ДЕРЖАВНА СЛУЖБА УКРАЇНИ З НАДЗВИЧАЙНИХ СИТУАЦІЙ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ БЕЗПЕКИ ЖИТТЄДІЯЛЬНОСТІ

Кваліфікаційна наукова
праця на правах рукопису

НАЗАР (КОРДУНОВА) ЮЛІЯ СЕРГІЇВНА

УДК 004.4:004.5

МОДЕЛІ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ДИНАМІЧНИХ УМОВАХ

122 Комп'ютерні науки
12 Інформаційні технології

Дисертація на здобуття наукового ступеня
доктора філософії

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

_____ Ю. С. Назар

Наукові керівники: Придатко Олександр Володимирович, кандидат технічних наук, доцент, Смотр Ольга Олексіївна, кандидат технічних наук, доцент.

Львів – 2024

АНОТАЦІЯ

Назар Ю. С. Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах. –Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії зі спеціальності 122 «Комп'ютерні науки» (галузь знань 12 Інформаційні технології). Львівськийдержавний університет безпеки життєдіяльності Державної служби України з надзвичайних ситуацій, Львів, 2024.

У дисертаційній роботі на підставі проведених досліджень розв'язано важливу науково-прикладну задачу підвищення ефективності процесу планування розроблення спеціалізованого програмного забезпечення у динамічних умовах, яка базується на обґрунтованих моделях та інформаційній технології. Зокрема, запропоновано концептуальну модель управління життєвим циклом розроблення програмного забезпечення, в основі якої є мережевий граф планування, а також імітаційні моделі обходу та визначення його основних показників. Даний підхід дає можливість підвищити ефективність планування окремих етапів розробки програмного забезпечення та здійснювати коригування термінів випуску MVP (мінімально-життєздатного продукту) в режимі реального часу.

Об'єкт дослідження – процеси опрацювання потоків інформації щодо управління життєвим циклом розробки програмного забезпечення в динамічному оточенні.

Предмет дослідження – моделі, методи та засоби інформаційної технології оперативного прийняття рішень щодо ітераційного планування обсягу робіт з розробки спеціалізованого програмного забезпечення в динамічному оточенні.

У роботі проаналізовано існуючі методи та моделі управління життєвим циклом програмного забезпечення та зроблено висновок про невідповідність сучасних методів та підходів управління життєвим

циклом розроблення програмного забезпечення до розроблення безпеко-орієнтованих сервісів, оскільки окрім змінних вимог принципово важливим є час виконання.

Обґрунтовано вибір гнучких підходів управління ІТ-проєктами для впровадження інноваційних підходів до процесу розроблення спеціалізованого безпеко-орієнтованого програмного забезпечення.

Систематизовано основні концепції методів гнучкої методології управління життєвим циклом розроблення програмних систем, зважаючи на специфіку процесу виконання проєктів для служби порятунку, розширено існуючі емпіричні дані про можливості та переваги використання гнучких методів у безпековій галузі.

Розроблено концептуальну модель процесу управління життєвим циклом розроблення спеціалізованого програмного забезпечення (безпеко-орієнтованих сервісів), котра адаптована під специфіку роботи Державної служби України із надзвичайних ситуацій та корелює із принципами гнучкої методології управління життєвим циклом розроблення програмного забезпечення.

Розроблено модель процесу розробки програмних систем із використанням понятійного апарату теорії множин, в результаті чого отримано математичну модель життєвого циклу даних систем, яка дає змогу охарактеризувати обсяги робіт на окремих етапах розроблення програмного продукту, встановити взаємозв’язки та взаємозалежності між ними та сформувати фундаментальні принципи автоматизації процедури підтримки прийняття управлінських рішень на різних етапах розроблення програмних систем, зокрема безпеко-орієнтованого спрямування.

Розроблені імітаційні моделі процесів обходу мережевих графів різної складності та конфігурації, надають змогу вирішувати задачі короткотермінового планування проєктів із розроблення безпеко-орієнтованих сервісів у динамічних умовах шляхом розрахунку

раннього та пізнього термінів виконання подій з одержанням на їх основі характеристики щодо тривалості проєктних робіт. Розроблені моделі дають змогу побудувати алгоритми обходу мережевого графа для оцінки тривалості виконання проєктних робіт з подальшою автоматизацією процесів короткотермінового планування за умови введення будь-яких змін до змісту та обсягу окремих спринтів проєкту або складу команди розробників.

Розроблена інформаційна технологія підтримки прийняття рішень в управлінні життєвим циклом розроблення спеціалізованого програмного забезпечення шляхом впровадження у неї отриманих моделей та алгоритмів для вирішення задачі короткотермінового планування розроблення безпеко-орієнтованих сервісів. Дана інформаційна технологія здійснює автоматизацію процесів короткотермінового планування, а також швидко і якісно переоцінює тривалість робіт за умови введення будь яких змін до змісту, обсягу, часу виконання окремих спринтів проєкту або складу команди розробників.

Практичне значення отриманих результатів полягає в тому, що розроблені моделі та інформаційна технологія реалізовані у форматі web-застосунку, що, у свою чергу, значно спрощує менеджмент життєвого циклу розроблення програмного забезпечення шляхом автоматизації процесу короткотермінового планування у динамічному середовищі.

Ключові слова: життєвий цикл програмного забезпечення, система підтримки прийняття рішень, Agile методологія, безпеко-орієнтований сервіс, програмне забезпечення, імітаційна модель, мережевий граф планування, мережі Петрі, веб-сервіс інформаційна система.

ABSTRACT

Nazar Yu. S. Models and information technology of software development life cycle management in dynamic conditions – scientific work on the rights of the manuscript.

Dissertation for the degree of Doctor of Philosophy in the specialty 122 Computer Sciences (12 Information Technologies). Lviv State University of Life Safety of the State Emergency Service of Ukraine, Lviv, 2024

The dissertation solves an important scientific and applied problem of improving the efficiency of the planning process of specialised software development in dynamic conditions, based on substantiated models and information technology. In particular, a conceptual model for managing the software development life cycle is proposed, based on a network planning graph, as well as simulation models for traversing and determining its main key indicators. This approach makes it possible to increase the efficiency of planning individual stages of software development and to adjust the timing of the release of the minimum viable product (MVP) in real time.

The object of research is the processes of processing information flows for managing the software development life cycle in a dynamic environment.

Subject of research - models, methods and means of information technology for operational decision-making on iterative planning of the scope of work on the development of specialised software in a dynamic environment.

The work analyses the existing methods and models of software life cycle management and concludes that modern methods and approaches to software development life cycle management are not suitable for the development of safety-oriented services. In addition to variable requirements, the execution time is of fundamental importance.

The choice of flexible approaches to IT project management for the implementation of innovative approaches to the process of developing specialised security-oriented software is substantiated.

The basic concepts of methods of agile methodology for managing the life cycle of software systems development are systematised, taking into account the specifics of the process of project implementation for the rescue service, and the existing empirical data on the possibilities and benefits of using agile methods in the security industry are expanded.

A conceptual model of the process of managing the life cycle of specialised software development ("safety-oriented services") is developed, which is adapted to the specifics of the State Emergency Service of Ukraine and correlates with the principles of a flexible methodology for managing the life cycle of software development.

A model of the software systems development process has been developed using the conceptual apparatus of set theory, which resulted in a mathematical model of the life cycle of these systems, which allows characterising the scope of work at certain stages of software product development, establishing interconnections and interdependencies between them and forming the fundamental principles of automation of the procedure for supporting management decision-making at various stages of software systems development, in particular, safety-oriented ones.

The developed imitation models of network graph traversal processes of varying complexity and configuration make it possible to solve the problems of short-term planning of projects for the development of safety-oriented services in dynamic conditions by calculating the early and late deadlines for the execution of events and obtaining characteristics of the duration of project work on their basis. The developed models make it possible to build algorithms for traversing the network graph to estimate the duration of project work with subsequent automation of short-term planning processes, subject to any changes to the content and scope of individual project sprints or the composition of the development team.

An information technology for decision support in managing the life cycle of specialised software development has been developed by

implementing the obtained models and algorithms to solve the problem of short-term planning of safety-oriented services development. This information technology automates short-term planning processes and quickly and efficiently reassesses the duration of work in the event of any changes to the content, scope, time of individual project sprints or the composition of the development team.

The practical significance of the obtained results is that the developed models and information technology are implemented in the format of a web application, which, in turn, greatly simplifies the management of the software development life cycle by automating the short-term planning process in a dynamic environment.

Keywords: software life cycle, decision support system, Agile methodology, safety-oriented service, software, simulation model, network planning graph, Petri nets, web-service, information system.

Список публікацій здобувача

Наукові праці, в яких опубліковані основні наукові результати дисертації:

Статті у міжнародних наукових виданнях і тих, що входять до міжнародних наукометричних баз (МНБ):

1. **Kordunova Y.**, Prydatko O., Smotr O., Golovatyi R. Expert Decision Support System Modeling in Lifecycle Management of Specialized Software. Lecture Notes on Data Engineering and Communications Technologies, Springer, Switzerland. Vol. 149, 2022, pp. 367-383, https://doi.org/10.1007/978-3-031-16203-9_22 (1,06 друк. арк.). **Видання входить до МНБ – Scopus.** *Особистий внесок автора полягає у розробці алгоритмів обходу мережевого графа та становить 0,26 друк. арк.*

Статті у наукових фахових виданнях України:

2. **Кордунова Ю. С.**, Смотр О. О., Кокотко І. Я., Малець Р. Б. Аналіз традиційного та гнучкого підходів до створення програмного забезпечення в динамічних умовах. Управління розвитком складних систем. Київ, 2021. № 47. С. 71 – 77, https://doi.org/10.32347/2412-9933.2021.47.71-77. (0,44 друк. арк.) *Особистий внесок автора полягає у проведеному аналізі існуючих підходів до розробки програмного забезпечення, що становить 0,11 друк. арк.*

3. **Кордунова Ю.**, Фелтіновскі М., Придатко О., Смотр О. Математичне моделювання процесу розробки спеціалізованих програмних систем безпеко-орієнтованого спрямування. Вісник Львівського державного університету безпеки життєдіяльності. Львів, 2023. № 27. С. 23-31. https://doi.org/https://doi.org/10.32447/20784643.27.2023.03. (0,56 друк. арк.) *Особистий внесок автора полягає у розробленні математичної моделі процесу розробки програмного забезпечення із використанням теорії множин, що становить 0,14 друк. арк.*

4. **Кордунова Ю. С.** Аналіз та розроблення концептуальної моделі управління життєвим циклом спеціалізованого програмного забезпечення безпеко-орієнтованого спрямування. Український журнал інформаційних технологій. 2023. Т. 5, № 2. С. 72–78. https://doi.org/10.23939/ujit2023.02.072 (0,44 друк. арк.)

5. **Назар Ю.,** Придатко О. Моделювання процесу обходу мережевого графа для розв'язання задач короткострокового планування IT-проєктів. Вісник Львівського державного університету безпеки життєдіяльності. Львів, 2024. № 29. С. 32-43. (0,75 друк. арк.) *Особистий внесок автора полягає у проведеному аналізі існуючих підходів до розробки програмного забезпечення, що становить 0,38 друк. арк.*

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. **Kordunova Yu.**, Prydatko O., Smotr O., Kokotko I. The network graph traversal method for solving the problem of short-term planning of safety-oriented services development. Monografia powstała w ramach Projektu dofinansowanego przez Ministra Edukacji i Nauki ze środków budżetu państwa w

ramach programu „Doskonała Nauka”, Warszawa 2022. p. 172 – 181. (0,625 друк. арк.) **(Видання є частиною закордонної монографії)** *Особистий внесок автора полягає у розробці методу короткострокового планування безпеко-орієнтованих сервісів та становить 0,16 друк. арк.*

7. Хлевной О., **Кордунова Ю.**, Райта Д., Гаврись А, Ковальчук В. Розробка та реалізація алгоритму розрахунку тривалості евакуації під час пожежі за спрощеною аналітичною моделлю. Надзвичайні ситуації: попередження та ліквідація. 2023. Т. 7, № 1. С. 169 – 181 (0,81 друк. арк.) *Особистий внесок автора полягає в розробці програмного забезпечення для розрахунку тривалості евакуації та становить 0,16 друк. арк.*

8. **Кордунова Ю. С.**, Придатко О. В., Смотр О. О. Переваги використання Agile- методології під час розробки програмного забезпечення в умовах сучасного ринку. Інформаційна безпека та інформаційні технології : зб. наук. праць IV Всеукр. наук.-практ. конф. молодих учених, студентів і курсантів. м. Львів 27 листопада 2020 р. Львів, 2020. С. 206-207. (0,125 друк. арк.) *Особистий внесок автора полягає в аналізі існуючих на сьогоднішній день моделей управління життєвим циклом програмного забезпечення та становить 0,04 друк. арк.*

9. **Кордунова Ю. С.**, Смотр О. О. Сенс Agile-маніфесту для сучасного проєкт-менеджменту. Проблеми та перспективи розвитку системи безпеки життєдіяльності: зб. наук. праць XVI Міжнар. наук.-практ. конф. молодих вчених, курсантів та студентів. – Львів: ЛДУ БЖД, 2021. С. 247-248. (0,125 друк. арк.) *Особистий внесок автора полягає у аналізі та випливі agile-маніфесту на сучасний проєкт-менеджмент та становить 0,04 друк. арк.*

10. **Кордунова Ю. С.,** Смотр О. О. Визначення ефективності використання Agile методології в сучасних організаціях. Проблеми та перспективи забезпечення цивільного захисту: матеріали міжнародної науково-практичної конференції молодих учених. Харків: НУЦЗУ, 2021. 166 с. (0,0625 друк. арк.) *Особистий внесок автора полягає у визначенні*

ефективності використання Agile методології в сучасних організаціях та становить 0,03 друк. арк.

