання	[FR] Значення КРЧР визначається за результатами експертної оцінки. [OOS] Інструмент підтримки прийняття рішень, що пропонує значення КРЧР, враховуючи специфіку проєкту.
29	EI-09	–	Підтримка прийняття рішень щодо графіку реалізації проєкту та складу команди	
30	EI-09.01	EI-09	Метод підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди	[FR] Реалізації методу підтримки прийняття рішень, описаного у підрозділі 2.11. [TECH] Використати Python в якості технології реалізації.
31	EI-09.02	EI-09	Графічний інтерфейс підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди	[FR] Відображення результатів роботи методу підтримки прийняття рішень на графічному інтерфейсі.
32	EI-09.03	EI-09	Прийняття рішення користувачем щодо розкладу	[FR] Прийняття рішення користувачем щодо вибору запропонованого варіанту графіку реалізації проєкту та складу команди.

№	*Ідентифікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
			реалізації проєкту та складу команди	[FR] Коригування користувачем обраного варіанту графіку реалізації проєкту та складу команди.
33	EI-10	–	Зворотний зв'язок щодо оцінки	
34	EI-10.01	EI-10	Зворотний зв’язок від експертів	[FR] Зворотний зв’язок від експертів, що здійснюють перевірку оцінки. [USER] Експерти, що здійснюють перевірку оцінки.
35	EI-10.02	EI-10	Затвердження оцінки керівництвом	[FR] Надання зворотного зв’язку керівництвом. [FR] Затвердження оцінки керівництвом. [USER] Керівник, відповідальний за затвердження оцінки.
36	EI-10.03	EI-10	Зворотний зв’язок від зацікавлених сторін	[FR] Збір зворотного зв’язку від зацікавлених сторін (наприклад, замовників). [USER] Команда оцінювання.
37	EI-11	–	Бібліотека шаблонів	[DESC] Зібрання компонентів, що призначені для повторного використання.
38	EI-11.01	EI-11	Шаблони графіку реалізації проєкту	[FR] CRUD-операції щодо шаблонів графіку реалізації проєкту. [FR] Зберігання графіку реалізації проєкту у бібліотеку шаблонів під час оцінювання. [FR] Вибір шаблону графіку реалізації проєкту із вже підготованої оцінки.
39	EI-11.02	EI-11	Шаблони складу проєктної команди	[FR] CRUD-операції щодо шаблонів складу проєктної команди. [FR] Зберігання складу проєктної команди у бібліотеку шаблонів під час оцінювання. [FR] Вибір шаблону складу проєктної команди із вже підготованої оцінки.
40	EI-11.03	EI-11	Шаблони ICEO	[FR] CRUD-операції щодо шаблонів фрагментів ICEO. [FR] Зберігання фрагментів ICEO у бібліотеку шаблонів під час оцінювання. [FR] Вибір фрагменту ICEO із вже підготованої оцінки.

№	Ідентифікатор	Батьківський елемент оцінювання	Заголовок елементу оцінювання	Атрибути елементу оцінювання
41	EI-12	–	Безпека та профіль користувача	
42	EI-12.01	EI-12	Інтеграція з сервісом безпеки організації	[ARCH] Аутентифікація здійснюється через сервіс аутентифікації організації. Реєстрація користувача, зміна паролю, двофакторна аутентифікація реалізуються системою безпеки організації.
43	EI-12.02	EI-12	Аутентифікація та вихід із системи	[FR] Аутентифікація із застосуванням імені корпоративного користувача та пароля. [FR] Вихід із системи (операція, зворотна до аутентифікації). [ARCH] Аутентифікація здійснюється через сервіс аутентифікації організації із застосуванням протоколу Open ID Connect.
44	EI-12.03	EI-12	Авторизація	[ARCH] Правила застосовуються як на клієнтській (графічному інтерфейсі), так і на серверній (сервісах) сторонах. [ARCH] Авторизація базується на ролях користувача в інформаційній технології. [ARCH] Ролі користувача фіксовані і не можуть бути змінені без випуску нової версії системи.
45	EI-12.04	EI-12	Профіль користувача	[FR] Відображення профілю користувача. [FR] Редагування профілю користувача самим користувачем.
46	EI-12.05	EI-12	Роль адміністратора системи	[FR] Користувач має можливість входу в систему з правами адміністратора.
47	EI-12.06	EI-12	Управління користувачами	[USER] Функція, доступна адміністратору системи. [FR] Відображення переліку користувачів системи у табличному вигляді. [FR] Перегляд деталей користувача. [FR] Зміна ролей користувача. [FR] Активування та деактивування користувача.
48	EI-13	–	Інтеграція з модулем RTBPM-E	[OOS] Реалізація модуля RTBPM-E не входить до цього змісту проєктних робіт.

№	*Ідентифікатор*	*Батьківський елемент оцінювання*	*Заголовок елементу оцінювання*	*Атрибути елементу оцінювання*
49	EI-13.01	EI-13	Збір event-даних для надсилання у RTBPM-E	[ARCH] Event-дані можуть генеруватися як клієнтськими (графічним інтерфейсом), так і серверними (сервісами) компонентами. [ARCH] Event-дані надсилаються в режимі, наближеному до реального часу, у модуль RTBPM-E. [ARCH] Event-дані не зберігаються в базі даних інформаційної технології.
50	EI-13.02	EI-13	Зворотна інтеграція з RTBPM-E	[FR] Отримання з модуля RTBPM-E прогнозу часу завершення оцінювання. [DESC] У початкових версія інформаційної технології зворотна інтеграція з RTBPM-E функціонуватиме у «експериментальному» режимі.
51	EI-14	–	База даних інформаційної технології	[DESC] База даних необхідна для накопичення даних щодо проєктів та їх оцінок, що дасть змогу в перспективі, накопичивши достатній обсяг даних, засовувати методи машинного навчання та ШІ для підвищення ефективності методів оцінювання. [DESC] Інженери-розробники спеціалізації «back end» беруть участь у проєктуванні основної частину структури бази даних. [DESC] Інженери-розробники спеціалізації «data science» беруть участь у проєктуванні частини структури бази даних, яка стосується реалізації методу побудови розкладу реалізації елементів оцінювання та методу підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди.
52	EI-15	–	Налаштування базової структури проєкту	[DESC] Налаштування базової структури проєкту стосується всіх спеціалізацій інженерів-розробників.
53	EI-16	–	Невідома частина змісту проєктних робіт	[DESC] Частина змісту проєктних робіт, невідома на етапі проміжного оцінювання.

Таблиця Д-10.2

Нормалізована оцінки розробки для проміжного оцінювання[1,2]

№	Елемент оцінювання	НОР для «back end»-спеціалізації, людино-години			НОР для «front end»-спеціалізації, людино-години			НОР для «data science»-спеціалізації, людино-години		
		Оптиміс-тична	*Песиміс-тична*	*Найбільш ймовірна*	*Оптиміс-тична*	*Песиміс-тична*	*Найбільш ймовірна*	*Оптиміс-тична*	*Песиміс-тична*	*Найбільш ймовірна*
1	EI-01.01	20	32	26	30	40	32	0	0	0
2	EI-01.02	18	29	23	27	35	29	0	0	0
3	EI-02.01	20	32	26	28	37	32	0	0	0
4	EI-02.02	15	23	19	20	29	23	0	0	0
5	EI-02.03	16	20	17	25	38	30	0	0	0
6	EI-03.01	18	23	20	12	18	15	0	0	0
7	EI-03.02	20	26	22	23	30	27	0	0	0
8	EI-03.03	10	15	12	26	37	30	0	0	0
9	EI-03.04	10	15	12	26	37	30	0	0	0
10	EI-03.05	12	18	15	16	20	18	0	0	0
11	EI-04.01	0	0	0	54	75	61	0	0	0
12	EI-04.02	16	22	19	50	69	59	0	0	0
13	EI-04.03	18	23	20	22	30	25	0	0	0
14	EI-05.01	25	37	32	45	60	32	0	0	0
15	EI-05.02	30	50	45	50	70	62	0	0	0
16	EI-06.01	18	25	20	27	39	31	0	0	0

[1] НОР надано, виходячи з припущення, що реалізацію всіх компонентів буде здійснено з нуля (без використання вже готових реалізації з прототипу інформаційної технології, див. підрозділ 4.8).

[2] НОР надається лише для простих елементів оцінювання.

№	Елемент оцінювання	НОР для «back end»-спеціалізації, людино-години			НОР для «front end»-спеціалізації, людино-години			НОР для «data science»-спеціалізації, людино-години		
		Оптиміс-тична	Песиміс-тична	Найбільш ймовірна	Оптиміс-тична	Песиміс-тична	Найбільш ймовірна	Оптиміс-тична	Песиміс-тична	Найбільш ймовірна
17	EI-06.02	15	25	20	21	36	27	0	0	0
18	EI-07.01	8	13	10	0	0	0	55	70	61
19	EI-07.02	18	25	20	55	70	62	0	0	0
20	EI-07.03	10	18	13	28	39	35	14	20	18
21	EI-08	6	13	10	12	17	15	0	0	0
22	EI-09.01	8	13	10	0	0	0	55	76	61
23	EI-09.02	12	19	16	37	52	45	0	0	0
24	EI-09.03	15	23	18	34	58	42	0	0	0
25	EI-10.01	16	24	19	26	35	30	0	0	0
26	EI-10.02	16	24	19	30	42	35	0	0	0
27	EI-10.03	16	24	19	26	35	30	0	0	0
28	EI-11.01	23	35	29	34	49	40	0	0	0
29	EI-11.02	23	35	29	34	49	40	0	0	0
30	EI-11.03	23	35	29	34	49	40	0	0	0
31	EI-12.01	18	25	22	0	0	0	0	0	0
32	EI-12.02	25	38	30	6	9	7	0	0	0
33	EI-12.03	34	43	38	45	60	52	0	0	0
34	EI-12.04	12	17	14	23	30	27	0	0	0
35	EI-12.05	6	11	9	18	27	23	0	0	0
36	EI-12.06	12	25	18	25	39	31	0	0	0
37	EI-13.01	20	28	25	17	25	20	0	0	0
38	EI-13.02	12	19	15	8	14	11	0	0	0

№	Елемент оцінювання	НОР для «back end»-спеціалізації, людино-години			НОР для «front end»-спеціалізації, людино-години			НОР для «data science»-спеціалізації, людино-години		
		Оптиміс-тична	*Песиміс-тична*	*Найбільш ймовірна*	*Оптиміс-тична*	*Песиміс-тична*	*Найбільш ймовірна*	*Оптиміс-тична*	*Песиміс-тична*	*Найбільш ймовірна*
39	ЕІ-14	25	40	33	0	0	0	12	20	15
40	ЕІ-15	24	35	28	24	35	28	10	17	13
41	ЕІ-16	80	120	100	100	160	120	40	80	60
Всього:		***743***	***1117***	***921***	***1118***	***1594***	***1296***	***186***	***283***	***228***

Таблиця Д-10.3

Шаблон складу проєктної команди для проміжного оцінювання

№	*Категорія ролі*[1]	*Роль проєктної команди*	*Рівень компетентності*[2]	*Спеціалізація*[3]	*Нормалізована собівартість*	*Крок зміни ЕПЗ*	*КПРК*	*КПСЗ*	*Коефіцієнт продуктивності*
1	D	Старший інженер-розробник (тех. лідер)[4]	S	BE	1.00	0.50	1.2	0.9	1.08
2	D	Старший інженер-розробник	S	BE	0.90	0.50	1.2	1	1.20
3	D	Інженер-розробник	M	BE	0.70	1.00	1	1	1.00
4	D	Молодший інженер-розробник	J	BE	0.50	1.00	0.7	1	0.70
5	D	Старший інженер-розробник	S	FE	0.90	0.50	1.2	1	1.20
6	D	Інженер-розробник	M	FE	0.70	1.00	1	1	1.00
7	D	Молодший інженер-розробник	J	FE	0.50	1.00	0.7	1	0.70
8	D	«Data science»-інженер	M	DS	0.85	0.25	1	0.8	0.80
9	ND	Старший проєктний менеджер	S	–	0.95	0.50	–	–	–
10	ND	Технічний лідер	S	–	1.00	0.50	–	–	–
11	ND	Старший бізнес-аналітик	S	–	0.90	0.50	–	–	–
12	ND	Бізнес-аналітик	M	–	0.75	0.50	–	–	–

[1] Пояснення позначень категорій проєктних ролей: D – роль інженера-розробника, ND – «нерозробницька» роль (див. підпункт 2.6).

[2] Пояснення позначень рівнів компетентності: S – старший (англ. «senior»), M – середній рівень компетентності (англ. «middle»), J – молодший (англ. «junior»).

[3] Пояснення позначень спеціалізацій інженерів-розробників: BE – «back end», FE – «front end», DS – «data science».

[4] Роль технічного лідера виконує старший інженер-розробник 1 із таким розподілом робочого часу: 50% – обов'язки технічного лідера, 50% – обов'язки старшого інженера розробника.

№	*Категорія ролі*[1]	*Роль проєктної команди*	*Рівень компетентності*[2]	*Спеціалізація*[3]	*Нормалізована собівартість*	*Крок зміни ЕПЗ*	*КПРК*	*КПСЗ*	*Коефіцієнт продуктивності*
13	ND	Дизайнер	M	–	0.75	0.50	–	–	–
14	ND	DevOps-інженер	M	–	0.70	0.50	–	–	–
15	ND	Старший інженер-тестувальник	S	–	0.80	0.50	–	–	–
16	ND	Інженер-тестувальник	M	–	0.70	1.00	–	–	–
17	ND	Молодший інженер-тестувальник	J	–	0.60	1.00	–	–	–

Таблиця Д-10.4

Шаблон графіку реалізації проєкту для проміжного оцінювання

№	*Фази реалізації проєкту*	*Основні завдання фази реалізації проєкту*	*Кількість спринтів*	*КЗПА*	*КСАР*			*КРПРЧ*	*Коефіцієнт часу відсутності*
					BE	*FE*	*DS*		
1	Початок проєкту	Формування основного складу проєктної команди та початок реалізації проєкту	2	0.5	0.40	0.40	0.30	0.10	0.12
2	Масштабування команди	Розширення проєктної команди	2	0.4	0.30	0.35	0.25	0.12	0.12
3	Основна фаза реалізації	Основна фаза реалізації, на якій проєктна команда виконує переважну більшість проєктних задач	від 2 до 15	0.2	0.20	0.25	0.20	0.15	0.12
4	Стабілізація	Виправлення дефектів та підготовка до випуску релізу-кандидату, який буде тестуватися майбутніми користувачами	1	0.3	0.75	0.8	0.6	0.18	0.12
5	Тестування користувачами (UAT)	Тестування релізу-кандидату майбутніми користувачами, збір зворотного зв’язку, виправлення виявлених дефектів	2	0.3	–	–	–	0.18	0.12
6	Запуск в експлуатацію	Запуск системи у виробниче використання	1	0.3	–	–	–	0.18	0.12

Таблиця Д-10.5

Проміжна оцінка складу проєктної команди[1,2]

Ідентифікатор ролі	***ЕПЗ ролей проєктної команди для оптимістичної оцінки***						***ЕПЗ ролей проєктної команди для песимістичної оцінки***					
	Початок проєкту	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*
	1	*2*	*3*	*4*	*5*	*6*	*1*	*2*	*3*	*4*	*5*	*6*
1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
2	1	1	1	1	0	0	1	1	1	1	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	1	1	0	0	0	1	1	1	0	0
5	1	1	**2**	1	1	1	1	1	**1**	1	1	1
6	1	2	3	2	1	1	1	2	3	2	1	1
7	0	1	2	1	0	0	0	1	2	1	0	0
8	**0.5**	**0.5**	**1.5**	0.5	0.5	0	**1**	**1**	**1**	0.5	0.5	0
9	1	1	1	1	1	1	1	1	1	1	1	1
10	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
11	1	1	1	1	1	1	1	1	1	1	1	1
12	0	1	1	1	0	0	0	1	1	1	0	0

[1] Інформація про ролі проєктної команди представлена у табл. Д-10.3.
[2] Відмінності у складі проєктної команди для оптимістичної та песимістичної оцінок виділені півжирним.

Ідентифікатор ролі	***ЕПЗ ролей проєктної команди для оптимістичної оцінки***						***ЕПЗ ролей проєктної команди для песимістичної оцінки***					
	Початок проєкту	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*
	1	*2*	*3*	*4*	*5*	*6*	*1*	*2*	*3*	*4*	*5*	*6*
13	1	1	0.5	0.5	0.5	0	1	1	0.5	0.5	0.5	0
14	1	1	0.5	0.5	0.5	1	1	1	0.5	0.5	0.5	1
15	0.5	1	1	1	1	1	0.5	1	1	1	1	1
16	0	1	1	1	1	1	0	1	1	1	1	1
17	0	0	1	1	1	0	0	0	1	1	1	0
Сума	**10.00**	**15.50**	**19.50**	15.50	10.50	9.00	**10.50**	**16.00**	**18.00**	15.50	10.50	9.00

Таблиця Д-10.6

Проміжна оптимістична оцінка

№	*Характеристика*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Сума*
		1	*2*	*3*	*4*	*5*	*6*	
1	Тривалість, спринти	2	2	4	1	2	1	12
2	Тривалість, години	152	152	304	76	152	76	912
3	Проєктний робочий час команди, людино-години	1520.00	2356.00	5928.00	1178.00	1596.00	684.00	13262.00
4	Проєктний робочий час «нерозробницьких» ролей, людино-години	760.00	1140.00	2280.00	570.00	988.00	418.00	6156.00
5	Проєктний робочий час інженерів-розробників, людино-години	760.00	1216.00	3648.00	608.00	608.00	266.00	7106.00
6	Резерв проєктного робочого часу, людино-години	76.00	145.92	547.20	109.44	109.44	47.88	1035.88
7	Час відсутності, людино-години	91.20	145.92	437.76	72.96	72.96	31.92	852.72
8	Робочий час загальних проєктних активностей, людино-години	334.40	428.03	642.05	160.51	160.51	70.22	1795.73
9	Загальний робочий час розробки, людино-години	258.40	496.13	2020.99	265.09	265.09	115.98	3421.67
10	Робочий час розробки, людино-години	185.99	375.52	1644.86	150.26	0.00	0.00	2356.64
11	НСР «back end»-розробників, людино-години	101.15	164.10	482.79	65.14	0.00	0.00	813.18
12	НСР «front end»-розробників, людино-години	81.21	179.16	916.18	71.79	0.00	0.00	1248.35
13	НСР «data science»-розробників, людино-години	15.90	19.85	168.42	8.28	0.00	0.00	212.45
14	Нормалізована собівартість	1265.40	1805.00	4468.80	893.00	1276.80	558.60	10267.60

Таблиця Д-10.7

Проміжна песимістична оцінка

№	*Характеристика*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Сума*
		1	*2*	*3*	*4*	*5*	*6*	
1	Тривалість, спринти	2	2	8	1	2	1	16
2	Тривалість, години	152	152	608	76	152	76	1216
3	Проєктний робочий час команди, людино-години	1596.00	2432.00	10944.00	1178.00	1596.00	684.00	18430.00
4	Проєктний робочий час «нерозробницьких» ролей, людино-години	760.00	1140.00	4560.00	570.00	988.00	418.00	8436.00
5	Проєктний робочий час інженерів-розробників, людино-години	836.00	1292.00	6384.00	608.00	608.00	266.00	9994.00
6	Резерв проєктного робочого часу, людино-години	83.60	155.04	957.60	109.44	109.44	47.88	1463.00
7	Час відсутності, людино-години	100.32	155.04	766.08	72.96	72.96	31.92	1199.28
8	Робочий час загальних проєктних активностей, людино-години	367.84	454.78	1123.58	160.51	160.51	70.22	2337.46
9	Загальний робочий час розробки, людино-години	284.24	527.14	3536.74	265.09	265.09	115.98	4994.26
10	Робочий час розробки, людино-години	205.87	400.33	2879.91	150.26	0.00	0.00	3636.37
11	НСР «back end»-розробників, людино-години	101.15	164.10	965.59	65.14	0.00	0.00	1295.97
12	НСР «front end»-розробників, людино-години	81.21	179.16	1509.01	71.79	0.00	0.00	1841.17
13	НСР «data science»-розробників, людино-години	31.80	39.69	224.55	8.28	0.00	0.00	304.33
14	Нормалізована собівартість	1330.00	1869.60	8132.00	893.00	1276.80	558.60	14060.00

Таблиця Д-10.8

Альтернативні варіанти проміжних оцінок, отримані із застосуванням методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту[1,2,3]

№	***Тривалість***		***Оптимістичні проміжні оцінки***				***Песимістичні проміжні оцінки***			
	Спринти	*Місяці*	*Норм. собівар-тість*	*Середнє зважене ЕПЗ команди*	*Норм. час простою*	*Ранг*	*Норм. собівар-тість*	*Середнє зважене ЕПЗ команди*	*Норм. час простою*	*Ранг*
1	10	4.6	9459.15	16.03	9.43	0.100	12345.25	20.48	1.53	0.085
2	11	5.0	9459.15	14.07	19.72	0.101	12352.85	18.57	3.52	0.083
3	12	5.5	**9440.15**	**13.19**	**48.65**	**0.102**	12225.55	17.03	23.74	0.087
4	13	5.9	9930.35	12.60	81.35	0.098	12413.65	15.95	18.20	0.082
5	14	6.4	10465.20	12.22	163.98	0.097	12059.30	13.90	4.66	0.094
6	15	6.8	11121.65	12.15	282.03	0.082	**12299.65**	**13.82**	**39.60**	**0.088**
7	16	7.3	11671.70	12.00	379.65	0.077	12711.95	13.15	54.63	0.089
8	17	7.7	12495.35	12.11	505.10	0.065	13297.15	12.99	116.02	0.075
9	18	8.2	13319.00	12.20	641.53	0.056	13973.55	12.74	224.65	0.065
10	19	8.6	13948.85	12.12	777.70	0.054	14585.35	12.62	338.86	0.057

[1] Обрані альтернативи для оптимістичної та песимістичної оцінок виділені півжирним.