11. **Кордунова Ю.**, Придатко О., Смотр О. Обґрунтування розподілу пріорітетів розробки програмного продукту у динамічному оточенні. Інформаційна безпека та інформаційні технології : зб. наук. праць V Всеукр. наук.-практ. конф. молодих учених, студентів і курсантів. м. Львів 26 листопада 2021 р. Львів, 2021. С. 140-142 (0,1875 друк. арк.) *Особистий внесок автора полягає у розв'язанні задачі вибору пріоритету між завданнями, які мають великий ризик та велику цінність для створення програмного продукту та становить 0,0625 друк. арк.*

12. Горностай Ю., **Кордунова Ю.** Програмна система «SOS» – пріоритетний спосіб зменшити ризик втрати життя та здоров'я населення. Інформаційна безпека та інформаційні технології : зб. наук. праць VI Всеукр. наук.-практ. конф. молодих учених, студентів і курсантів. м. Львів 30 листопада 2023 р. Львів, 2023. С. 266-268. (0,19 друк. арк.) *Особистий внесок автора полягає у менторстві студентським R&D проєктом та становить* 0,09 *друк. арк.*

13. Мечус Х., **Кордунова Ю.,** Смотр О. Сучасні інформаційні технології в управління ІТ проєктами. Інформаційна безпека та інформаційні технології : зб. наук. праць VI Всеукр. наук.-практ. конф. молодих учених, студентів і курсантів. м. Львів 30 листопада 2023 р. Львів, 2023. С. 353-355. (0,19 друк. арк.) *Особистий внесок автора полягає у менторстві студентським R&D проєктом та становить* 0,06 *друк. арк.*

14. **Кордунова Ю.**, Придатко О. Концептуальна модель процесу управління життєвим циклом спеціалізованого програмного забезпечення. Стратегічні комунікації у сфері забезпечення національної безпеки та оборони: проблеми, досвід, перспективи : IV міжнар. наук.-практ. конф., 27 верес. 2023 р.: тези доповідей / Міністерство оборони України, НУОУ. К.:НУОУ, 2023. С. 173 – 176. (0,25 друк. арк.) *Особистий внесок автора полягає у розробці*

концептуальної моделі процесу управління життєвим циклом спеціалізованого програмного забезпечення 0,125 друк. арк.

15. **Назар Ю.** Придатко О. Використання мереж Петрі в управлінні життєвим циклом безпеко-орієнтованого програмного забезпечення. Проблеми та перспективи розвитку системи безпеки життєдіяльності: Зб. наук. праць Міжнародної науково-практичної конференції молодих вчених, курсантів та студентів. – Львів: ЛДУ БЖД, 2024. С. 521 - 526. (0,375 друк. арк.) *Особистий внесок автора полягає в обґрунтуванні розроблених алгоритмів короткострокового планування розробки спеціалізованого програмного забезпечення із використанням мереж Петрі та становить 0,19 друк. арк.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVC	модель-представлення-контроллер
MVP (МЖП)	minimum viable product (мінімально життєздатний продукт)
R&D	Research and Development
UML	Unified Modeling Language
БОС	безпеко-орієнтований сервіс
ДСНС	Державна служба України із надзвичайних ситуацій
ЖЦ ПЗ	життєвий цикл програмного забезпечення
ІС	інформаційна система
ІСППР	інформаційна система підтримки прийняття рішень
МВС	Міністерство внутрішніх справ
НДР	науково-дослідна робота;
НС	надзвичайні ситуації;
ПП	програмний продукт
ПЗ	програмне забезпечення
СПЗ	спеціалізоване програмне забезпечення
ТЗ	технічне завдання;

ВСТУП

Актуальність теми. Існуючі моделі, методи та механізми гнучкого управління ІТ-проєктами, зокрема із розробки програмного забезпечення (далі – ПЗ), передбачають широкий спектр інструментальних засобів реалізації цілей проєкту. Відомі підходи апробовані реалізацією низки успішних проєктів у сфері аутсорсингу та продуктових компаній. Проте, як показує аналіз ринку праці та попереднє вивчення бізнес процесів у проєктах розробки спеціалізованого ПЗ, реалізація окремих етапів життєвого циклу розроблення такого ПЗ вимагає залучення додаткових фахівців вузької галузі. Особливої уваги у подібного роду проєктах потребує процес короткотермінового планування (планування спринтів). Проблема короткотермінового планування полягає у необхідності коректного визначення обсягу робіт та застосовуваних технологій, що, своєю чергою, впливає на успішність виконання робіт в умовах обмежених ресурсів. Зважаючи на існуючі методи короткотермінового планування, можна зробити висновок про їх часткову невідповідність процесам планування обсягів робіт у проєктах розробки саме спеціалізованого ПЗ (далі – СПЗ), зокрема у галузі безпеки людини. Подібні проєкти, як правило, характеризуються значною невизначеністю вхідних даних та динамічними специфікаціями. Такі передумови створюють несприятливий клімат у команді розробників ПЗ, внаслідок чого дестабілізації зазнає продуктивність цієї команди.

Варто також зауважити, що членами команд розробки безпеко-орієнтованих сервісів (далі – БОС), як правило, є кадрові працівники відповідних служб, які в міру своєї службової підготовки повинні поєднувати роботу, пов'язану із розробкою зазначених сервісів, з іншими різновидами оперативної та/або службової діяльності. Тому, зважаючи на специфіку роботи і особливих учасників проєктних команд із розробки БОС, динаміка проєктного середовища набуває дещо іншого значення.

Саме тому існує потреба у побудові нової моделі та інформаційної технології для розробки безпеко-орієнтованих сервісів, що відповідатиме парадигмі гнучкого управління процесом розробки програмного забезпечення в динамічних умовах оперативних формувань.

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

Мета і завдання дослідження. Метою роботи є удосконалити процес планування для розробки спеціалізованого програмного забезпечення безпеко-орієнтованого спрямування шляхом розроблення нових моделей та інформаційної технології підтримки прийняття рішень в управлінні життєвим циклом розроблення програмного забезпечення у динамічних умовах.

Для досягнення мети обумовлено виконання таких завдань:

- провести аналіз предметної області та наукових праць, присвячених питанням ефективності управління життєвим циклом розроблення програмного забезпечення;
- дослідити обсяги робіт на окремих етапах розроблення програмного забезпечення, встановити взаємозв’язки і взаємозалежності між ними та сформувати фундаментальні принципи автоматизації процедури підтримки прийняття управлінських рішень на різних етапах розроблення програмних систем;
- розробити концептуальну модель управління життєвим циклом розроблення спеціалізованого програмного забезпечення безпеко-орієнтованого спрямування;
- дослідити та розробити імітаційні моделі обходу мережевих графів планування задля подальшого їх використання для побудови алгоритмів обчислення ранніх та пізніх термінів виконання подій на мережевих графах різної складності та конфігурації;
- розробити алгоритми побудови та обходу графа мережевої моделі для підвищення ефективності процесів планування окремих етапів розробки програмного забезпечення;

- розробити компоненти інформаційної технології підтримки управлінських рішень щодо ітераційного планування обсягу робіт з розроблення спеціалізованого програмного забезпечення на підставі розроблених та адаптованих моделей та алгоритмів оцінювання спроможності виконання командою розробників декларованого обсягу проєктних робіт;
- виконати проєктування, реалізацію та апробацію запропонованої інформаційної технології підтримки управлінських рішень в процесі розроблення безпеко-орієнтованих сервісів у Львівському державному університеті безпеки життєдіяльності.

Об’єкт дослідження – процеси опрацювання потоків інформації щодо управління життєвим циклом розроблення програмного забезпечення в динамічному оточенні.

Предмет дослідження – моделі, методи та засоби інформаційної технології оперативного прийняття рішень щодо ітераційного планування обсягу робіт з розроблення спеціалізованого програмного забезпечення в динамічному оточенні.

Наукова новизна одержаних результатів.

вперше розроблено:

- імітаційну модель обходу мережевого графа шляхом відтворення паралельних (розподілених) процесів мережами Петрі, що дозволяє її застосування для розробки алгоритмів та автоматизації розрахунку параметрів мережевого графа планування ІТ-проєкту;
- концептуальну модель процесу управління життєвим циклом розроблення програмного забезпечення, що корелює із принципами гнучкої методології управління ІТ-проєктами для підвищення якості менеджменту розроблення програмного забезпечення безпеко-орієнтованого спрямування в динамічних умовах.

удосконалено:

- метод планування життєвого циклу розроблення програмного забезпечення шляхом використання мережевого графа для можливості оперативного перепланування проєкту в динамічних умовах;

набула подальшого розвитку:

- інформаційна технологія, покладена в основу розроблення інформаційної системи підтримки прийняття рішень щодо управління життєвим циклом розроблення спеціалізованого програмного забезпечення, яка базується на обґрунтованих методах, моделях та алгоритмах для підвищення ефективності менеджменту життєвого циклу розроблення програмного забезпечення безпеко-орієнтованого спрямування.

Практичне значення одержаних результатів полягає у імплементації розроблених та науково-обґрунтованих методів, моделей і алгоритмів в процеси управління життєвим циклом розроблення спеціалізованого програмного забезпечення у динамічних умовах.

Результати виконаних досліджень впроваджені в освітній процес Львівського державного університету безпеки життєдіяльності при реалізації студентських R&D проєктів, а також використані в науково-дослідних роботах на замовлення Державної служби України із надзвичайних ситуацій (акт впровадження від 04.06.2024р.). Зокрема, на базі розроблених методів та моделей удосконалено процес управління життєвим циклом розроблення наступних безпеко-орієнтованих сервісів: Інформаційно-аналітична система «Інтерактивний інспектор» (на замовлення ДСНС України, акт впровадження від 10.06.2024 р.), Веб-сервіс обліку пунктів вакцинації та тестування на COVID-19 (студентський R&D проєкт), інформаційна система «Я-Доброволець» (ініціативна, на замовлення ЛДУ БЖД, ДСНС, МВС України, акт впровадження від 06.06.2024 р).

Особистий внесок здобувача. Усі основні наукові положення, розробки та практичні результати дисертаційної роботи, які виносяться на захист, отримано автором самостійно. У роботах, які виконано у співавторстві, особисто автору належать такі наукові результати: виконано аналіз предметної галузі, науки та практики відомих методологій та методів управління життєвим циклом програмного забезпечення [2, 4, 8, 13], обґрунтовано потребу в розробленні та удосконаленні моделей, методів та інформаційної технології управління життєвим циклом розроблення спеціалізованого програмного забезпечення [1, 2, 3, 4, 5, 7 11], обґрунтовано вплив гнучких методів управління життєвим циклом програмного забезпечення на інновації спеціалізованого безпеко-орієнтованого програмного забезпечення, підвищення ефективності та надійності таких програмних продуктів; [1, 4, 7, 8, 9], розроблено концептуальну модель процесу управління життєвим циклом розроблення безпеко-орієнтованих сервісів, яка ґрунтується на гнучкому підході до розробки програмного забезпечення [4, 10, 14], розроблено імітаційні моделі обходу та обчислення основних параметрів мережевого графа, які в подальшому стали основою для інформаційної системи підтримки прийняття рішень в управлінні життєвим циклом розроблення спеціалізованого програмного забезпечення [5, 11, 15], розроблено алгоритми обходу та обчислення основних параметрів мережевого графа [1, 11], розроблено інформаційну систему підтримки прийняття рішень в управлінні життєвим циклом розроблення спеціалізованого програмного забезпечення [1], а також проведено апробацію моделей та інформаційної технології, оцінено їх ефективність та доцільність використання у предметній галузі [6, 12].

Апробація результатів дисертації. Основні результати наукових досліджень неодноразово доповідалися на Міжнародних та Всеукраїнських наукових конференціях, зокрема:

Міжнародна науково-практична конференція «Технологічні, технічні та стратегічні інновації в рятуванні» (Варшава, 2022 року), Всеукраїнська науково-практична конференція молодих учених, студентів і курсантів «Інформаційна безпека та інформаційні технології» (Львів, 2020 року), Міжнародна науково-практична конференція молодих вчених, курсантів та студентів «Проблеми та перспективи розвитку системи безпеки життєдіяльності» (Львів, 2021 року), Міжнародна науково-практична конференція молодих учених «Проблеми та перспективи забезпечення цивільного захисту» (Харків, 2021 року), Всеукраїнська науково-практична конференція молодих учених, студентів і курсантів «Інформаційна безпека та інформаційні технології» (Львів, 2021 року), Всеукраїнська науково-практична конференція молодих учених, студентів і курсантів «Інформаційна безпека та інформаційні технології» (Львів, 2023 року), Міжнародна науково-практична конференція «Стратегічні комунікації у сфері забезпечення національної безпеки та оборони: проблеми, досвід, перспективи» (Київ, 2023 року), Міжнародна науково-практична конференція «Проблеми та перспективи розвитку системи безпеки життєдіяльності» (Львів, 2024 року).

Публікації. За результатами дисертаційних досліджень опубліковано 15 наукових праць, з них – 1 колективна монографія, 5 наукових статей у фахових наукових виданнях, з них: 1 – у виданні, що входить до міжнародних наукометричних баз Scopus, 4 – у фахових наукових виданнях України; 9 – у матеріалах конференцій.

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, п’ятьох розділів, висновків переліку використаних джерел з 121 найменування та чотирьох додатків. Загальний обсяг дисертації становить 159 сторінок, з них основного тексту 124 сторінки, які містять 10 таблиць та 51 рисунок.

РОЗДІЛ 1. СУЧАСНИЙ СТАН ГАЛУЗІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

1.1. Аналіз предметної області наукових досліджень.

З плином часу та зростанням кількості нових інформаційних продуктів у сучасному суспільстві постає все більш актуальне питання про важливість методологій розробки програмного забезпечення. Щодня ми стикаємося з запуском нових проєктів, розробкою програмних продуктів і створенням програм, які потребують ефективного управління. На сьогоднішній день існує широкий спектр методів управління життєвим циклом розроблення програмного забезпечення. Проте основними методологіями управління є традиційні (Waterfall) та гнучкі (Agile). Вибір підходу, згідно з яким слід здійснювати розробку програмного забезпечення, залишається за командою та замовником.

Waterfall відноситься до традиційних методів управління життєвим циклом програмного забезпечення, яким притаманний чіткий та послідовний процес розробки програмного забезпечення. Попри те, що на сьогодні така модель майже не застосовується у сфері реалізації IT-проектів, вона є важливою, адже на ній базуються усі інші методології розробки програмного забезпечення.

Традиційна методологія Waterfall поділяє життєвий цикл програмного забезпечення на деякий набір фаз, кожна з яких розпочинається після завершення попередньої (рис. 1.1).

Рисунок 1.1. – Життєвий цикл моделі Waterfall [28]

Такий підхід до розробки є доволі простим та прозорим, проте водночас він занадто ідеалістичний та непрактичний в умовах динамічного оточення. Модель Waterfall зазвичай використовують у проєктах із чітко визначеними вимогами, що не змінюються протягом реалізації проєкту. Здебільшого це проєкти машинобудівної інфраструктури або державні проєкти, які вимагають чіткого документування, розрахунку бюджету та аналізу всіх можливих ризиків. Якщо брати до уваги IT-індустрію, то розробка програмного продукту за моделлю Waterfall буде доречною, якщо мова йтиме про однотипні інформаційні системи (ІС), ІС із складними обчисленнями, ІС, що працюють у реальному часі, або ІС, до яких чітко та у повній мірі можна сформулювати усі вимоги на стадії планування.

До переваг такої моделі можна віднести:

- зрозумілу та чітку послідовність розробки програмного забезпечення, що, в свою чергу, знижує поріг входження команд на проєкт;
- стабільність, оскільки на етапі затвердження вимог та документації проєкт стає незмінним протягом усього процесу розробки;
- можливість оцінити вартість та часові рамки ще до початку розробки проєкту;

- можливість легко відслідкувати процеси проєкту та матеріальні ресурси завдяки чіткій документації та суворій поетапності процесу розробки.

Як результат отримуємо вчасно розроблений в межах бюджету цілісний програмний продукт (ПП) з документацією на нього. Такий ПП володіє високою зручністю впровадження та супроводу.

Проте, як вже зазначалось, цей підхід до розробки програмного забезпечення в умовах сучасного динамічного середовища у більшості випадків є не достатньо ефективним. Найпоширенішими проблемами процесу розробки на основі традиційних методів є:

- неможливість зміни вимог безпосередньо в процесі розробки;
- відсутність чіткого розподілення обов'язків та відповідальності за виконану роботу і її результат;
- наявність безперервного потоку «дрібних» додаткових завдань, які відволікають розробників і менеджерів від основної роботи;
- невідповідність часовим рамкам, збільшення бюджету, втрата якості.

Проблема традиційного підходу до реалізації проєктів розробки програмного забезпечення полягає в тому, що виконання триває довше, ніж очікувалось, витрати виявляються більшими, ніж закладалось у бюджеті, і часто не досягають очікуваних результатів. Традиційні підходи до проєктного менеджменту беруть за основу співвідношення часу, витрат та обсягу робіт, що формують так званий «трикутник проєктного менеджменту» (рисунок 1.2), який Роб Коул та Едвард Скотчер називають «Бермудським трикутником» [65].

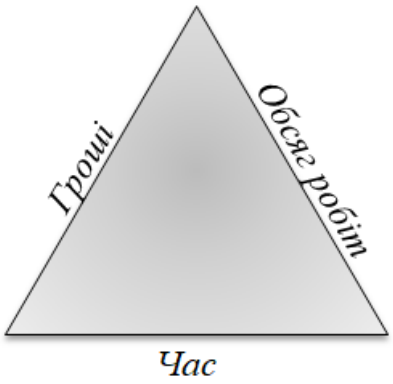


Рисунок 1.2. – Трикутник проєкт менеджменту [28]

Через жорсткі зв'язки, які покладені в основу цього трикутника, неможливо усунути одну з його сторін, не вплинувши на інші. Зміна будь-яких параметрів завжди матиме наслідки під час реалізації програмного продукту. Здебільшого це відбувається тоді, коли вносяться зміни до проєкту, скорочується час або бюджет. Проєкт-менеджерам дуже важко зберігати ці три складові в рівновазі і при цьому забезпечувати всі бажання клієнта. У результаті ми отримуємо або неякісний продукт, випущений вчасно у межах свого бюджету, або дійсно вартісний результат, ціна якого значно перевищуватиме початкову, а на часові проміжки вже не звертають увагу.

В процесі розробки продукту із використанням гнучких методологій головною є якість продукту. Тому методологія Agile відходить від традиційної одержимості термінами та бюджетами, зосереджуючись у першу чергу на тому, чого хоче клієнт, або чого він дійсно потребує [27].

Рух Agile бере початок з концепції Lean – техніки «ощадливого виробництва», котра широко застосовується в автомобільній індустрії. Спершу методологію гнучкого управління використовували в галузі інформаційних технологій (IT). Тому не дивно, що саме у лютому 2001 року сімнадцятьма фахівцями, які зібрались на гірськолижному курорті The Lodge at Snowbird у штаті Юта, щоб обговорити принципи розробки програмного забезпечення, був опублікований «Маніфест гнучкої розробки програмного забезпечення», який раз і назавжди змінив підходи

до процесу розробки та створення нових проєктів не тільки у галузі IT, а і у всіх інших сферах життєдіяльності.

«Маніфест гнучкої розробки програмного забезпечення» твердить, що [1]:

- люди та взаємодія важливіші за процеси та інструменти. Agile робить акцент на людській комунікації, командній роботі та розвитку розробників. Він говорить, що в першу чергу у команді повинен бути дух співпраці та взаємодопомоги. А дотримання жорстких правил та процесів швидше пасуватиме одинакам та диктаторам, які звикли працювати у одному стилі та не бажають змінюватись;
- працездатне програмне забезпечення важливіше за вичерпну документацію. На думку Agile-спеціалістів, не варто витрачати дорогоцінний час та гроші на написання документації. Набагато краще втілити його у розробку готового функціоналу продукту. Це допоможе замовнику набагато швидше зрозуміти цінність та перспективи розвитку проєкту;
- співпраця із клієнтом набагато важливіша за обговорення умов контракту. Гнучка методологія Agile приділяє велику увагу спілкуванню із замовником. Нерідко буває так, що клієнт сам до кінця не знає, які саме функції повинен реалізовувати продукт та який термін та бюджет потрібні для його виконання. Дуже важко наперед передбачити всі умови створення проєкту. Тому саме тісна співпраця замовника із командою допомагає випустити дійсно вартісний продукт, який задовольнятиме всі його потреби;
- готовність до змін важливіша за дотримання плану. Agile методологія вважає зміни невід'ємною частиною розвитку хорошого проєкту. Втілені в процесі розробки ідеї можна реалізувати значно швидше, а, отже, і тестувати їх можна набагато раніше.

Насправді в процесі створення проєктів важливо зрозуміти одне – замовник не хоче кращого управління. Він хоче кращого кінцевого

продукту. Методологія гнучкого управління Agile спрямована на це [27]. Не важливо, які саме техніки та процеси використовуються для досягнення кращого результату, не потрібно зосереджуватись на самих методах. Результат важливіший за шляхи, якими його досягнуто.

Розглянемо, чим відрізняється процес створення продукту на основі гнучких методологій від традиційних. Почнемо з того, що більшість гнучких методологій направлені на те, щоб мінімізувати ризики шляхом ітеративної розробки проєкту (рисунок 1.3).

Рисунок 1.3. – Життєвий цикл Agile методології [28]

Кожна ітерація – це міні-проєкт, який включає всі процеси розробки, починаючи від планування і завершуючи випуском готової функціональності. Зазвичай, вони тривають два…три тижні (все залежить від того, який темп обере команда розробників) і закінчуються ретроспективою, тобто обговоренням того, що було зроблено та що можна зробити краще.

На стадії планування проєкту Agile починає із визначення необхідного мінімуму й працює вже з ним. Цей мінімум так і називається – мінімально життєздатний продукт (minimum viable product, MVP), або мінімальний набір функціональності (minimum feature set, MFS). На практиці мінімально життєздатний продукт вже відповідає основним бізнес вимогам проєкту і може бути проданий. Такий підхід зменшує

часові та матеріальні витрати, необхідні на створення проєкту. Він може стати основою для більш складного і функціонального бізнес-рішення. До того ж його набагато краще і легше тестувати. А підсумки тестування виявляються більш продуктивними, адже ґрунтуються на реальних фактах [26].

Не менш важливим є залучення розробника на всіх стадіях розробки проєкту. В Agile прийнято у кожну команду включати людину з боку бізнесу. Таку людину зазвичай називають власником продукту. Власник продукту представляє інтереси бізнесу і кінцевого користувача. Він точно знає, що саме потрібно людям, а не як це реалізувати. Основна його функція – донести до команди завдання, які мають бути реалізовані з точки зору бізнесу, а команда, у свою чергу, реалізує це із технічного боку. Для максимального ефекту замовник і команда розробників завжди повинні працювати в парі. Тільки таким чином можна створити якісний продукт, який задовольнятиме всі бажання клієнта.

Після визначення практичного бачення проєкту команді потрібно якомога детальніше сформулювати вимоги до продукту. Такий перелік вимог у Agile називають беклогом. На відміну від звичного для традиційних моделей технічного завдання беклог формує список важливих ідей для бізнесу. В цьому підтверджується орієнтація гнучких методологій на результат, а не на процеси. Команда бачить беклог і вже після того вирішує які дії, тобто технічні рішення, їй потрібно виконати для задоволення конкретної вимоги. В свою чергу, виконання тільки технічного завдання, формування якого властиве для традиційних моделей, не завжди означає одержання бажаного продукту.

Отож, до переваг використання Agile методології можна віднести:

- мінімізація ризиків завдяки гнучкій системі внесення змін;
- робота на результат (наприкінці кожної ітерації замовник отримує готовий продукт);
- зменшення документації (за рахунок живого спілкування);

- можливість змінювати беклог проєкту в ході розробки.

Пропри все, гнучкі методології розробки програмного забезпечення мають свої недоліки. Зокрема це:

- складність підрахунку кінцевої суми та часового проміжку проєкту;
- підвищені вимоги до кваліфікації та досвіду учасників команди;
- постійні зміни, які можуть призвести до того, що проєкт ніколи не дійде до фінальної версії.

Порівняння традиційної методології (Waterfall) та гнучкої методології (Agile) наведено у таблиці 1.

Таблиця 1.1 – Порівняння традиційної методології та гнучкої методологій

Опис	**Waterfall**	**Agile**
Робота над проєктом чітко спланована та послідовно виконується	так	ні
Внесення змін до проєкту в процесі розробки	ні	так
Сталий темп процесу розробки	ні	так
Тестування ПП в процесі розробки	ні	так
Управління директивне	так	ні
Контроль за роботою команди, що працює над розробкою	так	ні
У команді можлива взаємозаміна, розподілене лідерство	ні	так
Команда співпрацює в процесі розробки із замовником	ні	так

Узагальнюючи вищенаведене, можна стверджувати, що при організації життєвого циклу програмного продукту відповідно до гнучких методологій розробниками значно швидше буде здійснено передачу замовникові прототипу ПП, проведено його тестування та адаптацію до мінливих потреб ринку та вимог замовника і, як результат, значно швидша доставка готового ПП на ринок.