[2] Для реалізації методу підтримки прийняття рішень було використано Python 3.10.7, FICO Xpress 9.4.0, Pyomo 6.5.0, Ahpy 2.0.

[3] Нормалізована собівартість отримана згідно з (2.52); середнє зважене ЕПЗ команди отриманий згідно з (2.54); нормалізований час простою отриманий згідно з (2.55); ранжування альтернатив проведено методом аналізу ієрархій, використовуючи критерії порівняння із табл. Д-9.6.

№	Тривалість		Оптимістичні проміжні оцінки				Песимістичні проміжні оцінки			
	Спринти	*Місяці*	*Норм. собівар-тість*	*Середнє зважене ЕПЗ команди*	*Норм. час простою*	*Ранг*	*Норм. собівар-тість*	*Середнє зважене ЕПЗ команди*	*Норм. час простою*	*Ранг*
11	20	9.1	14756.35	12.19	921.03	0.047	15410.90	12.68	486.13	0.051
12	21	9.5	15563.85	12.25	1066.18	0.044	15789.95	12.42	543.38	0.053
13	22	10.0	16371.35	12.31	1212.49	0.040	16613.60	12.48	670.98	0.047
14	23	10.5	17178.85	12.36	1359.59	0.037	17437.25	12.54	806.60	0.044

Таблиця Д-10.9

Проміжна оцінка складу проєктної команди, отримана із застосуванням методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту [1,2]

Ідентифікатор ролі	***ЕПЗ ролей проєктної команди для оптимістичної оцінки***						***ЕПЗ ролей проєктної команди для песимістичної оцінки***					
	Початок проєкту	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*
	1	*2*	*3*	*4*	*5*	*6*	*1*	*2*	*3*	*4*	*5*	*6*
1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
2	0.5	1	1	1	0.5	0.5	0.5	1	1	1	0.5	0.5
3	1	1	1	1	0	0	1	1	1	1	0	0
4	0	1	1	1	0	0	0	1	1	1	0	0
5	1	2	**2**	2	1	1	1	1.5	**2**	2	1	1
6	1	2	2	2	0	0	1	2	2	2	0	0
7	1	1	1	1	0	0	1	1	1	1	0	0
8	**0.75**	1	1	1	0.25	0.25	**1**	1	1	1	0.25	0.25
9	1	1	1	1	1	1	1	1	1	1	1	1
10	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
11	1	1	1	1	1	1	1	1	1	1	1	1

[1] Інформація про ролі проєктної команди представлена у табл. Д-10.3.
[2] Відмінності у складі проєктної команди для оптимістичної та песимістичної оцінок виділені півжирним.

Ідентифікатор ролі	***ЕПЗ ролей проєктної команди для оптимістичної оцінки***						***ЕПЗ ролей проєктної команди для песимістичної оцінки***					
	Початок проєкту	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*
	1	*2*	*3*	*4*	*5*	*6*	*1*	*2*	*3*	*4*	*5*	*6*
12	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	1	0.5	0.5	0.5	1	1	1	0.5	0.5	0.5
14	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
15	1	1	1	1	0.5	0.5	1	1	1	1	0.5	0.5
16	0	1	1	1	0	0	0	1	1	1	0	0
17	0	1	1	1	1	1	0	1	1	1	1	1
Сума	**10.75**	**16.5**	16.5	16	7.25	7.25	**11**	**16**	16.5	16	7.25	7.25

Таблиця Д-10.10

Проміжна оптимістична оцінка, отримана із застосуванням методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту

№	*Характеристика*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Сума*
		1	*2*	*3*	*4*	*5*	*6*	
1	Тривалість, спринти	2	2	4	1	2	1	12
2	Тривалість, години	152	152	304	76	152	76	912
3	Проєктний робочий час команди, людино-години	1634.00	2508.00	5016.00	1216.00	1102.00	551.00	12027.00
4	Проєктний робочий час «нерозробницьких» ролей, людино-години	760.00	1064.00	2128.00	494.00	760.00	380.00	5586.00
5	Проєктний робочий час інженерів-розробників, людино-години	874.00	1444.00	2888.00	722.00	342.00	171.00	6441.00
6	Резерв проєктного робочого часу, людино-години	87.40	173.28	433.20	129.96	61.56	30.78	916.18
7	Час відсутності, людино-години	104.88	173.28	346.56	86.64	41.04	20.52	772.92
8	Робочий час загальних проєктних активностей, людино-години	384.56	508.29	508.29	190.61	90.29	45.14	1727.18
9	Загальний робочий час розробки, людино-години	297.16	589.15	1599.95	314.79	149.11	74.56	3024.72
10	Робочий час розробки, людино-години	214.39	446.27	1305.22	179.03	0.00	0.00	2144.91
11	НСР «back end»-розробників, людино-години	79.00	164.10	482.79	65.14	0.00	0.00	791.03
12	НСР «front end»-розробників, людино-години	107.05	234.28	687.14	93.89	0.00	0.00	1122.36
13	НСР «data science»-розробників, людино-години	23.85	39.69	112.28	16.57	0.00	0.00	192.39
14	Нормалізована собівартість	1312.90	1930.40	3860.80	936.70	932.90	466.45	9440.15

Таблиця Д-10.11

Проміжна песимістична оцінка, отримана із застосуванням методу підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту

№	*Характеристика*	*Початок проєкту*	*Масштабування команди*	*Основна фаза реалізації*	*Стабілізація*	*Тестування користувачами (UAT)*	*Запуск в експлуатацію*	*Сума*
		1	*2*	*3*	*4*	*5*	*6*	
1	Тривалість, спринти	2	2	7	1	2	1	15
2	Тривалість, години	152	152	532	76	152	76	1140
3	Проєктний робочий час команди, людино-години	1672.00	2432.00	8778.00	1216.00	1102.00	551.00	15751.00
4	Проєктний робочий час «нерозробницьких» ролей, людино-години	760.00	1064.00	3724.00	494.00	760.00	380.00	7182.00
5	Проєктний робочий час інженерів-розробників, людино-години	912.00	1368.00	5054.00	722.00	342.00	171.00	8569.00
6	Резерв проєктного робочого часу, людино-години	91.20	164.16	758.10	129.96	61.56	30.78	1235.76
7	Час відсутності, людино-години	109.44	164.16	606.48	86.64	41.04	20.52	1028.28
8	Робочий час загальних проєктних активностей, людино-години	401.28	481.54	889.50	190.61	90.29	45.14	2098.36
9	Загальний робочий час розробки, людино-години	310.08	558.14	2799.92	314.79	149.11	74.56	4206.60
10	Робочий час розробки, людино-години	224.33	423.30	2284.14	179.03	0.00	0.00	3110.79
11	НСР «back end»-розробників, людино-години	79.00	164.10	844.89	65.14	0.00	0.00	1153.12
12	НСР «front end»-розробників, людино-години	107.05	206.72	1202.49	93.89	0.00	0.00	1610.15
13	НСР «data science»-розробників, людино-години	31.80	39.69	196.49	16.57	0.00	0.00	284.55
14	Нормалізована собівартість	1345.20	1862.00	6756.40	936.70	932.90	466.45	12299.65

Додаток 11. Детальне оцінювання програмного забезпечення інформаційної технології

Таблиця Д-11.1

Ієрархічна структура елементів оцінювання та нормалізована оцінка розробки для детального оцінювання[1]

№	*Елемент оцінювання*	*Заголовок елементу оцінювання*	*Залежності елементу оцінювання*[2]	*Оптимістична НОР, людино-години*			*Песимістична НОР, людино-години*		
				BE	*FE*	*BS*	*BE*	*FE*	*BS*
1	EI-01	Проєкти та сценарії їх реалізації		38	57	0	61	75	0
2	EI-02	Оцінки проєктів	EI-01	51	73	0	75	104	0
3	EI-03	Побудова ICEO	EI-01	70	103	0	97	142	0
4	EI-04	Оцінювання трудоємності групування за спорідненістю		34	126	0	45	174	0
5	EI-05	Склад проєктної команди	EI-01	55	95	0	87	130	0
6	EI-06	Графік реалізації проєкту	EI-01	33	48	0	50	75	0
7	EI-07	Розклад реалізації елементів оцінювання	EI-05,EI-06	36	83	69	56	109	90
8	EI-08	Визначення КРЧР для попереднього оцінювання	EI-02	6	12	0	13	17	0
9	EI-09	Підтримка прийняття рішень щодо графіку реалізації проєкту та складу команди	EI-05,EI-06	35	71	55	55	110	76
10	EI-10	Зворотний зв'язок щодо оцінки	EI-02	48	82	0	72	112	0

[1] Для побудови розкладу реалізації елементів оцінювання з табл. Д-10.1 взято лише елементи оцінювання першого рівня. НОР для цих елементів оцінювання отримано з табл. Д-10.2.

[2] Всі залежності належать до типу «завершення – початок».

№	*Елемент оцінювання*	*Заголовок елементу оцінювання*	*Залежності елементу оцінювання*[1]	*Оптимістична НОР, людино-години*			*Песимістична НОР, людино-години*		
				BE	*FE*	*BS*	*BE*	*FE*	*BS*
11	EI-11	Бібліотека шаблонів		69	102	0	105	147	0
12	EI-12	Безпека та профіль користувача		107	117	0	159	165	0
13	EI-13	Інтеграція з модулем RTBPM-E		32	25	0	47	39	0
14	EI-14	База даних інформаційної технології		25	0	12	40	0	20
15	EI-15	Налаштування базової структури проєкту		24	24	10	35	35	17
16	EI-16	Невідома частина змісту проєктних робіт		80	100	40	120	160	80
			Сума	**743**	**1118**	**186**	**1117**	**1594**	**283**

Таблиця Д-11.2

Оптимістичний розклад реалізації елементів оцінювання

Фаза реалізації		Початок проєкту		Розширення команди		Основна фаза реалізації				Стабілі-зація	Тестування користувачами (UAT)		Запуск в експлуа-тацію	Сума
Спринт		1	2	3	4	5	6	7	8	9	10	11	12	
НСР, людино-години	BE	40	40	83	83	121	121	121	121	66	0	0	0	**796**
	FE	54	54	118	118	172	172	172	172	94	0	0	0	**1126**
	DS	12	12	20	20	29	29	29	29	17	0	0	0	**197**

# ЕО	Залежності	НОР, л-г	Розклад реалізації елементів оцінювання											
EI-01		38	16	22										
		57	30	27										
		0												
EI-02	EI-01	51			51									
		73		27	46									
		0												
EI-03	EI-01	70					70							
		103					103							
		0												
EI-04		34				20	14							
		126				47	44	35						
		0												
EI-05	EI-01	55			25	30								
		95			72	23								
		0												
EI-06	EI-01	33				33								
		48				48								
		0												
EI-07	EI-05,EI-06	36						36						
		83						50	33					
		69			20	20	19	10						
EI-08	EI-02	6						6						
		12						12						
		0												
EI-09	EI-05,EI-06	35							20	15				
		71							32	39				
		55						9	19	19	8			

Фаза реалізації		Початок проєкту		Розширення команди		Основна фаза реалізації				Стабілі-зація	Тестування користувачами (UAT)		Запуск в експлуа-тацію	Сума
Спринт		1	2	3	4	5	6	7	8	9	10	11	12	
НСР, людино-години	BE	40	40	83	83	121	121	121	121	66	0	0	0	**796**
	FE	54	54	118	118	172	172	172	172	94	0	0	0	**1126**
	DS	12	12	20	20	29	29	29	29	17	0	0	0	**197**
# ЕО	**Залежності**	**НОР, л-г**	**Розклад реалізації елементів оцінювання**											
EI-10	EI-02	48						48						
		82						50	32					
		0												
EI-11		69							50	19				
		102							50	52				
		0												
EI-12		107							31	35	41			
		117								31	86			
		0												
EI-13		32								32				
		25								25				
		0												
EI-14		25		18	7									
		0												
		12	2	10										
EI-15		24	24											
		24	24											
		10	10											
EI-16		80				20	20	20	20					
		100				25	25	25	25					
		40				10	10	10	10					
НСР - НОР, людино-години	BE	0	0	0	0	17	11	0	0	25	0	0	0	**53.0**
	FE	0	0	0	0	0	0	0	0	8	0	0	0	**8.0**
	DS	0	2	0	0	0	0	0	0	9	0	0	0	**11.0**
Коеф. штрафу, θ		0.92	0.83	0.75	0.67	0.58	0.50	0.42	0.33	0.25	0.17	0.08	0.00	
θ · (НСР - НОР)		0.00	1.67	0.00	0.00	9.92	5.50	0.00	0.00	10.50	0.00	0.00	0.00	**27.6**

Таблиця Д-11.3

Песимістичний розклад реалізації елементів оцінювання

Фаза реалізації		Початок проєкту		Розширення команди		Основна фаза реалізації							Стабілі-зація	Тестування користувачами (UAT)		Запуск в експлуа-тацію	Сума
Спринт		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
НСР, людино-години	BE	40	40	83	83	121	121	121	121	121	121	121	66	0	0	0	**1159**
	FE	54	54	104	104	172	172	172	172	172	172	172	94	0	0	0	**1614**
	DS	16	16	20	20	29	29	29	29	29	29	29	17	0	0	0	**292**

# ЕО	Залежності	НОР, л-г	Розклад реалізації елементів оцінювання														
EI-01		61	15	10	36												
		75	34	39	2												
		0															
EI-02	EI-01	75				65	10										
		104				32	72										
		0															
EI-03	EI-01	97					14	59	24								
		142					20	22	100								
		0															
EI-04		45			27	18											
		174			102	72											
		0															
EI-05	EI-01	87					87										
		130					70	60									
		0															
EI-06	EI-01	50						50									
		75						75									
		0															
EI-07	EI-05,EI-06	56							56								
		109							57	52							
		90			15	15	15	15	15	15							
EI-08	EI-02	13								13							
		17									17						
		0															
EI-09	EI-05,EI-06	55							29	26							
		110								100	10						
		76				5	6	6	6	4	19	14	8	8			

Фаза реалізації		Початок проєкту		Розширення команди		Основна фаза реалізації							Стабілі-зація	Тестування користувачами (UAT)		Запуск в експлуа-тацію	Сума
Спринт		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
НСР, людино-години	BE	40	40	83	83	121	121	121	121	121	121	121	66	0	0	0	**1159**
	FE	54	54	104	104	172	172	172	172	172	172	172	94	0	0	0	**1614**
	DS	16	16	20	20	29	29	29	29	29	29	29	17	0	0	0	**292**
# ЕО	Залежності	НОР, л-г	Розклад реалізації елементів оцінювання														
EI-10	EI-02	72							67	5							
		112								112							
		0															
EI-11		105								50	55						
		147								13	134						
		0															
EI-12		159								51	46	62					
		165									13	117	35				
		0															
EI-13		47										23	24				
		39											39				
		0															
EI-14		40	5	15	20												
		0															
		20	10	5	5												
EI-15		35	20	15													
		35	20	15													
		17	6	11													
EI-16		120					10	12	12	15	15	20	36				
		160					10	15	15	20	20	25	55				
		80					8	8	8	10	10	15	21				
НСР - НОР, людино-години	BE	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	**42.0**
	FE	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	**20.0**
	DS	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	**9.0**
Коеф. штрафу, θ		0.93	0.87	0.80	0.73	0.67	0.60	0.53	0.47	0.40	0.33	0.27	0.20	0.13	0.07	0.00	
θ · (НСР - НОР)		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	14.20	0.00	0.00	0.00	**14.2**

Додаток 12. Вихідний код реалізації методу підтримки прийняття рішень щодо графіку реалізації проєкту та складу команди[1]

```
# %% [markdown]
# # Optimization Task

# %% [markdown]
# ## Normalized Development Capacity

# %% [markdown]
# Generic formula for NDC of specialization $s \in S$ for development role $m \in T_{D}$ on phase $h \in H$
(applicable for a development team with mixed specializations):
#
# $$
#
# C_{h}^{m\left( s \right)}=\frac{\phi _{h}^{m\left( s \right)}}{{\alpha }^{m\left( s \right)}}\eta
_{h}^{m}}{\phi _{h}^{m}\left( 1+d_{h}^{\left( s \right)} \right)}\left[ \left( 1-{{g}_{h}}-{{r}_{h}}
\right)\phi _{h}^{m}{{L}_{h}}-\left( 1-{{g}_{h}} \right)\underset{\mathbb{O}_{h}^{m}}{\mathop{\min
}}\,\,O_{h}^{m}-\underset{\mathbb{I}_{h}^{m}}{\mathop{\min }}\,\,I_{h}^{m} \right],\ h\in H,\ m\in {{T}_{D}},
\ s \in S
#
# $$

# %% [markdown]
# Let assume the following:
#
# 1. The development team has differentiated (not mixed) specializtions, then $\phi _{h}^{m} = \phi
_{h}^{m\left( s \right)}$.
# 2. Productivity coefficient of role $m$ does not depend on project phase $h \in H$, i.e., $\rho^{m}$ stays
the same for the whole project implementation.
# 3. Project leaves time $O^{m}_{h}$ proportinally depends on project working time $W^{m}_{h}$:
$\underset{\mathbb{O}_{h}^{m}}{\mathop{\min }}\,\,O_{h}^{m} = o_{h} \phi _{h}^{m}{{L}_{h}}$.
```