1.2. Аналіз наукових досліджень в предметній області

Аналіз останніх досліджень та публікацій свідчить, що питання вибору методології управління ІТ-проєктом є актуальним як для українських, так і зарубіжних вчених. Із появою «Маніфесту розробки програмного забезпечення» [1] все більше праць присвячено Agile методології та порівнянню її із традиційними методологіями управління життєвим циклом програмного забезпечення. Зокрема, теоретичні основи Agile висвітлені у роботах Роба Коула, Едварда Скотчера, Роберта Мартіна. В Україні питанням управління життєвим циклом програмного забезпечення займались такі вчені як С. Д. Бушуєв, І. Ю. Лєбєдєва, О. Б. Зачко, І. М. Якубенко, І. І. Оберемок, Т. О. Говорущенко, О. О. Павлова та інші.

Проте, вказані праці зосереджені, в основному, на розробці програмного забезпечення у класичних проєктних командах. Якщо ж говорити про розробку критично важливих сервісів (спеціалізованого програмного забезпечення безпекового спрямування), то підхід до управління життєвим циклом такого ПП відрізнятиметься, зважаючи на умови, проєктну команду та часові терміни впровадження такого продукту у життя.

Відомі наукові праці, у яких зроблені висновки про несумісність використання гнучких методологій та розроблення спеціалізованого програмного забезпечення [64; 74]. Тим не менш новіші наукові результати поставили під сумнів ці висновки, визначивши основні переваги, що виникають в процесі впровадження гнучких методів у розробку критично важливих систем, а саме: документація, оскільки вона не є необхідною для гнучкої розробки програмного забезпечення [100]; змінні вимоги, оскільки традиційні методи цьому не сприяють [100]; життєвий цикл розробки програмного продукту, оскільки дані проєкти не розробляються ні ітераційно, ні поступово (проблема із плануванням випуску програмного

продукту) [99]; тестування, яке, у традиційній методології, проводиться тільки на завершальних стадіях розробки продукту [97].

Станом на сьогодні все частіше успішно використовують гнучкі методології управління життєвим циклом програмних систем у військовій [58, 59, 66, 99], залізничній [55], аерокосмічній [67, 110], медичній [101, 97] та інших сферах діяльності. Зокрема, у роботі [58] проведено дослідження на основі розробки системи командування та управління для Генерального штабу армії Італії із використанням гнучких методологій управління. Після тринадцяти п’ятитижневих спринтів розробники змогли надати готовий продукт, який відповідав усім функціональним та нормативним вимогам армії. Попри витрати, спрямовані на навчання та пристосування до нового методу розробки програмного забезпечення, витрати на саму розробку стали нижчими, ніж були раніше при використанні традиційних методів управління життєвим циклом програмних систем. У роботі [67] представлено результати реорганізації команд розробників у Scrum-команди в Agile Transformation Бразильського аеронавтичного обчислювального центру. Зокрема був побудований симулятор польоту та розроблена система, яка контролює політ іноземних літаків у повітряному просторі Бразилії. Автори у роботі [97] описали традиційний підхід та виклики, з якими зустрічаються команди розробників в процесі створення програмного забезпечення для медичних установ, а також успішно впроваджені гнучкі практики до розробки в даній галузі. А у джерелі [73] наводяться рекомендації щодо відповідності програмного забезпечення міжнародним стандартам та документам регуляторних органів та при використанні гнучких практик agile у розробці програмного забезпечення для медичних установ.

Автори у роботі [63] запропонували метод, заснований на розширенні підходу DevOps до безпекових систем. Даний автоматизований підхід можна легко зіставити з відомим методом гнучкої методології Scrum [62, 107]. У роботі [55] авторами також розширено цей

метод, який надалі отримав назву Scrum for Safety (S4S). Він спрямований допомогти науково-дослідницьким групам розробляти безпекові рішення для залізничної галузі. У статті [110] йдеться про використання гнучкого підходу для забезпечення безпеки критично важливого вбудованого програмного забезпечення для космічного корабля NASA Orion. Йдеться, зокрема, про виклики, пов'язані з забезпеченням критичних функцій програмної системи та необхідність використання адаптивного підходу до забезпечення якості та валідації цих функцій у контексті еволюції методів розробки.

Автори у роботі [54] запропонували новий метод, що підтримує гнучку методологію розробки програмного забезпечення. Він базується на концепції MVC і складається із трьох основних частин: модель, вигляд та контролер. Ідея полягає у тому, що замовник самостійно проектує користувацький інтерфейс. Це зменшує ризик відхилення або обмеженого використання програмного забезпечення і сприяє більш простій та безпомилковій розробці програмного забезпечення.

З огляду на проведений аналіз існує потреба у створенні нового гнучкого підходу до розробки спеціалізованого програмного забезпечення, котрий буде адаптований під специфіку роботи Державної служби України із надзвичайних ситуацій та корелюватиме із принципами гнучкої методології управління ІТ-проєктом. Для цього варто детально проаналізувати процес розробки програмного забезпечення на кожному етапі [79].

Станом на сьогодні існує велика кількість публікацій, присвячених процесу розробки програмних систем. Зокрема, у роботі [3] запропоновано залучати когнітивні технології на рівні експертних систем оцінки часу розробки, прийняття рішень щодо вибору архітектури, побудови програмних застосунків тощо. В науковій праці [109] автори розглянули процес розробки програмного забезпечення, який складається із чотирьох фаз, де архітектура програмного забезпечення є найголовнішою, оскільки

забезпечує абстрактне представлення загальної структури програмної системи. У роботі [31] розроблено модель системи управління якістю процесу розробки програмного забезпечення на основі процесного підходу PDCA. У дослідженні [19] автори розробили методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення. У статті [105] використано теорію обмежень у контексті процесу розробки програмного забезпечення. В дослідженнях [28, 55, 56, 84, 88, 80, 114], основна увага приділена вибору методології для управління процесами розробки програмного забезпечення. В наукових роботах [21, 40, 94] описано процес створення сервісів, орієнтованих на забезпечення якості підготовки команд рятувальників у динамічному оточенні. Проте в згаданих роботах не зазначено, які саме моделі управління життєвим циклом розроблення програмних систем були використані при їх реалізації. Аналіз та математичне моделювання окремих етапів життєвого циклу спеціалізованого програмного забезпечення дозволили автоматизувати розробку безпеко-орієнтованих сервісів та удосконалити процес управління життєвим циклом такого програмного забезпечення.

Зважаючи на основну мету роботи, варто також загадати про наукові праці, в яких автори більш детально досліджували етап планування в управлінні життєвим циклом розробки програмного забезпечення. Зокрема, у роботі [57] доведено, що планування займає центральне місце у Agile проектах, а всі інші сфери обертаються навколо нього. У роботі [60] також доведено важливість процесу планування та експертної оцінки завдань під час покер планування, а у науковій праці [61] вказано на важливість використання «дошки завдань» в процесі планування. В науковому джерелі [68] розглянуті обов’язкові кроки, які слід зробити, щоб отримати максимальну віддачу від гнучкої розробки програмного забезпечення. В [69] запропоновано використовувати інструмент графічного моделювання для прийняття рішень у міждисциплінарному дослідницькому співробітництві.

Оскільки у даній роботі запропоновано використовувати мережеві графи на етапі планування життєвого циклу спеціалізованого програмного забезпечення, варто також розглянути їх застосування у сучасній науці.

Теорія графів – спеціальний розділ математики, який вирішує велику кількість наукових та прикладних завдань. Зокрема, розв'язання задач із кібернетики, програмування, штучного інтелекту, логістики. Із використанням графів можна швидко та якісно дослідити різні електричні, транспортні або комп'ютерні мережі, системи електронної комунікації, водопостачання або електрифікації. Чіткий та зрозумілий механізм побудови графів дозволяє з легкістю змоделювати та інтерпретувати потрібні процеси (події чи явища), реалізувати їх наочне представлення без необхідності фізичного проектування.

Станом на сьогоднішні існує велика кількість різноманітних алгоритмів та методів обходу графів. Більшість із них спрямовані на визначення найкоротшого шляху. Це, зокрема, алгоритми Дейкстри, Флойда-Уоршела, Беллмана-Форда. У роботах [10, 33, 48, 104] описано основні сфери застосування даних алгоритмів. Здебільшого це логістика, комп'ютерні мережі, системи електронної комунікації та інші. Завдання цих алгоритмів полягає у знаходженні оптимального шляху між заданими вершинами. Існує також ряд алгоритмів, спрямованих на знаходження пропускної здатності мережі. Це, зокрема, алгоритми Форда-Фалкерсона, Едмондса-Карпа, Дініца. У роботах [32, 37, 69] описане практичне застосування даних алгоритмів. Основною їх задачею є знаходження такого потоку, щоб його величина була максимальна. Здебільшого вони використовуються для оцінки завантаження потокової чи транспортної мережі.

Математичний апарат теорії графів широко використовується у галузі штучного інтелекту. Існує окремий клас методів глибинного навчання (графові нейронні мережі), призначений для формулювання логічних висновків на основі даних, описаних цими графами [118].

Бібліотека машинного навчання TensorFlow використовує графи потоків даних для здійснення чисельних обчислень. Вузли графів представляють собою математичні операції, а грані – багатовимірні масиви даних (“тензори”), що “протікають” між вузлами [7]. У статті [13] запропоновано використовувати векторний скелет (граф) для побудови алгоритму розпізнання символів для систем штучного інтелекту.

Теорія графів широко застосовується у криптографії та стеганографії. У роботі [52] представлено новий стеганографічний підхід до приховування даних у растровому зображенні. Запропонований підхід розподіляє зображення на словник пар «ключ-значення» у стилі JSON і використовує метадані графа для визначення розташування різних частин і позицій зображення корисної інформації у всьому кластері покритого зображення. Теорія графів активно використовується і у маркетинговій справі, зокрема, для побудови ефективної реклами. Якщо перші пошукові сервіси використовували ключові слова для пошуку інформації, таким чином створювали ієрархію сторінок за кількістю переглядів, то пошукова система Google повністю змінила свою концепцію, цим самим стала однією із монополістів на ринку пошукових систем. Дана система для надання рангу сторінці використовує сімейство алгоритмів RankPage. Алгоритм заснований на методі графів виявлення ступеня важливості сторінки через кількість цитувань [6] (до речі, підґрунтям для цього став алгоритм для визначення науковометричних рейтингів журналів).

Також відома низка робіт [75, 81, 82], присвячених використанню мережевих графів у процесі планування проекту. Зокрема, авторами у роботі [75] обрано метод розв’язання задачі планування, заснований на застосовуванні методу мережевого планування. Він базується на ідеї оптимізації критичного шляху із залученням додаткових обмежених коштів. Даний підхід не корелює із розробкою спеціалізованого програмного забезпечення. У статті [81] описано процес розробки комп’ютерної програми розв’язання задач мережевої оптимізації. Проте

розрахунок найкоротших шляхів проходить за алгоритмом Дейкстри, який не є універсальним і не адаптований для мережевого планування процесу розробки програмного забезпечення. В основу дослідження [82] закладено методи мережевого планування PERT, засоби елементів теорії графів та методу діаграм Ганта. Підхід актуальний для каскадної моделі управління, якому притаманне послідовне виконання завдань та визначений час виконання.

1.3. Розробка безпеко-орієнтованих сервісів у форматі студентських R&D проєктів

Реалізація науково-дослідних робіт із розробки безпеко-орієнтованих сервісів є розповсюдженою практикою у Львівському державному університеті безпеки життєдіяльності. Наукові дослідження дозволяють студентам та аспірантам здобути не тільки нові знання, але й підготувати здобувачів освіти до практичної діяльності, сформувати базові принципи наукового підходу до розв’язання практичних завдань тощо [45]. В свою чергу під менторством досвідчених викладачів здійснюється розробка важливих сервісів для служби порятунку. Останнім часом широкого розповсюдження набуває один із різновидів дослідження – реалізація проєктів у форматі R&D (Research and Development). Цей різновид роботи відрізняється від класичного наукового дослідження обов’язковою реалізацією отриманого наукового результату у форматі інформаційної, програмної або комп’ютерної (робототехнічної) системи.

В наукових колах зустрічається низка досліджень та публікацій, що присвячені організаційним питанням науково-дослідної діяльності здобувачів освіти Зокрема, в роботах [12, 16, 42] розглядається різновид такої діяльності як підґрунтя до студентського розвитку. Наукові праці [20, 23] висвітлюють питання студенто-центрованого навчання та дослідження комплексно в освітньому середовищі закладу освіти. Поряд із вагомим

внеском в означених та інших наукових працях не розглядається питання менеджменту науково-дослідної роботи. Саме тому існує потреба у дослідженні оптимальних методів управління виконанням науково-дослідної діяльності у форматі реалізації R&D проєктів.

Зважаючи на те, що R&D проєкти, як і будь які інші різновиди проєктної діяльності, обмежені ресурсами (часовими, людськими, матеріальними), а результатом їх виконання є унікальний продукт (послуга), то реалізацію рекомендовано супроводжувати відповідно до певної методології управління проєктами. На сьогоднішні найбільшої популярності набули дві моделі управління саме ІТ-проєктами: гнучка та каскадна моделі реалізації ІТ-проєктів.

Як вже було зазначено, основною характеристикою каскадної моделі є фіксований обсяг робіт, який не змінюється від початку до кінця їх виконання. Реалізація безпеко-орієнтованого сервісу за такою моделлю передбачає виконання чітко визначених та послідовно структурованих етапів. Основний обсяг робіт за проєктом передбачає застосування визначених на його початку технологій, мов програмування та фреймворків. Пересвідчитись в доцільності вірного вибору обраної технології можливо тільки після завершення проекту. Додаткові ризики можуть бути пов’язані з тим, що здобувачі освіти (які здебільшого займаються розробкою таких БОС у ЛДУ БЖД) на початковому етапі можуть до кінця не володіти необхідною технологією (мовою програмування тощо) для реалізації проєкту, а для її освоєння потрібен додатковий час або залучення до команди досвідчених учасників. Подібний хід реалізації проєкту стимулюватиме до збільшення загального часу його реалізації, що не корелює з вимогами до каскадної моделі управління.

Для реалізації R&D роботи безпеко-орієнтованого напряму за каскадною методологією усі вимоги до проєкту та його продукту мають бути чітко сформульовані ще на стадії планування. А так як дані проєкти є

свого роду навчальними зі значною частиною експериментальних спроб, а їх учасниками є здобувачі освіти, які тільки вдосконалюють свою професійну майстерність, то стає очевидно, що визначення чітких вимог на стадії планування проєкту є завданням близьким до неможливого.

Зважаючи на зазначене, реалізацію R&D проєктів слід організовувати за гнучкою методологією, де здобувачі освіти мають обмежений час і фіксовану команду, проте можуть обирати довільну технологію (якою володіють), змінювати вимоги та адаптувати проєкт (продукт) під цільову аудиторію в ході реалізації проєкту та виконання окремих його частин.

В якості гнучкої (Agile) методології управління R&D проєктами обрано Scrum-модель. Вона передбачає поділ реалізації усього проєкту на спринти (поділ загального обсягу робіт за проєктом на окремі підзадачі). Очевидно, що здобувачам освіти значно легше працювати з меншими обсягами завдань та отримувати результат своєї роботи окремими працездатними частинами. Тривалість одного спринта складає від двох до чотирьох тижнів залежно від складності взятого на спринт обсягу робіт. Складність робіт також визначає сама команда проєкту. Додаткова перевага такого підходу полягає у формуванні в здобувачів освіти уяви про організацію справжніх бізнес-процесів в середовищі ІТ-компанії. Таким чином, випускник освітньої програми буде адаптованим до командної роботи та процесів проєктного менеджменту ІТ-компаній.

На початку реалізації проєкту роль проєктного менеджера, бізнес-аналітика та архітектора виконують ментори з подальшою передачею цих ролей учасникам команди. Здобувачі освіти формують основу команди. Це розробники ІТ-рішень, тестери та DevOps-інженери. За scrum-моделлю один із учасників команди виконує функції scrum-майстра (лідера команди), та один – роль Product Owner. Учасники команди можуть мінятись ролями після декількох ітерацій (спринтів). Для контролю за ходом виконання проєкту після визначеного обсягу спринтів команда

представляє мінімально життєздатний продукт (Minimum viable product), який оцінюють ментори (науково-педагогічні працівники) із внесенням коректив та побажань.

Взаємовідносини між учасниками scrum-команди подамо за допомогою кругів Ейлера (рисунок 1.4).

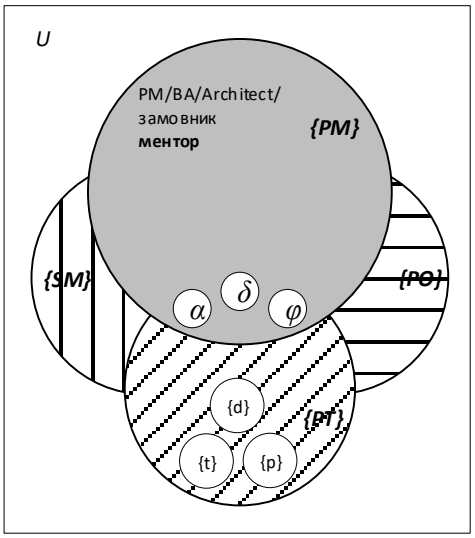


Рисунок 1.4. – Взаємовідношення множини учасників проєктної команди із розробки безпеко-орієнтованих сервісів

Освітнє середовище, в якому реалізуються такі R&D проєкти, представлено у вигляді універсуму <*U*>, елементами якого є підмножини, що імітують учасників проєктної команди. До згаданих підмножин відносять підмножину scrum-майстра (*SM*). Основне завдання здобувача освіти, який виконує роль scrum-майстра, пролягає в організації роботи команди, стимулюванні до розвитку шляхом спільного вивчення нових технологій. Він повинен брати на себе ініціативу щодо вирішення складних завдань тощо. Наступна підмножина Product Owner *(PO)* включає у себе учасника, що відповідає за наповнення (формування) загального переліку робіт за проєктом, а також переліку робіт на визначений спринт. Здобувач освіти, який виконує роль Product Owner, тримає постійний зв'язок з ментором проєкту, який водночас може виступати як замовник продукту, та регулює разом з ним означений перелік робіт (загальний список та

список на спринт). Підмножина учасників команди *(PT)* включає у себе підмножини: розробників *{d}*, тестувальників *{t}* та DevOps-інженерів *{p}*. Розглянемо множину учасників проєктної команди *(PT)* детальніше:

$$PT = \{d_i, i = \overline{1,x}; t_j, j = \overline{1,e}; p_k = k = \overline{1,s}\}, \quad (1.1)$$

де x – чисельність розробників в проєктній команді;

e – чисельність тестувальників в проєктній команді;

s – чисельність DevOps інженерів в проєктій команді.

Як видно з моделі, приведеної на рисунку 1.4, описаному середовищу притаманні операції перетину, які є комутативними:

$$U \ni \bigcap_{i=1}^{4}(SM, PO, PT, PM). \quad (1.2)$$

Розглянемо операції об'єднання (перетину) множин в контексті спільної реалізації проєкту його учасниками (взаємовідношення між учасниками проєктної команди):

$$PM \cap SM \cap PT = \{\alpha: \alpha \in PM; \alpha \in SM; \alpha \in PT\}, \quad (1.3)$$

$$PM \cap PO \cap PT = \{\varphi: \varphi \in PM; \varphi \in PO; \varphi \in PT\}, \quad (1.4)$$

$$PM \cap SM \cap PO \cap PT = \{\delta: \delta \in PM; \delta \in SM; \delta \in PO; \varphi \in PT\}, \quad (1.5)$$

де α – елементи проєктного середовища, орієнтовані на організацію роботи команди R&D-проєкту;

φ – елементи проєктного середовища, орієнтовані на раціональний вибір обсягу проєктних робіт та етапів їх виконання;

δ – множина елементів проєктного середовища, орієнтована на досягнення основної цілі проєкту (його реалізації).

На завершення слід зазначити, що виконання науково-дослідних проєктів із розробки безпеко-орієнтованих сервісів в рамках здобуття

вищої освіти в жодному разі не може замінити традиційних підходів до навчання, але може яскраво їх доповнювати із максимальним наближення освітнього процесу до реальних практичних кейсів.

Висновки до першого розділу

Зважаючи на особливості розробки програмного забезпечення у динамічному оточенні та отримані результати аналізу, зроблено висновки про неспроможність традиційних методологій (Waterfall) забезпечити належний рівень ефективності на різних етапах життєвого циклу розроблення спеціалізованого програмного забезпечення. Натомість окреслено основні переваги застосування методології Agile у розробці програмного забезпечення та її адаптивність до постійних змін у вимогах, чисельності команди розробки, бюджету тощо. А все тому, що гнучкість – це основна перевага розробника на ринку. Лише ті команди, які можуть іти в ногу з часом, які працюють на результат та на задоволення будь-яких потреб замовника, будуть мати місце на сучасному ринку праці. Саме такі команди і пропагує Agile методологія.

У результаті проведеного аналізу використання Agile методів в процесі розробки критично важливих сервісів обґрунтовано як гнучкі методи можуть допомогти у інноваціях спеціалізованого безпеко-орієнтованого програмного забезпечення, систематизовано основні концепції методів гнучкої методології управління життєвим циклом програмних систем, розширено існуючі емпіричні дані про можливість та переваги використання гнучких методів у безпековій галузі.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ЖИТТЄВОГО ЦИКЛУ РОЗРОБКИ СПЕЦІАЛІЗОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ДИНАМІЧНОМУ ОТОЧЕННІ

2.1. Математична модель процесу розробки спеціалізованих програмних систем безпеко-орієнтованого спрямування

Відомі дослідження в царині менеджменту життєвого циклу програмних систем використовують теоретичні відомості про етапи розробки нового програмного продукту і не вивчають можливостей їх автоматизації. Саме тому існує потреба у розробці математичної моделі, яка описуватиме процес розробки спеціалізованих програмних систем і надасть змістовні підстави для провадження подальших досліджень щодо оптимізації ресурсної складової проєктів шляхом забезпечення якості продукту за умови зниження часу та матеріальних витрат на розробку, а також підвищення ефективності процесів управління життєвим циклом розроблення безпеко-орієнтованих програмних систем. Для побудови математичної моделі використано понятійний апарат теорії множин, а також визначено основні взаємозв’язки окремих етапів розробки означених програмних систем.

Побудову геометричної моделі життєвого циклу спеціалізованих програмних систем проведемо у наступному порядку. На рисунку 2.1 поданий універсум <U> з елементами у вигляді множини, які імітують етапи розроблення програмного забезпечення.

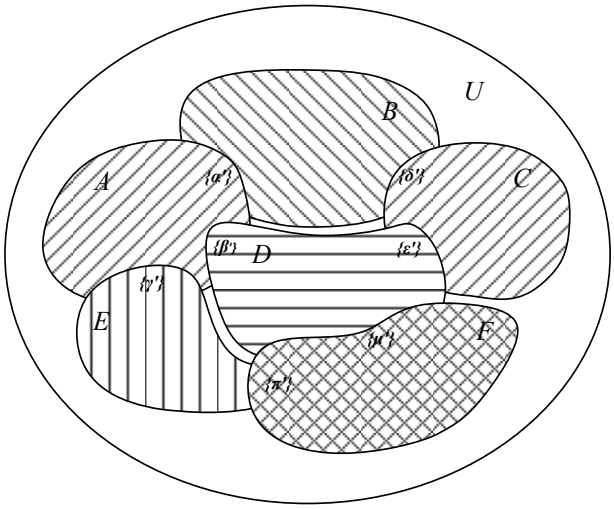


Рисунок 2.1 – Геометрична модель життєвого циклу спеціалізованих програмних систем у вигляді універсуму <U>

Вміст універсуму можна описати так:

$$U = \{A, B, C, D, E, F\}, \tag{2.1.}$$

де $\{A\}$ – множина робіт з аналізу предметної області та формулювання вимог;

$\{B\}$ – множина робіт з проєктування архітектури програми та UX-дизайну;

$\{C\}$ – множина робіт з реалізації програмної системи в кодах;

$\{D\}$ – множина робіт з тестування програмної системи;

$\{E\}$ – множина робіт з впровадження програмної системи;

$\{F\}$ – множина робіт з супроводу програмної системи в процесі експлуатації.

В універсумі $<U>$ існують елементи, які належать перетину множин:

$$A \cap B; A \cap D; A \cap E; B \cap C; C \cap D; D \cap F; E \cap F. \tag{2.2}$$

У результаті виконання робіт універсуму <U> отримуємо готовий програмний продукт. Позначимо $R(X)$ – результат виконання множини X, зокрема:

$$\{\alpha'\} = R(A \cap B), \tag{2.3}$$

де α' – визначені вимоги до програмного забезпечення на основі аналізу предметної області і формулювання системних вимог;

$$\{\beta'\} = R(A \cap D), \tag{2.4}$$

де β' – вимоги до тестування програмного забезпечення, які визначаються на етапі аналізу предметної області та формулювання системних вимог (перевірка відповідності цим вимогам);

$$\{\gamma'\} = R(A \cap E), \tag{2.5}$$

де γ' – вимоги до впровадження програми, які визначаються на етапі аналізу предметної області і формулювання системних вимог (до прикладу вибір платформи для розгортання продукту, визначення етапів розгортання продукту на боці замовника, визначення етапів навчання користувачів тощо);

$$\{\delta'\} = R(B \cap C), \tag{2.6}$$

де δ' – програмний код, що реалізує функціонал програмної системи відповідно до специфікації та поданої архітектури;

$$\{\varepsilon'\} = R(C \cap D), \tag{2.7}$$

де ε' – результати тестування програмного коду на відповідність специфікації та результати опрацювання журналу помилок (допомагає виявляти проблеми в роботі програмної системи та їх природу);

$$\{\mu'\} = R(D \cap F), \tag{2.8}$$

де μ' – технічні звіти за результатами тестування, які містять інформацію про виявлені проблеми, шляхи їх вирішення та рекомендації щодо тестування і виправлення недоліків в процесі експлуатації програмної системи;

$$\{\pi'\} = R(E \cap F), \tag{2.9}$$

де π' – вимоги до процесів підтримки програмного забезпечення.