[1] Вихідний код в цьому додатку розроблено із використанням Visual Studio Code v1.99.3, Python v3.10.7, та Jupyter v2025.3.0 (плагін до Visual Studio Code).

```
# 4. Minimal value of the idle working time equals 0, i.e., $\underset{\mathbb{I}_{h}^{m}}{\mathop{\min
}}\,\,I_{h}^{m} = 0$.
# 5. Development FTE is expressed as follows: $\phi_{h}^{m} = \mu^{m} f_{h}^{m}$.
#

# %% [markdown]
# After applying changes 1-5, NDC of role $m$ on phase $h$ is the following:
#
# $$
# C_{h}^{m}=\frac{\rho^{m}}{1+d_{h}^{\left( s \right)}}\left[ \left( 1-{{g}_{h}}-{{r}_{h}} \right)\phi
_{h}^{m}{{L}_{h}}-\left( 1-{{g}_{h}} \right)o_{h} \phi _{h}^{m}{{L}_{h}} \right],\ h\in H,\ m\in {{T}_{D}}
# $$

# %% [markdown]
# After some simplifications, NDC of role $m$ on phase $h$ is expressed as follows:
#
# $$
# C_{h}^{m}=\frac{\rho^{m} \mu^{m} f_{h}^{m}{{L}_{h}}}{1+d_{h}^{\left( s \right)}}\left[ \left( 1-{{g}_{h}-
{{r}_{h}}} \right) - \left(1 - g_{h} \right)o_{h}  \right],\ h\in H,\ m\in {{T}_{D}}
# $$

# %% [markdown]
# # Constants

# %%
WORKING_DAYS_PER_MONTH = 21
WORKING_HOURS_PER_DAY = 8
WORKING_HOURS_PER_MONTH = WORKING_DAYS_PER_MONTH * WORKING_HOURS_PER_DAY

# %% [markdown]
# ## Team Composition

# %% [markdown]
# $$
# {{\left[ \phi _{h}^{{{T}^{*}}} \right]}_{\min }}\le \sum\limits_{m\in {{T}^{*}}}{\phi _{h}^{m}}\le {{\left[
\phi _{h}^{{{T}^{*}}} \right]}^{\max }}
```

```
#
# $$

# %% [markdown]
# # Imports

# %%
import platform
print("Python", platform.python_version())

import pandas as pd
print("Pandas", pd.__version__)

import numpy as np
print("NumPy", np.__version__)

import xpress
print("FICO Xpress", xpress.__version__)
# print("Using Xpress Optimizer version", xpress.getversion())

import pyomo
print("Pyomo", pyomo.__version__)
import pyomo.environ as pyo

import ahpy
print("Ahpy", ahpy.__version__)

import pprint as pp

import math

# %% [markdown]
# # Input Data

# %% [markdown]
# ## Project Team
```

```
# %%
team_columns = \
    ['ID', 'Type', 'Name', 'Competency', 'Specialization', 'Rate', 'FTE Step', 'Alpha', 'Eta', 'Rho']

team_data = [
    [1,  'D',  'Старший інженер-розробник (тех. лід.)','S', 'BE',  1.0,  0.5,  1.2,  0.9,  1.08],
    [2,  'D',  'Старший інженер-розробник',            'S', 'BE',  0.9,  0.5,  1.2,  1.0,  1.2 ],
    [3,  'D',  'Інженер-розробник',                    'M', 'BE',  0.7,  1.0,  1.0,  1.0,  1.0 ],
    [4,  'D',  'Молодший інженер-розробник',           'J', 'BE',  0.5,  1.0,  0.7,  1.0,  0.7 ],
    [5,  'D',  'Старший інженер-розробник',            'S', 'FE',  0.9,  0.5,  1.2,  1.0,  1.2 ],
    [6,  'D',  'Інженер-розробник',                    'M', 'FE',  0.7,  1.0,  1.0,  1.0,  1.0 ],
    [7,  'D',  'Молодший інженер-розробник',           'J', 'FE',  0.5,  1.0,  0.7,  1.0,  0.7 ],
    [8,  'D',  'Data science інженер',                  'M', 'DS',  0.85, 0.25, 1.0,  0.8,  0.8 ],
    [9,  'ND', 'Старший проєктний менеджер',           'S',  None, 0.95, 0.5,  None, None, None],
    [10, 'ND', 'Технічний лідер',                      'S',  None, 1.0,  0.5,  None, None, None],
    [11, 'ND', 'Старший бізнес-аналітик',              'S',  None, 0.9,  0.5,  None, None, None],
    [12, 'ND', 'Бізнес-аналітик',                      'M',  None, 0.75, 0.5,  None, None, None],
    [13, 'ND', 'Дизайнер',                             'M',  None, 0.75, 0.5,  None, None, None],
    [14, 'ND', 'DevOps-інженер',                       'M',  None, 0.7,  0.5,  None, None, None],
    [15, 'ND', 'Старший інженер-тестувальник',         'S',  None, 0.8,  0.5,  None, None, None],
    [16, 'ND', 'Інженер-тестувальник',                 'M',  None, 0.7,  1.0,  None, None, None],
    [17, 'ND', 'Молодший інженер-тестувальник',        'J',  None, 0.6,  1.0,  None, None, None]
]

team_df = pd.DataFrame(data=team_data, columns=team_columns).set_index('ID')

display(team_df)

# %% [markdown]
# ## Project Phases

# %%
phases = [
    [1, 'Початок проєкту',                 1, 2, 2],
    [2, 'Масштабування команди',           1, 2, 2],
```

```
    [3, 'Основна фаза реалізації',          1, 2, 15],
    [4, 'Стабілізація',                     1, 1, 1],
    [5, 'Тестування користувачами (UAT)',   0, 2, 2],
    [6, 'Запуск в експлуатацію',            0, 1, 1]
]

phases_df = pd.DataFrame(
    data=phases,
    columns=["ID", "Phase", "Delta", "Sprints Min", "Sprints Max"]
).set_index('ID')

phases_df['Sprint Duration'] = 76
phases_df['Phase Duration Min'] = phases_df['Sprints Min'] * phases_df['Sprint Duration']
phases_df['Phase Duration Max'] = phases_df['Sprints Max'] * phases_df['Sprint Duration']

# Coefficient of project working time contingency, CPWTC
phases_df['CPWTC (r)'] = [0.1, 0.12, 0.15, 0.18, 0.18, 0.18]
# Leave coefficient, LC
phases_df['LC (o)'] = [0.12, 0.12, 0.12, 0.12, 0.12, 0.12]
# General project activities coefficient, GPAC
phases_df['GPAC (g)'] = [0.5, 0.4, 0.2, 0.3, 0.3, 0.3]

# Supplementary development activities coefficient, SDAC
phases_df['SDAC BE'] = [0.4, 0.3, 0.2, 0.75, None, None]
phases_df['SDAC FE'] = [0.4, 0.35, 0.25, 0.8, None, None]
phases_df['SDAC DS'] = [0.3, 0.25, 0.2, 0.6, None, None]

display(phases_df)

# %% [markdown]
# ## Normalized Development Estimate

# %%
nde_opt_df = pd.DataFrame()
nde_opt_df["Specialization"] = ['BE', 'FE', 'DS']
nde_opt_df = nde_opt_df.set_index("Specialization")
```

```
nde_opt_df["NDE"] = [743.0, 1118.0, 186.0]

nde_psm_df = pd.DataFrame()
nde_psm_df["Specialization"] = ['BE', 'FE', 'DS']
nde_psm_df = nde_psm_df.set_index("Specialization")
nde_psm_df["NDE"] = [1117.0, 1594.0, 283.0]

display(nde_opt_df)

display(nde_psm_df)

# %% [markdown]
# ## Constraints

# %% [markdown]
# ### Team FTEs Constraints (Lower-Upper)

# %% [markdown]
# $$
# {{\left[ \phi _{h}^{{{T}^{*}}} \right]}_{\min }}\le \sum\limits_{m\in {{T}^{*}}}{\phi _{h}^{m}}\le {{\left[
\phi _{h}^{{{T}^{*}}} \right]}^{\max }}
#
# $$

# %%
team_fte_columns = ['Index', 'Phases', 'Team', 'Lower', 'Upper', 'Description']

whole_team_ids = team_df.index.tolist()

team_fte_contraints = [
    [[1,2,3,4,5,6], [1],  0.5, None, 'Залучення старшого BE-розробника (техліда) на всю тривалість проєкту'],
    [[1,2,3,4,5,6], [10], 0.5, None, 'Залучення техліда на всю тривалість проєкту'],
    [[1,2,3,4,5,6], [2],  0.5, None, 'Залучення старшого BE-розробника на фази 1-6'],
    [[2,3,4],       [4],  1.0, None, 'Залучення молодшого BE-розробника'],

    [[1,2,3,4,5,6], [5],  1.0, None, 'Залучення старшого FE-розробника на всю тривалість проєкту'],
```

```
    [[2,3,4],       [7],  1.0, None, 'Залучення молодшого FE-розробника'],

    [[1,2,3],       [8],  0.5, None, 'Залучення "Data Science"-інженера на початку проєкту'],
    [[4,5,6],       [8],  0.25,None, 'Залучення "Data Science"-інженера на завершальних фазах проєкту'],

    [[1,2,3,4,5,6], [9],  1.0, None, 'Залучення проєктного менеджера на всю тривалість проєкту'],
    [[1,2,3,4,5,6], [11], 1.0, None, 'Залучення старшого бізнес-аналітика на всю тривалість проєкту'],

    [[1,2,3],       [13], 1.0, None, 'Залучення UX-дизайнера на на початкові фази проєкту'],
    [[4,5,6],       [13], 0.5, None, 'Залучення UX-дизайнера на завершальні фази проєкту'],
    [[1,2,3,4,5,6], [14], 0.5, None, 'Залучення DevOps-інженера на початкові та завершальну фази проєкту'],

    [[1,2,3,4],     [15], 1.0, None, 'Залучення старшого інженера-тестувальника на початкових фазах проєкту'],
    [[5,6],         [15], 0.5, None, 'Залучення старшого інженера-тестувальника на початкових фазах проєкту'],
    [[2,3,4,5,6],   [17], 1.0, None, 'Залучення молодшого інженера-тестувальника на фази реалізації проєкту']
]

for i in range(len(team_fte_contraints)):
    team_fte_contraints[i].insert(0, i + 1)

team_fte_constraints_df = pd.DataFrame(data=team_fte_contraints, columns=team_fte_columns).set_index('Index')

display(team_fte_constraints_df)

# %% [markdown]
# ### Team FTEs Interrelations

# %% [markdown]
# $$
# {{\gamma }_{1}}\sum\limits_{{{m}_{1}}\in {{T}_{1}}}{\phi _{{{h}_{1}}}^{{{m}_{1}}}}\le {{\gamma
}_{2}}\sum\limits_{{{m}_{2}}\in {{T}_{2}}}{\phi _{{{h}_{2}}}^{{{m}_{2}}}}
# $$

# %%
team_rel_columns = \
    ['Index', 'Phases', 'Gamma1', 'Phase1', 'Team1', 'Gamma2', 'Phase2', 'Team2', 'Description']
```

```
def scaling_between_adjacent_phases(phase1, phase2, team_roles, scaling, description):
    constraints = []
    for t in team_roles:
        if scaling > 1:
            constraints.append([None, 1, phase2, [t], scaling, phase1, [t], description])
            constraints.append([None, 1, phase1, [t], 1, phase2, [t], description])
        elif scaling < 1:
            constraints.append([None, 1, phase2, [t], scaling, phase1, [t], description])
            constraints.append([None, 1, phase2, [t], 1, phase1, [t], description])
        elif scaling == 1:
            constraints.append([None, 1, phase1, [t], 1, phase2, [t], description])
            constraints.append([None, 1, phase2, [t], 1, phase1, [t], description])

    return constraints


team_rel_contraints = [
    [[1,2,3,4,5,6], 1, None, [1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17], 18, None, [9],
     'На 1 проєктного менеджера не повинно бути більше 18 учасників команди'],
    [[1,2,3,4,5], 1, None, [1,2,3,4,5,6,7,8],   10, None, [11,12],
     'На 1 бізнес-аналітика не повинно бути більше 10 інженерів-розробників'],

    [[3,4,5,6], 1, None, [1,2,3,4,5,6,7,8], 4, None, [15,16,17],
     'На 1 тестувальника не повинно бути більше 4 інженерів-розробників'],
    [[1,2,3], 1, None, [1,2,3,4,5,6,7,8], 12, None, [13],
     'На 1 UX-дизайнера не повинно бути більше 12 інженерів-розробників'],
    [[1,2,3,4,5,6], 0.5, None, [1,2,3,4,5,6,7,8], 10, None, [14],
     'На 0.5 DevOps-інженера не повинно бути більше 10 інженерів-розробників'],

    [[1,2,3,4,5,6], 0.5, None, [2,3,4,5,6,7,8], 10, None, [10],
     'На 0.5 техліда не повинно бути більше 10 інженерів-розробників'],

    [[1,2,3,4,5,6], 1, None, [10], 1, None, [1],
     'Техлід та старший BE-розробник 1 - це один і той самий учасник команди'],
    [[1,2,3,4,5,6], 1, None, [1], 1, None, [10],
```

```
     'Техлід та старший BE-розробник 1 - це один і той самий учасник команди'],

    [[1,2,3,4,5,6], 2, None, [4], 1, None, [1,2,3],
      'На 2 BE-розробники не повинно бути більше як 1 молодший BE-розробник'],
    [[1,2,3,4,5,6], 2, None, [7], 1, None, [5,6],
      'На 2 FE-розробники не повинно бути більше як 1 молодший FE-розробник'],
    [[1,2,3,4], 2, None, [17], 1, None, [15,16],
      'На 2 інженери-тестувальники не повинно бути більше як 1 молодший інженер-тестувальник'],
]

# залученість на суміжних фазах
#  ----- 1 => 2 -----
team_rel_contraints += scaling_between_adjacent_phases(
    phase1=1, phase2=2, team_roles=[2,3,5,6,7,8], scaling=2,
    description='Залученість інженерів-розробників не повинна зрости більше як у 2 рази')

#  ----- 2 => 3 -----
team_rel_contraints += scaling_between_adjacent_phases(
    phase1=2, phase2=3, team_roles=[2,8,5], scaling=1.5,
    description='Залученість інженерів-розробників не повинна зрости більше як у 1.5 рази')
team_rel_contraints += scaling_between_adjacent_phases(
    phase1=2, phase2=3, team_roles=[3,6,7], scaling=2.0,
    description='Залученість інженерів-розробників не повинна зрости більше як у 1.5 рази')

#  ----- 3 => 4 -----
team_rel_contraints += scaling_between_adjacent_phases(