Для глибшого аналізу процесу розроблення програмного забезпечення розглянемо кожну із множин універсуму <*U*>. На рисунку 2.2 зображена модель множини *A*, яка відтворює етап аналізу предметної області і формулювання системних вимог (постановки завдання) до програмного продукту.

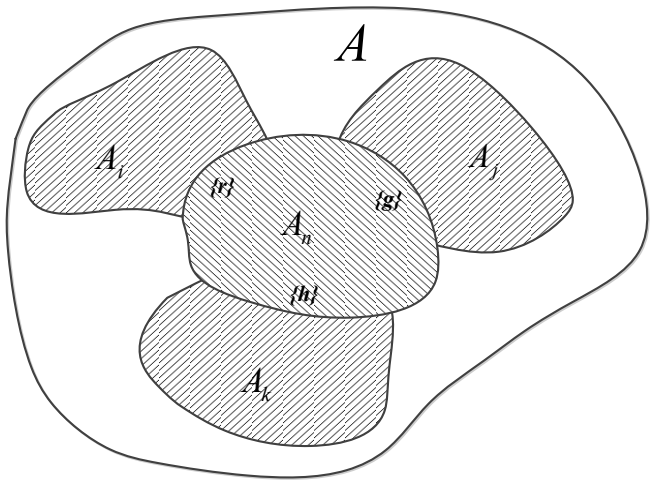


Рисунок 2.2 – Множина робіт з аналізу предметної області і формулювання вимог

Множина A у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$A = \{A_i, A_j, A_k, A_n\}, \tag{2.10}$$

де A_i – множина робіт із оцінки вимог та потреб стейкхолдерів;

A_j – множина робіт з визначення обмежень та вимог до функціональності;

A_k – множина робіт з визначення фінансових та ресурсних обмежень проєкту;

A_n – множина робіт із створення документації проєкту.

В множині A між підмножинами існують елементи, які належать їх перетину:

$$A_i \cap A_n; A_j \cap A_n; A_k \cap A_n. \tag{2.11}$$

У результаті виконання робіт множини A отримуємо сформульований аналіз предметної області і системних вимог до програмного продукту. Виконання робіт, які належать перетину підмножин множини A, призводять до отримання нових результатів :

$$\{r\} = R(A_i \cap A_n), \tag{2.12}$$

де r – сформоване технічне завдання проєкту;

$$\{g\} = R(A_j \cap A_n), \tag{2.13}$$

де g – сформована специфікація програмного забезпечення;

$$\{h\} = R(A_k \cap A_n), \tag{2.14}$$

де h – сформований фінансовий та ресурсний план проєкту.

Наступний етап життєвого циклу програмного забезпечення – проєктування архітектури програми та UX-дизайну інтерфейсу. На рисунку 2.3 зображена графічна модель цього етапу у вигляді множини *B*.

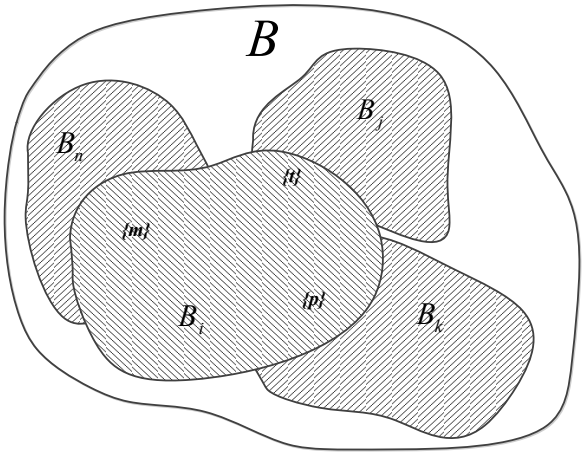


Рисунок 2.3 – Множина робіт з проєктування архітектури програми та UX-дизайну

Множина *B* у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$B = \{B_i, B_j, B_k, B_n\}, \tag{2.15}$$

де B_i – множина робіт із створення архітектури системи;

B_j – множина робіт із вибору програмних технологій;

B_k – множина робіт із побудови та написання алгоритмів;

B_n – множина робіт з розробки дизайну та моделі людино-машинної взаємодії.

У множині *B* між підмножинами існують елементи, які належать до їх перетину:

$$B_i \cap B_j; B_n \cap B_i; B_k \cap B_i. \quad (2.16)$$

У результаті виконання робіт множини *B* отримуємо спроєктовану архітектуру програми та UX-дизайн інтерфейсу. Виконання робіт, які належать перетину підмножин множини *B,* призводять до отримання нових результатів :

$$\{t\} = R(B_j \cap B_i), \quad (2.17)$$

де t – архітектура структурних елементів програмної системи у вигляді діаграми класів та діаграми об’єктів;

$$\{m\} = R(B_n \cap B_i), \quad (2.18)$$

де m – архітектура структурних елементів програмної системи у вигляді діаграми прецедентів, станів та компонентів (прототип програми);

$$\{p\} = R(B_k \cap B_i), \quad (2.19)$$

де p – архітектура структурних елементів програмної системи у вигляді діаграм взаємодії та розміщення.

Наступний етап життєвого циклу програмного забезпечення – реалізація програмного продукту в кодах (імплементація програмної системи). На рисунку 2.4 зображена геометрична модель даного етапу у вигляді множини *C*.

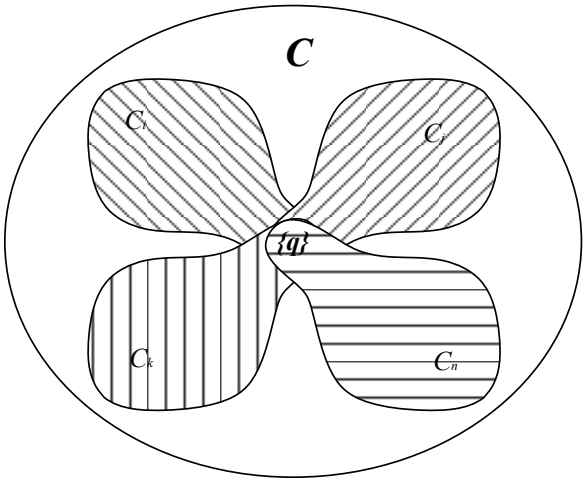


Рисунок 2.4 – Множина робіт з реалізації програмної системи в кодах

Множина *C* у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$C = \{C_i, C_j, C_k, C_n\}, \tag{2.20}$$

де C_i – множина робіт з реалізації програмної системи в кодах;

C_j – множина робіт з оптимізації та тестування коду (Unit tests);

C_k – множина робіт з виявлення та усунення дефектів, що виникають в процесі тестування (Unit Tests);

C_n – множина робіт із розробки документації до коду програми.

В множині С між підмножинами існують елементи, які належать їх перетину:

$$C_i \cap C_j; C_i \cap C_k; C_i \cap C_n; C_j \cap C_k; C_j \cap C_n; C_n \cap C_k. \tag{2.21}$$

У результаті виконання робіт множини *C* отримуємо реалізацію програмної системи в кодах. Виконання робіт, які належать перетину підмножин множини *C,* призводять до одержання нового результату :

$$\{q\} = R(C_i \cap C_j \cap C_k \cap C_n), \tag{2.22}$$

де q – готовий програмний модуль із супровідною документацією.

Черговий етап життєвого циклу програмного забезпечення – тестування програмного продукту (QC/QA). На рисунку 2.5 зображена геометрична модель цього етапу у вигляді множини *D*.

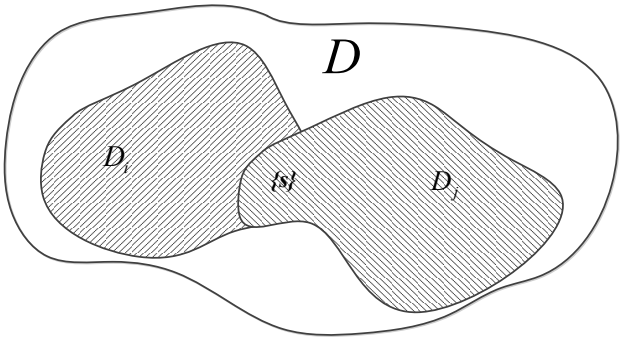


Рисунок 2.5 – Множина робіт із тестування програмної системи

Множина *D* у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$D = \{D_i, D_j\}, \tag{2.23}$$

де D_i – множина робіт із виконання тестів на функціональність, продуктивність, безпеку тощо;

D_j – множина робіт із перевірки відповідності вимогам технічного завдання.

В множині *D* між підмножинами існують елементи, які належать їх перетину:

$$D_i \cap D_j. \tag{2.24}$$

У результаті виконання робіт множини *D* отримуємо протестовану програмну систему. Виконання робіт, які належать перетину підмножин множини *D,* призводять до отримання нового результату:

$$\{s\} = R(D_i \cap D_j), \tag{2.25}$$

де s – пройдені тест-кейси (test cases) або/та звіти про помилки (bug reports).

По завершенні етапу тестування життєвий цикл програмного забезпечення переходить до етапу впровадження програмного продукту шляхом його розгортання на боці клієнта (видача замовнику). На рисунку 2.6 відтворена геометрична модель цього етапу у вигляді множини *E*.

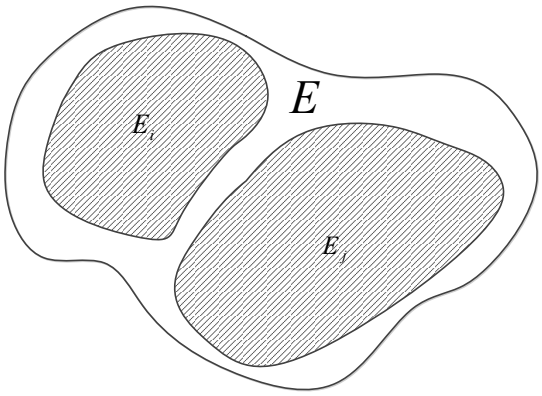


Рисунок 2.6 – Множина робіт із впровадження програмної системи

Множина *E* у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$E = \{E_i, E_j\}, \tag{2.26}$$

де E_i – множина робіт з підготовки програмного середовища для розгортання програмного продукту;

E_j – множина робіт із навчання користувачів.

У результаті виконання робіт множини E отримуємо впроваджену програмну систему на боці клієнта. Виконання робіт, які належать перетину підмножин множини E, призводять до отримання наступного результату:

$$R(E_i \cap E_j) = \varnothing. \tag{2.27}$$

Нарешті останній етап життєвого циклу програмного забезпечення – супровід програми в процесі експлуатації (підтримка програмного забезпечення) до моменту завершення експлуатації продукту. На рисунку 2.7 зображена геометрична модель цього етапу у вигляді множини F.

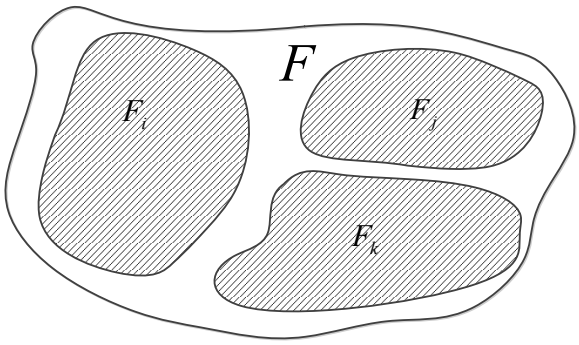


Рисунок 2.7 – Множина робіт із супроводу програмної системи під час експлуатації.

Множина F у відтвореній моделі містить підмножини та може бути представлена у вигляді:

$$F = \{F_i, F_j, F_k\}, \tag{2.28}$$

де F_i – множина робіт із виправлення помилок та виявлення нових дефектів;

F_j – множина робіт із надання користувачам необхідної допомоги та підтримка;

F_k – множина робіт із доповнення програмного продукту новим функціоналом, що концептуально відтворює усі попередньо описані етапи ЖЦ розроблення програмних систем.

У результаті виконання робіт множини F отримуємо супровід програмної системи під час експлуатації. Виконання робіт, які належать перетину підмножин множини F призводять до отримання наступних результатів:

$$\begin{aligned} R(F_i \cap F_j) &= \varnothing; \\ R(F_i \cap F_k) &= \varnothing; \\ R(F_j \cap F_k) &= \varnothing. \end{aligned} \tag{2.29}$$

Приведені геометричні та математичні моделі множин із використанням понятійного апарату теорії множин допомагають узагальнити структуру та описати усі етапи життєвого циклу програмного забезпечення на різних етапах (аналіз предметної області та формулювання системних вимог, проектування архітектури програми та UX-дизайну інтерфейсу, реалізація програми в кодах, тестування програми, впровадження програми, супровід програми під час експлуатації).

Математична модель дає підстави сформувати повну та чітку уяву про обсяги робіт, виконання яких покладене в основу успішної реалізації програмних систем, у тому числі безпеко-орієнтованого спрямування. Отримана модель дозволяє охарактеризувати окремі етапи розробки продукту та аналітично описати життєвий цикл розроблення програмного забезпечення, що є передумовою для пошуку шляхів автоматизації окремих процесів підтримки прийняття управлінських рішень.

2.2. Концептуальна модель управління життєвим циклом спеціалізованого програмного забезпечення безпеко-орієнтованого спрямування

Проведений літературний аналіз вказує на потребу створення нових методів управління життєвим циклом розроблення спеціалізованого програмного забезпечення (безпеко-орієнтованих сервісів), котрі адаптовані під специфіку роботи Державної служби України із надзвичайних ситуацій та корелюють із принципами гнучкої методології управління ІТ-проєктом. У даному підрозділі обґрунтовано як гнучкість може допомогти у інноваціях спеціалізованого безпеко-орієнтованого програмного забезпечення, щоб підвищити ефективність та надійність таких програмних продуктів, об’єднано основні концепції методів гнучкої методології управління життєвим циклом програмного забезпечення, зважаючи на специфіку розробки такого ПЗ для служби порятунку (зокрема основний акцент здійснено на гнучкий scrum метод) та розроблено концептуальну модель процесу управління життєвим циклом розроблення безпеко-орієнтованих сервісів, яка ґрунтується на гнучкому підході.

Відомо, що кожна ітерація (спринт) процесу розробки програмного забезпечення за найпопулярнішою на сьогоднішній день гнучкою моделлю Scrum складається з наступних етапів: створення беклогу продукту, початок спринта та визначення mvp, створення беклогу спринта, пріоритизація користувацьких історій (завдань із беклогу) та їх оцінка, подрібнення користувацьких історій та їх оцінка, розробка продукту, демо та ретроспектива [107]. Проте означені процеси адаптовані та апробовані під використання в традиційних (класичних) проєктних командах, що працюють за гнучкими моделями розробки програмного забезпечення. Це команди розробників, що налічують від п’яти до дев’яти осіб із розподілом обов’язків за ролями (позиціями). Більшість часу спринта вони

використовують на вирішення (розв’язання) поставлених завдань. Динамічність подібного процесу розробки полягає тільки у можливій зміні обсягу або змісту робіт. Чисельність команди та час на виконання фіксований. Проте змінність оточення процесів розробки програмного забезпечення, що реалізується за Agile-методологією, може полягати не тільки в можливій зміні визначеного обсягу подій, а й у динаміці визначеного часового ресурсу.

До прикладу, як у воєнний час, що запроваджено на території України, так і у період функціонування оперативних служб у повсякденному режимі, окремі ІТ-підрозділи Державної служби України із надзвичайних ситуацій займаються розробкою програмного забезпечення, вбудованих систем та інших безпеко-орієнтованих сервісів [84], що орієнтовані на підвищення ефективності функціонування оперативно-рятувальних служб.

Членами команд розробки таких БОС в оперативних формуваннях є кадрові працівники відповідних служб, які в міру своєї службової підготовки повинні поєднувати діяльність, пов’язану із розробкою означених сервісів, з іншими різновидами оперативної та/або службової діяльності. Тому, зважаючи на специфіку роботи і особливих учасників проєктних команд із розробки безпеко-орієнтованих сервісів, динаміка проєктного середовища набуває дещо іншого значення.

Динамічність тепер зосереджується не лише на обсягу робіт, а й у часі їх реалізації. З першого погляду можна припустити можливість розробки БОС в подібних умовах за каскадною моделлю. Проте таке припущення буде хибним, адже більшість БОС не мають чіткого терміну необхідних робіт, а функціонал зазнає динамічних змін та поправок у ході розробки.

За означених умов розроблення безпеко-орієнтованих сервісів проєктними командами оперативних формувань здійснюється за допомогою наступної моделі (рисунок 2.8).

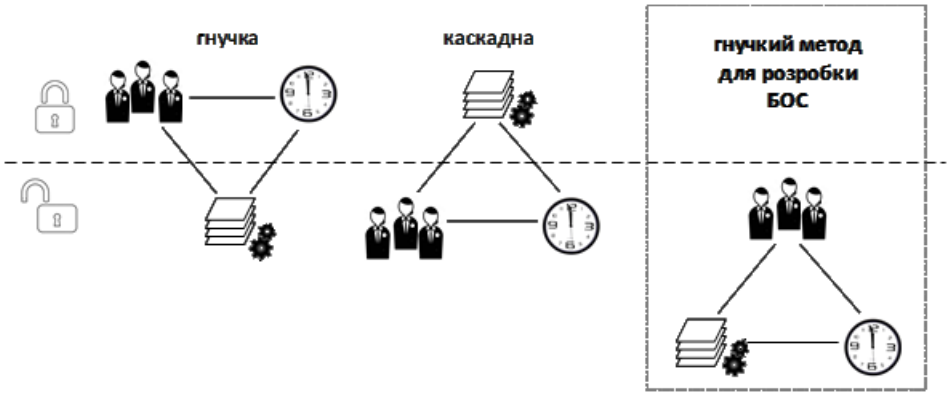


Рисунок 2.8 – Порівняння гнучкої та каскадної моделей розроблення програмного забезпечення з фактичною моделлю реалізації БОС[5]

Бачимо, що динамічні процеси з регламентованим переліком робіт, а також відсутність контролю (обмеження) часу на їх виконання може значно ускладнювати процеси своєчасного виконання роботи. Саме тому виникає потреба у побудові нового підходу для розробки БОС, що відповідатиме парадигмі гнучкого управління процесом розроблення програмного забезпечення в оперативних формуваннях.

Для розв’язання цієї потреби використаний математичний апарат теорії графів, адаптований до процесів мережевого планування за рахунок оптимізації розрахунку його основних часових параметрів під динамічні умови проєкту. Запропонований нами підхід дозволить у реальному часі здійснювати перепланування завдань із спринта, враховуючи критично важливі задачі для створення мінімально-життєздатного продукту (mvp) програмної системи.

Тоді додаткове завдання для команди розробників після пріоретизаціїї та оцінки користувацьких історій у беклозі спринта полягає у визначенні попередніх користувацьких історій чи функцій, які мають бути виконані, оскільки існують такі задачі, виконання яких можливе тільки після завершення попередніх. У такому випадку процес розроблення безпеко-орієнтованих сервісів набуде наступного вигляду (рисунок 2.9).

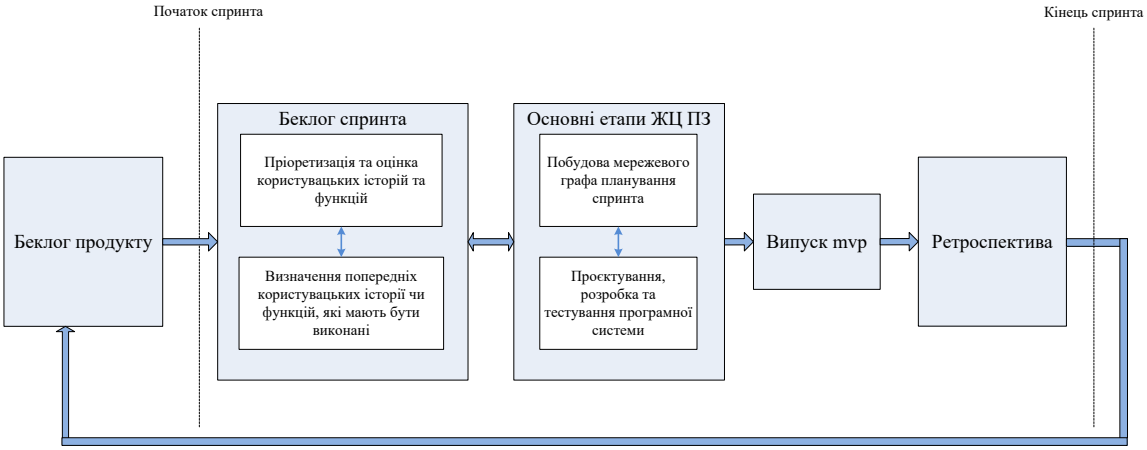


Рисунок 2.9 – Розроблена концептуальна модель процесу управління життєвим циклом БОС

Відповідно до розробленої нами концептуальної моделі управління БОС починається із формування беклогу продукту. У даному випадку беклог містить всі завдання (користувацькі історії) та функції, які необхідно виконати для розробки усієї програмної системи. Після формування беклогу весь процес розробки, як і в звичній Scrum команді, розбивається на ітерації (спринти), кожна з яких повинна завершуватись випуском готового функціоналу (mvp) та ретроспективою.

Зважаючи на специфіку розроблення безпеко-орієнтованих сервісів, беклог спринта формується наступним чином:

1. Пріоретизуються та оцінюються користувацькі історії та функції, які необхідно виконати за спринт.

2. Визначаються попередні користувацькі історії чи функції, які мають бути виконані, оскільки існують такі задачі, виконання яких можливе тільки після завершення попередніх.

Між цими пунктами існує двосторонній зв'язок, оскільки при зміні пріоритету користувацької історії може змінюватись також і попередня користувацька історія.

Після формування беклогу спринта відбувається розроблення самої системи. Для автоматизації етапу планування запропоновано побудувати мережевий граф планування спринта та обчислити його основні показники

(критичний шлях, ранні та пізні терміни виконання подій, резерви на виконання завдань). Такий підхід дозволить проєктній команді відслідковувати в режимі реального часу стан розробки системи та, за потреби, змінювати хід виконання завдань. Мережеве планування надасть можливість ефективно розподіляти час на розробку та автоматизувати процес визначення критично важливих функцій для випуску mvp.

Після планування реалізуються звичні етапи життєвого циклу розроблення програмного забезпечення такі як проектування, розробка та тестування програмної системи. На концептуальній схемі між плануванням та наступними етапами ЖЦ ПЗ показаний двосторонній зв’язок, оскільки залежно від процесу розробки, складності завдань, досвіду проєктної команди та інших чинників час на реалізацію може змінюватись, тому може виникнути потреба у перепланування мережевого графа. Двосторонній зв’язок існує також і між формуванням беклогу спринта та основними етапами ЖЦ ПЗ, оскільки залежно від визначеного критичного шляху під час мережевого планування, завдання та користувацькі історії беклогу спринта можуть змінюватись і навпаки.

Завершується спринт випуском готового продукту, який може бути використаний у службовій діяльності працівниками ДСНС та ретроспективою, на якій обговорюються шляхи його покращення та вносяться корективи на наступну ітерацію.

У наступних розділах дисертації описані розроблені моделі, алгоритми та інформаційна технологія оцінки тривалості розробки БОС із використанням розробленої нами концептуальної моделі.

Висновки до другого розділу

За результатами розділу можна зробити такі висновки.

1. Моделюванням процесу розробки програмних систем із використанням понятійного апарату теорії множин отримано математичну

модель їх життєвого циклу, яка дає змогу охарактеризувати обсяги робіт на окремих етапах створення програмного продукту, встановити взаємозв'язки і взаємозалежності між ними та сформулювати фундаментальні принципи автоматизації процедури підтримки прийняття управлінських рішень на різних етапах створення програмних систем, зокрема, безпеко-орієнтованого спрямування;

2. Розроблено концептуальну модель процесу управління життєвим циклом розроблення спеціалізованого програмного забезпечення (безпеко-орієнтованих сервісів), котра адаптована під специфіку роботи Державної служби України із надзвичайних ситуацій та корелює із принципами гнучкої методології управління ІТ-проєктів.

Одержані наукові результати використані в процесі розроблення інформаційної підтримки прийняття рішень щодо управління життєвим циклом розроблення програмних систем безпеко-орієнтованого спрямування.