    phase1=3, phase2=4, team_roles=[2,3,5,6,7,8], scaling= 1.0,
    description='Залученість на 3-ій та 4-ій фазах не повинна відрізнятися')

#  ----- 4 => 5 -----
team_rel_contraints += scaling_between_adjacent_phases(
    phase1=4, phase2=5, team_roles=[2,3,5,6,7,8], scaling=0.5,
    description='Залученість інженерів-розробників повинна зменшитися мінімум у 2 рази')

# ----- 5 => 6 -----
team_rel_contraints += scaling_between_adjacent_phases(
```

```
    phase1=5, phase2=6, team_roles=[2,3,5,6,7,8], scaling=1.0,
    description='Залученість інженерів-розробників на 5-ій та 6-ій фазах не повинна відрізнятися')

for i in range(len(team_rel_contraints)):
    team_rel_contraints[i].insert(0, i + 1)

team_rel_constraints_df = pd.DataFrame(
    data=team_rel_contraints, columns=team_rel_columns).set_index('Index')

display(team_rel_constraints_df)

# %% [markdown]
# # Optimization Task (Pyomo & Xpress)

# %%
def create_estimation_optimization_task():
    task = pyo.AbstractModel()

    # Project team (both D & ND)
    task.T = pyo.Set()

    # Development team
    task.D = pyo.Set()
    # Productivity coefficient
    task.rho = pyo.Param(task.D, within=pyo.NonNegativeReals)
    # Normalized hourly rate for each role
    task.rate = pyo.Param(task.T, within=pyo.NonNegativeReals)
    # FTE increment for each role
    task.mu = pyo.Param(task.T, within=pyo.PositiveReals)


    # Development specializations
    task.S = pyo.Set()
    # Development roles by specializations
    task.SD = pyo.Param(task.S, within=pyo.Any)
    # NDE (by development specialization)
```

```
task.NDE = pyo.Param(task.S, within=pyo.NonNegativeReals)

# Project phases
task.H = pyo.Set()
# Development project phases (delta == 1)
task.HD = pyo.Set()
# Duration of project phases
task.L = pyo.Param(task.H, within=pyo.NonNegativeReals)
# General project activities coefficient (GPAC)
task.g = pyo.Param(task.H, within=pyo.NonNegativeReals)
#
task.o = pyo.Param(task.H, within=pyo.NonNegativeReals)
#
task.r = pyo.Param(task.H, within=pyo.NonNegativeReals)

# Supplementary development activities coefficient (SDAC)
task.d = pyo.Param(task.S * task.HD, within=pyo.NonNegativeReals)

# min/max team FTE
task.TeamFTE_Index = pyo.Set()
task.TeamFTE = pyo.Param(task.TeamFTE_Index, within=pyo.Any)

# Team interrelations
task.TeamRel_Index = pyo.Set()
task.TeamRel = pyo.Param(task.TeamRel_Index, within=pyo.Any)

# Decision variables
task.F = task.T * task.H
task.f = pyo.Var(task.F, domain=pyo.NonNegativeIntegers)

# Objective function
task.OBJ = pyo.Objective(
    rule=lambda m: sum(sum((m.L[h] * m.rate[t] * m.mu[t] * m.f[t, h]) for t in m.T) for h in m.H),
    sense=pyo.minimize
)
```

```
    # NDC >= NDE
    task.NDC_NDE_Constraint = pyo.Constraint(
        task.S,
        rule=lambda m, s: (
            sum(
                sum(
                    ((m.rho[t] * m.mu[t] * m.f[t, h] * m.L[h]) / (1 + m.d[s, h])) \
                        * ((1 - m.g[h] - m.r[h]) - (1 - m.g[h]) * m.o[h]) for t in m.SD[s]
                ) for h in m.HD
            ) >= m.NDE[s]
        )
    )

    # Project roles FTE should not be less and/or greater than the specified value(s)
    def team_fte_constraint(m, i):
        h = m.TeamFTE[i]['Phase']
        if 'Lower' in m.TeamFTE[i]:
            return (m.TeamFTE[i]['Lower'] <= sum(m.mu[t] * m.f[t, h] for t in m.TeamFTE[i]['Team']))
        elif 'Upper' in m.TeamFTE[i]:
            return (sum(m.mu[t] * m.f[t, h] for t in m.TeamFTE[i]['Team']) <= m.TeamFTE[i]['Upper'])

    task.TeamFTE_Constraint = pyo.Constraint(task.TeamFTE_Index, rule=team_fte_constraint)

    # Project team role FTEs interrelations
    def team_rel_constraint(m, i):
        h1 = m.TeamRel[i]['Phase1']
        h2 = m.TeamRel[i]['Phase2']
        return (m.TeamRel[i]['Gamma1'] * sum(m.mu[t1] * m.f[t1, h1] for t1 in m.TeamRel[i]['Team1'])
             <= m.TeamRel[i]['Gamma2'] * sum(m.mu[t2] * m.f[t2, h2] for t2 in m.TeamRel[i]['Team2']))

    task.TeamRel_Constraint = pyo.Constraint(task.TeamRel_Index, rule=team_rel_constraint)

    return task

# %% [markdown]
# # Input Parameters
```

```
# %%
def project_schedule_alternatvies(phases_df):
    phase_durations = [list(range(dr[0], dr[1] + 1))
                       for dr in phases_df[["Sprints Min", "Sprints Max"]].values.tolist()]

    def _duration_combinations(_phase_durations):
        if (len(_phase_durations) == 1):
            return [[d] for d in _phase_durations[0]]

        ext_combinations = []
        combinations = _duration_combinations(_phase_durations[1:])
        for duration in _phase_durations[0]:
            ext_combinations += [[duration] + c for c in combinations]

        return ext_combinations

    phase_duration_combinations = _duration_combinations(phase_durations)
    alternatives_dfs = {}

    for i in range(len(phase_duration_combinations)):
        phase_durations_df = pd.DataFrame.from_dict(
            {p + 1 : phase_duration_combinations[i][p] for p in range(len(phase_duration_combinations[i])) },
            orient='index',
            columns=['Sprints'])
        phase_durations_df['Duration'] = phase_durations_df['Sprints'] * phases_df['Sprint Duration']

        alternatives_dfs[i + 1] = phase_durations_df

    return alternatives_dfs


# %%
def team_fte_constraint_params(constraints_df):
    constraints = {}
    key = 0
```

```
    for index, row in constraints_df.iterrows():
        phases = row['Phases']
        for h in phases:
            if row['Lower'] is not None and not np.isnan(row['Lower']):
                key += 1
                constraints[key] = {
                    'Phase' : h,
                    'Team' : row['Team'],
                    'Lower': row['Lower']
                }
            if row['Upper'] is not None and not np.isnan(row['Upper']):
                key += 1
                constraints[key] = {
                    'Phase' : h,
                    'Team' : row['Team'],
                    'Upper': row['Upper']
                }

    return constraints

# %%
def team_rel_constraint_params(constraints_df):
    constraints = {}
    key = 0
    for index, row in constraints_df.iterrows():
        if row['Phases'] is not None:
            for phase in row['Phases']:
                key += 1
                constraints[key] = {
                    'Phase1' : phase,
                    'Team1' : row['Team1'],
                    'Gamma1': row['Gamma1'],
                    'Phase2' : phase,
                    'Team2' : row['Team2'],
                    'Gamma2': row['Gamma2'],
                }
```

```
        else:
            key += 1
            constraints[key] = {
                'Phase1' : row['Phase1'],
                'Team1' : row['Team1'],
                'Gamma1': row['Gamma1'],
                'Phase2' : row['Phase2'],
                'Team2' : row['Team2'],
                'Gamma2': row['Gamma2'],
            }

    return constraints

# %%
team_rel_constraints = team_rel_constraint_params(team_rel_constraints_df)

# for key in team_rel_constraints.keys():
#     print(team_rel_constraints[key])

# %%
def optimization_input_params(team_df,
                              phases_df,
                              phase_durations_df,
                              nde_df,
                              team_fte_constraints_df,
                              team_rel_constraints):

    team_fte_constraints = team_fte_constraint_params(team_fte_constraints_df)
    team_rel_constraints = team_rel_constraint_params(team_rel_constraints_df)

    pp.pprint(team_fte_constraints)

    input_params = { None: {
        'T'    : { None: team_df.index.to_list() },
        'D'    : { None: team_df.query("Type == 'D'")['Type'].index.to_list() },
        'S'    : { None: team_df.query("Type == 'D'")["Specialization"].unique().tolist() },
```

```
        'SD'   : {
                    s: team_df.query(f"Specialization == '{s}'").index.to_list() \
                        for s in team_df.query("Type == 'D'")["Specialization"].unique().tolist()
                 },
        'NDE'  : nde_df["NDE"].to_dict(),
        'H'    : { None: phases_df.index.to_list() },
        'HD'   : { None: phases_df.query("Delta == 1").index.to_list() },
        'L'    : phase_durations_df["Duration"].to_dict(),
        'rate' : team_df["Rate"].to_dict(),
        'mu'   : team_df["FTE Step"].to_dict(),
        'rho'  : team_df.query("Type == 'D'")["Rho"].to_dict(),
        'g'    : phases_df["GPAC (g)"].to_dict(),
        'r'    : phases_df["CPWTC (r)"].to_dict(),
        'o'    : phases_df["LC (o)"].to_dict(),
        'd'    :  {('BE', h) : d for (h, d) in phases_df.query("Delta == 1")["SDAC BE"].to_dict().items() }
                | {('FE', h) : d for (h, d) in phases_df.query("Delta == 1")["SDAC FE"].to_dict().items() }
                | {('DS', h) : d for (h, d) in phases_df.query("Delta == 1")["SDAC DS"].to_dict().items() },
        'TeamFTE_Index': team_fte_constraints.keys(),
        'TeamFTE': team_fte_constraints,
        'TeamRel_Index': team_rel_constraints.keys(),
        'TeamRel': team_rel_constraints
    }}

    return input_params

# %% [markdown]
# # Solving Optimization Tasks

# %%
class Estimation:

    def __init__(self, alternative_id, schedule_df=None, optimization_task=None) -> None:
        self.alternative_id = alternative_id
        self.schedule_df = schedule_df
        self.optimization_task = optimization_task
```

```
    # Duration Summary
    self.duration_sprints = self.schedule_df['Sprints'].sum()
    self.duration_hours = self.schedule_df['Duration'].sum()
    self.duration_months = self.schedule_df['Duration'].sum() / (WORKING_HOURS_PER_MONTH)

# Inputs
alternative_id = None
schedule_df = None
optimization_task = None
team_df = None
phases_df = None
nde_df = None

# Optimization
optimization_input_params = None
optimization_task_instance = None
optimization_task_solver = None
optimization_results = None
optimization_solved = None

# Outcomes
results_df = None
summary_df = None
ndc_df = None

duration_sprints = None
duration_hours = None
duration_months = None

normalized_cost = None

def create_optimization_task_instance(
        self,
        team_df,
        phases_df,
        nde_df,
```

```
        team_fte_constraints_df,
        team_rel_constraints):

    self.team_df = team_df
    self.phases_df = phases_df
    self.nde_df = nde_df

    self.specializations = []
    for s in self.team_df["Specialization"].unique():
        if s is not None:
            self.specializations.append(s)


    self.optimization_input_params = optimization_input_params(
        self.team_df,
        self.phases_df,
        self.schedule_df,
        self.nde_df,
        team_fte_constraints_df,
        team_rel_constraints)

    self.optimization_task_instance = self.optimization_task \
        .create_instance(self.optimization_input_params)

    return self.optimization_task_instance


def solve_optimization_task(self):
    solver = pyo.SolverFactory("xpress")
    self.optimization_result = solver.solve(self.optimization_task_instance, tee=False)
    return self.optimization_result

def process_optimization_results(self):
    solver = self.optimization_result.solver

    self.optimization_solved = (solver.status == pyo.SolverStatus.ok) \
```

```
    and (solver.termination_condition == pyo.TerminationCondition.optimal)

if self.optimization_solved:

    self.f_dict = {
        p : [self.optimization_task_instance.f[(t, p)].value for t in self.team_df.index] \
            for p in self.phases_df.index
    }

    d_df = self.team_df["Type"].transform(lambda x: 1 if x == "D" else None)

    self.results_df = self.team_df.copy(deep=True)

    summary_columns = []

    for p in self.phases_df.index:
        self.results_df[f"f-{p}"] = self.f_dict[p]

    for p in self.phases_df.index:
        self.results_df[f"FTE-{p}"] = self.results_df["FTE Step"] * self.results_df[f"f-{p}"]
        summary_columns.append(f"FTE-{p}")

    for p in self.phases_df.index:
        self.results_df[f"ND-FTE-{p}"] = self.team_df["Alpha"] * self.results_df[f"FTE-{p}"]
        summary_columns.append(f"ND-FTE-{p}")

    for p in self.phases_df.index:
        self.results_df[f"W-{p}"] = self.results_df[f"FTE-{p}"] * self.schedule_df.at[p, "Duration"]
        summary_columns.append(f"W-{p}")

    for p in self.phases_df.index:
        self.results_df[f"F-{p}"] = self.results_df[f"W-{p}"] * self.results_df["Rate"]
        summary_columns.append(f"F-{p}")

    for p in self.phases_df.index:
        self.results_df[f"R-{p}"] = self.results_df[f"W-{p}"] * \
```

```
        (d_df * self.phases_df.at[p, "CPWTC (r)"])
    summary_columns.append(f"R-{p}")

for p in self.phases_df.index:
    self.results_df[f"O-{p}"] = self.results_df[f"W-{p}"] * \
        (d_df * self.phases_df.at[p, "LC (o)"])
    summary_columns.append(f"O-{p}")

for p in self.phases_df.index:
    self.results_df[f"G-{p}"] = (self.results_df[f"W-{p}"] - self.results_df[f"O-{p}"]) * \
        (d_df * self.phases_df.at[p, "GPAC (g)"])
    summary_columns.append(f"G-{p}")

for p in self.phases_df.index:
    self.results_df[f"M-{p}"] = self.results_df[f"W-{p}"] - self.results_df[f"R-{p}"] - \
        self.results_df[f"O-{p}"] - self.results_df[f"G-{p}"]
    summary_columns.append(f"M-{p}")

def get_SDAC(t, p):
    s = self.team_df.at[t, "Specialization"]
    if s == "BE":
        return self.phases_df.at[p, "SDAC BE"]
    elif s == "FE":
        return self.phases_df.at[p, "SDAC FE"]
    elif s == "DS":
        return self.phases_df.at[p, "SDAC DS"]
    else:
        return None

for p in self.phases_df.index:
    self.results_df[f"SDAC-{p}"] = pd.Series({t: get_SDAC(t, p) for t in self.results_df.index})
    summary_columns.append(f"SDAC-{p}")

for p in self.phases_df.index:
    self.results_df[f"D-{p}"] = self.results_df[f"M-{p}"] / (1 + self.results_df[f"SDAC-{p}"])
    summary_columns.append(f"D-{p}")
```

```
    for p in self.phases_df.index:
        self.results_df[f"C-{p}"] = self.results_df[f"D-{p}"] * self.results_df["Rho"]
        summary_columns.append(f"C-{p}")

    # Summary

    self.summary_df = pd.DataFrame(
        data=[],
        columns=summary_columns,
        index=self.specializations + ['D', 'ND', 'Total'])

    for column in self.summary_df.columns:
        for spec in self.specializations:
            self.summary_df.at[spec, column] = \
                self.results_df[self.results_df["Specialization"] == spec][column].sum()

        self.summary_df.at['D', column] = \
            self.results_df[self.results_df["Type"] == 'D'][column].sum()
        self.summary_df.at['ND', column] = \
            self.results_df[self.results_df["Type"] == 'ND'][column].sum(skipna=False)
        self.summary_df.at['Total', column] = \
            self.results_df[column].sum(skipna=True)

    # Normalized Development Capacity (NDC)

    self.ndc_df = self.nde_df.copy(deep=True)
    ndc_cols = [f"C-{p}" for p in self.phases_df.index]
    self.ndc_df['NDC'] = self.summary_df.loc[self.ndc_df.index][ndc_cols].sum(axis=1)
    self.ndc_df['NDC-NDE'] = self.ndc_df['NDC'] - self.ndc_df['NDE']

    # Total Normalized Cost
    self.normalized_cost = \
        self.summary_df.loc['Total'][[f"F-{p}" for p in self.phases_df.index]].sum()

else:
```

```
            pass

        return self


# %%
# Dictionaries of estimations indexed by project schedule alternative ID
Estimations_OPT = {} # Optimistic estimations
Estimations_PSM = {} # Pessimistic estimations

# %% [markdown]
# ## Project Schedule Alternatives

# %%
schedule_alternatives_dfs = project_schedule_alternatvies(phases_df)

optimization_task = create_estimation_optimization_task()

for key in schedule_alternatives_dfs:
    Estimations_OPT[key] = Estimation(key, schedule_alternatives_dfs[key], optimization_task)
    Estimations_PSM[key] = Estimation(key, schedule_alternatives_dfs[key], optimization_task)

merged_alternatives_df = pd.DataFrame()
for key in schedule_alternatives_dfs:
    merged_alternatives_df[key] = schedule_alternatives_dfs[key]['Sprints']
merged_alternatives_df.index.names = ['Phase']
display(merged_alternatives_df)

# %%
for estimation in Estimations_OPT.values():
    estimation.create_optimization_task_instance(
        team_df,
        phases_df,
        nde_opt_df,
        team_fte_constraints_df,
        team_rel_constraints)
```

```
    estimation.solve_optimization_task()
    estimation.process_optimization_results()

for estimation in Estimations_PSM.values():
    estimation.create_optimization_task_instance(
        team_df,
        phases_df,
        nde_psm_df,
        team_fte_constraints_df,
        team_rel_constraints)
    estimation.solve_optimization_task()
    estimation.process_optimization_results()

# %% [markdown]
# # Optimization Summary

# %%
def summarize_estimations(estimations):
    estimation_summary_df = pd.DataFrame(index=estimations.keys())
    for k, e in estimations.items():

        estimation_summary_df.at[k, 'Sprints'] = e.duration_sprints
        estimation_summary_df.at[k, 'Months'] = e.duration_months

        if e.optimization_solved:
            estimation_summary_df.at[k, 'Cost'] = e.normalized_cost
            for s in e.ndc_df.index:
                estimation_summary_df.at[k, f'NDC: {s}'] = e.ndc_df.at[s, 'NDC']

            ndc_nde_2 = 0
            for s in e.ndc_df.index:
                estimation_summary_df.at[k, f'NDC-NDE: {s}'] = e.ndc_df.at[s, 'NDC-NDE']
                ndc_nde_2 += (e.ndc_df.at[s, 'NDC-NDE'])**2
            estimation_summary_df.at[k, 'NDC-NDE'] = math.sqrt(ndc_nde_2)

            avg_fte = 0
```

```
            for p in e.phases_df.index:
                estimation_summary_df.at[k, f'FTE-{p}'] = e.summary_df.at['Total', f'FTE-{p}']
                avg_fte += (e.summary_df.at['Total', f'FTE-{p}'] * e.schedule_df.at[p, 'Sprints'])
            estimation_summary_df.at[k, 'AVG FTE'] = (avg_fte / e.duration_sprints)

    return estimation_summary_df

# %%
summary_estimation_opt_df = summarize_estimations(Estimations_OPT)
display(summary_estimation_opt_df)

# %%
summary_estimation_psm_df = summarize_estimations(Estimations_PSM)
display(summary_estimation_psm_df)

# %% [markdown]
# # Ranking of the Alternatives

# %%
def do_compare_numeric(values, max_importance, lower_better):
    max_diff = abs(max(values) - min(values))
    def do_compare(r, c):
        v = round((abs(values[r] - values[c]) / max_diff) * (max_importance - 1)) + 1
        if lower_better:
            if values[r] < values[c]:
                return v
            elif values[r] > values[c]:
                return 1.0 / v
            else:
                return 1
        else:
            if values[r] > values[c]:
                return v
            elif values[r] < values[c]:
                return 1.0 / v
            else:
```

```
            return 1
    return do_compare

def alternatives_comparison(keys, values, do_compare):
    comparison = {}
    for i in range(0, len(keys)):
        for j in range(i + 1, len(keys)):
            comparison[(keys[i], keys[j])] = do_compare(keys[i], keys[j])

    return comparison

def compare_project_scenarios(scenarios_summary_df):

    # ======== Criteria =========
    # 0 - cost
    # 1 - duration
    # 2 - team_size
    # 3 - idle_time

    MAX_IMPORTANCE = 7

    criteria_comparison = {
        ('cost', 'duration'): 3, ('cost', 'team_size'): 3, ('cost', 'idle_time'): 7,
        ('duration', 'team_size'): (1 / 7) , ('duration', 'idle_time'): 7,
        ('team_size', 'idle_time'): 5
    }

    criteria_importance = ahpy.Compare(
        name="Criteria Importance",
        comparisons=criteria_comparison,
        precision=3,
        random_index='dd')
    #print("criteria_importance.target_weights:", criteria_importance.target_weights)

    summary_feasible_df = scenarios_summary_df[(scenarios_summary_df['Cost'].notnull())]
    #display(opt_summary_feasible_df)
```

```
cost_comparison = alternatives_comparison(
    summary_feasible_df.index, summary_feasible_df["Cost"],
    do_compare_numeric(summary_feasible_df["Cost"], MAX_IMPORTANCE, True))

duration_comparison  = alternatives_comparison(
    summary_feasible_df.index, summary_feasible_df["Sprints"],
    do_compare_numeric(summary_feasible_df["Sprints"], MAX_IMPORTANCE, True))

team_size_comparison = alternatives_comparison(
    summary_feasible_df.index, summary_feasible_df["AVG FTE"],
    do_compare_numeric(summary_feasible_df["AVG FTE"], MAX_IMPORTANCE, True))

idle_time_comparison = alternatives_comparison(
    summary_feasible_df.index, summary_feasible_df["NDC-NDE"],
    do_compare_numeric(summary_feasible_df["NDC-NDE"], MAX_IMPORTANCE, True))

cost_importance = ahpy.Compare(
    name="cost", comparisons=cost_comparison, precision=3, random_index='dd')
duration_importance = ahpy.Compare(
    name="duration", comparisons=duration_comparison, precision=3, random_index='dd')
team_size_imporrance = ahpy.Compare(
    name="team_size", comparisons=team_size_comparison, precision=3, random_index='dd')
idle_time_importance = ahpy.Compare(
    name="idle_time", comparisons=idle_time_comparison, precision=3, random_index='dd')

criteria_importance.add_children([
    cost_importance,
    duration_importance,
    team_size_imporrance,
    idle_time_importance
])

scenarios_summary_df["Rank"] = float("nan")
for index, row in scenarios_summary_df.iterrows():
     if not pd.isna(row["Cost"]):
```

```
            scenarios_summary_df.at[index, 'Rank'] = criteria_importance.target_weights[index]

    return scenarios_summary_df

# %%
compare_project_scenarios(summary_estimation_opt_df)
display(summary_estimation_opt_df)

# %%
compare_project_scenarios(summary_estimation_psm_df)
display(summary_estimation_psm_df)

# %% [markdown]
# # Export to Excel

# %%
def export_estimations_to_excel(estimations, estimation_summary_df, file_name):
    with pd.ExcelWriter(file_name) as writer:
        for k, e in estimations.items():
            df_to_export = pd.concat([e.results_df, e.summary_df]) \
                if e.optimization_solved else pd.DataFrame()
            df_to_export.to_excel(writer, sheet_name=f"ALT-{k}", engine='xlsxwriter')

        estimation_summary_df.to_excel(writer, sheet_name=f"Summary", engine='xlsxwriter')

# %%
export_estimations_to_excel(
    Estimations_OPT,
    summary_estimation_opt_df,
    "export/Intermediate_Estimations_OPT.xlsx")

export_estimations_to_excel(
    Estimations_PSM,
    summary_estimation_psm_df,
    "export/Intermediate_Estimations_PSM.xlsx")
```

Додаток 13. Акти впровадження результатів дисертаційного дослідження

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
д.т.н., проф. Іван ДЕМИДОВ
«03» 03 2025 р.

АКТ
про використання результатів дисертаційної роботи
Войтишина Володимира Володимировича
«Інформаційна технологія оцінювання проєктів
з розробки програмного забезпечення»,
представленої на здобуття наукового ступеня доктора філософії
за спеціальністю 122 «Комп'ютерні науки»,
при виконанні науково-дослідної роботи за темою:
«Інтелектуальні інформаційні технології
багаторівневого управління енергоефективністю регіону»

Комісія у складі начальника НДЧ д.т.н., ст. досл. Романа НЕБЕСНОГО та членів: зав. відділу науково-організаційного супроводу наукових досліджень к.т.н. Галини ЛАЗЬКО, завідувача кафедри автоматизованих систем управління д.т.н. проф. Василя ТЕСЛЮКА та заст. начальника планово-фінансового відділу Ірини ФАСТ – цим актом підтверджують, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії Войтишина Володимира Володимировича на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки» використані при виконанні науково-дослідної роботи, яка виконувалася за рахунок коштів загального фонду державного бюджету за темою: «Інтелектуальні інформаційні технології багаторівневого управління енергоефективністю регіону» (номер державної реєстрації 0117U004450).

Зокрема, Володимиром ВОЙТИШИНОМ було розроблено метод побудови схеми слабоструктурованого бізнес-процесу з підтримкою опрацювання потоків даних (Розділ 3. Оцінювання проєктів з розробки програмного забезпечення як слабоструктурований бізнес-процес).

Голова комісії:

Начальник науково-дослідної частини,
д.т.н., ст. досл. Роман НЕБЕСНИЙ

Члени комісії:

Зав. відділу науково-організаційного
супроводу наукових досліджень Галина ЛАЗЬКО

В.о. заступника начальника
планово-фінансового відділу Ірина ФАСТ

Завідувач кафедри
автоматизованих систем управління, д.т.н., проф. Василь ТЕСЛЮК

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
д.т.н., проф. Іван ДЕМИДОВ
«___» _____ 2025 р.

АКТ
про використання результатів дисертаційної роботи
Войтишина Володимира Володимировича
«Інформаційна технологія оцінювання проєктів
з розробки програмного забезпечення»,
представленої на здобуття наукового ступеня доктора філософії
за спеціальністю 122 «Комп'ютерні науки»,
при виконанні науково-дослідної роботи за темою:
«Методи та засоби інтелектуального вимірювання параметрів руху та визначення просторової орієнтації наземних мобільних робототехнічних платформ»

Комісія у складі начальника НДЧ д.т.н., ст. досл. Романа НЕБЕСНОГО та членів: зав. відділу науково-організаційного супроводу наукових досліджень к.т.н. Галини ЛАЗЬКО, завідувача кафедри автоматизованих систем управління д.т.н. проф. Василя ТЕСЛЮКА та заст. начальника планово-фінансового відділу Ірини ФАСТ – цим актом підтверджують, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії Войтишина Володимира Володимировича на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки» використані при виконанні науково-дослідної роботи, яка виконувалася за рахунок коштів загального фонду державного бюджету за темою: «Методи та засоби інтелектуального вимірювання параметрів руху та визначення просторової орієнтації наземних мобільних робототехнічних платформ» (номер державної реєстрації 0124U000822).

Зокрема, Володимиром ВОЙТИШИНОМ було розроблено метод підтримки прийняття рішень щодо складу команди та графіку реалізації проєкту з розробки програмного забезпечення (Розділ 2. Розроблення методів оцінювання проєктів з розробки програмного забезпечення).

Голова комісії:

Начальник науково-дослідної частини,
д.т.н., ст. досл. Роман НЕБЕСНИЙ

Члени комісії:

Зав. відділу науково-організаційного
супроводу наукових досліджень Галина ЛАЗЬКО

В.о. заступника начальника
планово-фінансового відділу Ірина ФАСТ

Завідувач кафедри
автоматизованих систем управління, д.т.н., проф. Василь ТЕСЛЮК

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
д.т.н., проф. Іван ДЕМИДОВ
«__» _____ 03 _____ 2025 р.

АКТ
про використання результатів дисертаційної роботи
Войтишина Володимира Володимировича
«Інформаційна технологія оцінювання проєктів
з розробки програмного забезпечення»,
представленої на здобуття наукового ступеня доктора філософії
за спеціальністю 122 «Комп'ютерні науки»,
при виконанні науково-дослідної роботи за темою:
«Методи та засоби нейронечіткого управління
групою мобільних робототехнічних платформ»

Комісія у складі начальника НДЧ д.т.н., ст. досл. Романа НЕБЕСНОГО та членів: зав. відділу науково-організаційного супроводу наукових досліджень к.т.н. Галини ЛАЗЬКО, завідувача кафедри автоматизованих систем управління д.т.н. проф. Василя ТЕСЛЮКА та заст. начальника планово-фінансового відділу Ірини ФАСТ – цим актом підтверджують, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії Войтишина Володимира Володимировича на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки» використані при виконанні науково-дослідної роботи, яка виконувалася за рахунок коштів загального фонду державного бюджету за темою: «Методи та засоби нейронечіткого управління групою мобільних робототехнічних платформ» (номер державної реєстрації 0123U101688).

Зокрема, Володимиром ВОЙТИШИНОМ було розроблено метод поетапного оцінювання проєктів з розробки програмного забезпечення (Розділ 2. Розроблення методів оцінювання проєктів з розробки програмного забезпечення), що охоплює стадії аналізу та проєктування, які передують початку реалізації проєкту, та передбачає послідовне виконання трьох етапів оцінювання: попереднього, проміжного та детального.

Голова комісії:

Начальник науково-дослідної частини,
д.т.н., ст. досл. Роман НЕБЕСНИЙ

Члени комісії:

Зав. відділу науково-організаційного
супроводу наукових досліджень Галина ЛАЗЬКО

В.о. заступника начальника
планово-фінансового відділу Ірина ФАСТ

Завідувач кафедри
автоматизованих систем управління, д.т.н., проф. Василь ТЕСЛЮК

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи
Національного університету
«Львівська політехніка»
к.т.н., доц. Олег ДАВИДЧАК
«___» _____ 2025 р.

АКТ

про впровадження в навчальний процес результатів дисертаційної роботи Войтишина Володимира Володимировича

Цей акт складено про те, що результати дисертаційної роботи Войтишина Володимира Володимировича на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки» на тему «Інформаційна технологія оцінювання проєктів з розробки програмного забезпечення» впроваджено у навчальний процес кафедри автоматизованих систем управління Національного університету «Львівська політехніка».

Впровадження результатів дисертаційної роботи полягає у їхньому використанні при викладанні навчальних дисциплін як окремих розділів лекційних курсів, так і в циклах лабораторних робіт.

Для викладання дисципліни «Командна робота і презентаційні навички» для студентів освітньо-кваліфікаційного рівня «бакалавр», що навчаються за спеціальністю 122 «Комп'ютерні науки», використано наступні результати дисертаційної роботи:

– огляд, класифікація та порівняння методів оцінювання проєктів з розробки програмного забезпечення (Розділ 1. Аналіз існуючих методів оцінювання проєктів з розробки програмного забезпечення);

– метод поетапного оцінювання проєктів з розробки програмного забезпечення, що застосовний під час аналізу та проєктування та передбачає підготовку попередньої, проміжної та детальної оцінок (Розділ 2. Розроблення методів оцінювання проєктів з розробки програмного забезпечення);

– метод побудови розкладу реалізації елементів оцінювання (Розділ 2. Розроблення методів оцінювання проєктів з розробки програмного забезпечення).

Для викладання дисципліни «Математичні методи дослідження операцій» для іноземних студентів освітньо-кваліфікаційного рівня «бакалавр», що навчаються за спеціальністю 122 «Комп'ютерні науки», використано наступні результати дисертаційної роботи:

– метод підтримки прийняття рішень щодо складу проєктної команди та графіку реалізації проєкту, що, в свою чергу, передбачає розв'язання задач цілочисельного програмування та застосування методу аналізу ієрархій для ранжування альтернатив (Розділ 2. Розроблення методів оцінювання проєктів з розробки програмного забезпечення).

Доцент кафедри АСУ,
к.т.н., доц. Анатолій БАТЮК

Завідувач кафедри АСУ,
д.т.н., проф. Василь ТЕСЛЮК

Директор ІКНІ,
д.т.н., проф. Наталія ШАХОВСЬКА