РОЗДІЛ 3. НАУКОВІ РЕЗУЛЬТАТИ ТА ІМПЛЕМЕНТАЦІЯ МЕТОДІВ УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ РОЗРОБЛЕННЯ СПЕЦІАЛІЗОВАНИХ ПРОГРАМНИХ СИСТЕМ В ДИНАМІЧНОМУ ОТОЧЕННІ

3.1. Розробка імітаційної моделі процесу обходу мережевого графа для інформаційної системи підтримки рішень управління життєвим циклом спеціалізованих програмних систем

В працях [9, 10, 12] означено, що мережеві графи дають змогу відобразити роботи проєкту та зв’язки між ними. Такі графи зазвичай використовують в управлінні проєктами для довгострокового планування та оцінки стану виконання проєкту на поточному етапі. Найпопулярнішим алгоритмом мережевого планування є метод PERT (Project Evaluation and Review Technique). Ця мережа передбачає оцінку кожної роботи із використанням трьох тимчасових станів: оптимістичний, песимістичний та найбільш ймовірний час. Як відомо, мережевий граф дає змогу розрахувати ключові показники для знаходження критичного шляху виконання робіт, а також резерви часу для кожної роботи окремо. Цей метод добре зарекомендував себе в управлінні масштабними проєктами, проте вивчення ефективності його застосування для короткострокового планування розробки програмного забезпечення у динамічних умовах досліджено не достатньо. Крім того, попередні наукові досягнення вказують на складність використання стандартних підходів проєктного менеджменту (гнучкі та каскадні) до управління процесом розроблення спеціалізованого програмного забезпечення у динамічному оточенні. Зважаючи на зазначене, в розділі поставлено мету моделювання процесів обходу мережевого графа для розрахунку ключових параметрів мережевої моделі, що дасть змогу швидко та якісно оцінити тривалість проєктних робіт в умовах динамічності ресурсів та змісту робіт. Моделювання дискретних процесів мережевого планування допоможе сформувати

повноцінну уяву щодо застосування PERT-підходів для задач короткострокового планування проєктів з розроблення безпеко-орієнтованих сервісів, які володіють значним показником динамічності. Створення моделі є визначальним етапом у побудові алгоритмів обходу мережевого графа для оцінки тривалості виконання проєктних робіт. Своєю чергою розроблені алгоритми дадуть змогу надалі автоматизувати процеси короткострокового планування, а також швидко і якісно проводити переоцінку тривалості робіт за умови введення будь яких змін до змісту, обсягу, часу виконання окремих спринтів проєкту або складу команди розробників.

Для кращого розуміння об’єкта моделювання, процесу обходу мережевого графа для визначення його ключових параметрів, варто розглянути основні елементи мережевої моделі в контексті гнучкої методології управління IT-проєктами. Оскільки в мережевому плануванні основними елементами моделі є роботи та події, то в контексті Agile-технології: робота – це користувацька історія або функція; подія – початок або завершення будь якої роботи (користувацької історії). Події у мережевій моделі позначаються вершинами графа, а роботи – його ребрами. За одиницю вимірювання робіт в процесі моделювання буде використано універсальну відносну величину story points (далі – s.p.), яка враховує поєднання трудомісткості завдання, його складності та ризиків, які з ним пов'язані. Такий різновид оцінки дає змогу враховувати часові та ресурсні обмеження процесу розроблення спеціалізованого програмного забезпечення залежно від досвіду команди розробників.

Для виконання процедури мережевого планування (побудови мережевого графа) необхідно [12]:

1. Пріоретизувати користувацькі історії та функції, які необхідно виконати.
2. Оцінити користувацькі історії та функції у s.p.

3. Визначити попередні користувацькі історії чи функції, які мають бути виконані, оскільки існують такі завдання, виконання яких можливе лише після завершення попередніх.

Наявність мережевого графа дає можливість сформувати уяву про орієнтовні обсяги робіт та їх черговість. Проте наявність лише самого графа не дає змоги оцінити тривалість робіт за проєктом. Побудова графа мережевої моделі в початковому стані не дасть змоги проводити оперативний перерозподіл обсягів робіт із подальшим визначенням тривалості їх виконання в динамічному оточенні. З цією метою математичний апарат мережевого планування володіє низкою методів розрахунку параметрів мережевого графа для часової оцінки процесів розробки специфічного програмного забезпечення на стадії планування, а саме: ранній термін виконання подій T_f; пізній термін виконання подій T_l; критичний шлях виконання робіт; часові резерви $R_{\text{подій}}$, $R_{\text{робіт}}$. В математичному відношенні визначення цих параметрів не передбачає жодної складності. Проте в алгоритмічному представленні, з метою подальшої автоматизації розрахунків, можуть виникати труднощі, пов’язані з коректністю роботи алгоритму, його безвідмовністю тощо. Це може бути зумовлене складністю самої мережевої моделі, наявністю великої кількості робіт, у тому числі фіктивних, наявністю великої кількості подій, наявністю або відсутністю взаємозв’язків між ними, розгалуженістю мережевої моделі тощо.

Зважаючи на це, процедуру обходу мережевого графа для розрахунку його ключових параметрів необхідно представити у вигляді уніфікованої імітаційної моделі. Універсальність моделі полягатиме у її спроможності відтворення процедури обходу будь якого мережевого графа незалежно від його складності та конфігурації. І, найголовніше, уніфікована модель створить передумови для побудови відповідних алгоритмів та автоматизації процесів розрахунку параметрів мережевого графа.

Зважаючи на те, що мережевий граф та процедура його обходу є дискретною системою, для побудови імітаційної моделі процесів визначення параметрів T_f, T_l обрано математичний апарат моделювання розподілених дискретних систем мережами Петрі. Мережі Петрі обрано з точки зору зручності імітаційного відтворення паралельних (розподілених) процесів, а також їх динамічної взаємодії в системі. Саме такими характеристиками володіють мережеві моделі, які досліджуються в попередніх роботах [9, 10, 12].

В дискретній моделі, описаній за допомогою мережі Петрі, події відображаються переходами t, умовами виконання яких є позиції P. На першому етапі реалізується визначення раннього терміну виконання подій. Цей параметр характеризує термін, раніше якого подія відбутись не може, та дає змогу контролювати початок виконання робіт, які не регламентовані порядковістю. Для першої події $T_{f(1)} = 0$, для всіх подальших подій мережевої моделі Tf визначається відповідно до методики.

Отже, відтворимо та опишемо модель процесу обходу мережевого графа з метою визначення раннього часу виконання робіт T_f (рисунок 3.1).

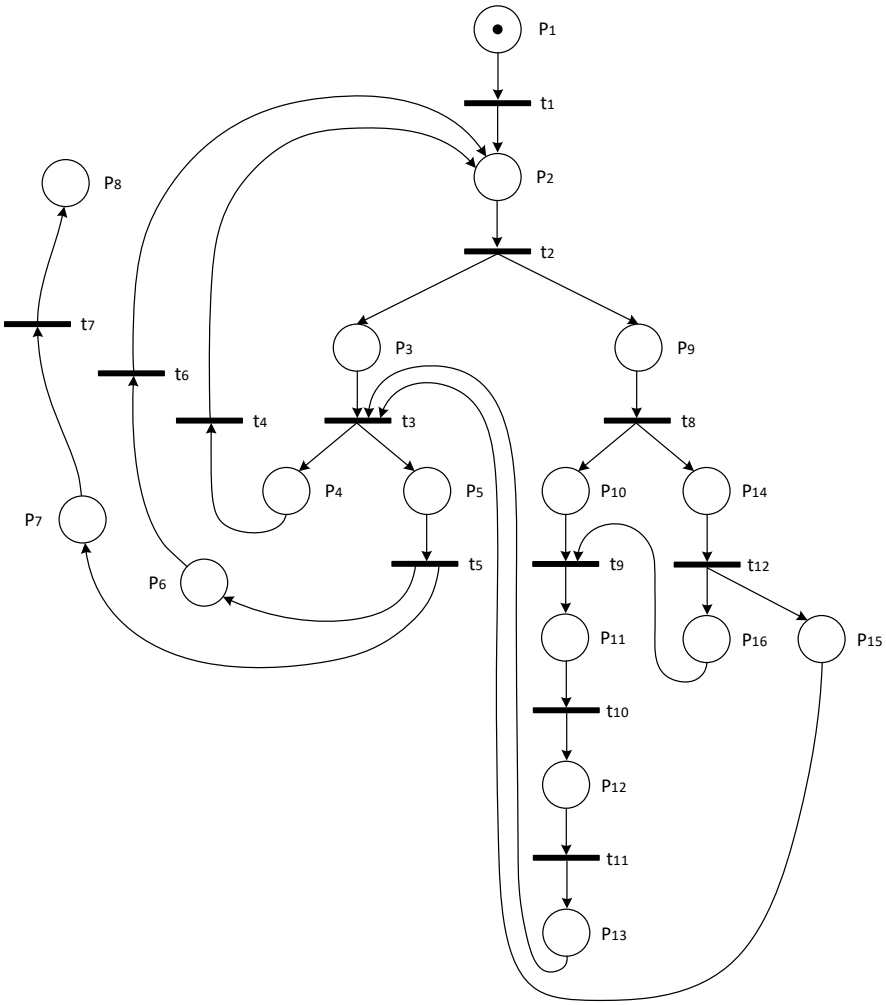


Рисунок 3.1 – Імітаційна модель процесу обходу мережевого графа з метою визначення раннього часу виконання робіт T_f

Процес обходу мережевого графа для розрахунку раннього терміну виконання подій починається з позиції (умови) P_1, яка відповідає за перевірку наявності початкових даних, а саме: індексу попередньої події $i = 0$; індексу наступної події $j = 1$; обсягу подій мережевої моделі n; множини робіт мережевої моделі $\{t_{i-j}\}$. Перехід (подія) t_1 відповідає за підготовку початкових параметрів циклічного обходу мережевого графа з такими значеннями: ранній термін виконання першої події $T_{f(1)} = 0$; i++; $j = i+1$. Позиція P_2 перевіряє відповідність параметрів циклу на черговій ітерації (в ході імплементації моделі буде реалізовано алгоритмічно). Власне новаційність поданої моделі обходу мережевого графа полягає у циклічності встановлення зв’язків між подіями та виконанні розрахунку

раннього терміну виконання подій в межах поточної ітерації. Перехід t_2 реалізує процедуру входження у цикл та визначення поточного ребра (роботи) $i - j$. Позиція P_3 перевіряє умову чи ребро $i - j$ не належить множині $\{t_{i-j}\}$ (тобто мережевим графіком робота $i - j$ не передбачена): $i - j \notin \{t_{i-j}\}$. Перехід t_3 відповідає за визначення поточного значення події j, адже на кожній ітерації циклу значення індексу наступної події j буде змінюватись. В позиції P_4 проводиться перевірка, чи значення j менше або однакове з кількістю подій мережевої моделі n $(j \leq n)$. Якщо нерівність $j \leq n$ виконується, здійснюється перехід t_4, де значення індексу j збільшується на 1 $(j++)$. Після чого процедура обходу мережевого графа повертається у позицію P_2. Позиція P_5 перевіряє зворотну умову, коли значення j більше за кількість подій мережевої моделі n. За умови виконання нерівності $j > n$, здійснюється перехід t_5, де індекс i збільшується на 1 $(i++)$. В позиції P_6 здійснюється перевірка не рівності індексу попередньої події і та кількості подій мережевої моделі n: $i \neq n$ (не досягнуто граничного значення циклу). Якщо не рівність виконується – спрацьовує перехід t_6, який відповідає за встановлення нового значення індексу $j = i+1$. Після чого здійснюється циклічне повернення до перевірки умови P_2. Позиція P_7 перевіряє рівність індексу попередньої події i та кількості подій мережевої моделі n. Якщо значення індексу i досягає n (досягнуто граничного значення циклу), то виконується перехід t_7, із виводом результатів розрахунку та подальшим завершенням обходу мережевого графа у позиції P_8.

Повернемось до розгалуження моделі після виконання переходу t_2 (вхід у цикл та перевірка поточного значення i - j). В позиції P_9 відбувається перевірка умови, чи поточна робота i - j належить множині $\{ti-j\}$ (вхідними даними мережевої моделі передбачена робота $i - j$). У разі наявності відповідної роботи у заданій множині робіт $i - j \in \{t_{i-j}\}$ виконується перехід t_8, завданням якого є визначення раннього терміну виконання події на поточному етапі обходу мережевого графа для події j за залежністю:

$$T_{f(j)} = T_{f(i)} + t_{(i \to j)}, \tag{3.1}$$

де $T_{f(i)}$ – ранній термін виконання попередньої події; $t_{(i \to j)}$ – обсяг роботи, яка передує події j (враховує складність та обсяги робіт у Story Point).

Далі у позиції P_{10} здійснюється перевірка, чи значення розрахованого раннього терміну $T_{f(j)}$ на поточній ітерації циклу не міститься у множині ранніх термінів виконання подій: $T_{f(j)} \notin \{T_{fj}\}$. Якщо поточне значення раннього терміну $T_{f(j)}$ не міститься у множині, це означає, що значення отримано вперше. Після чого в переході t_9 відбувається визначення попередньої події T_i для поточної події T_j: $T_{prev(j)} = T_i$, (фіксується номер попередньої події для T_j). В позиції P_{11} відбувається перевірка факту запису (наявності) $T_{prev(j)}$ до поточної події T_j (факт запису = true). В переході t_{10} відбувається запис зв'язку поточна подія T_j – попередня подія $T_{prev(j)}$, до *HashMap* $<T_j, T_{prev(j)}>$. Позиція P_{12} перевіряє факт запису зв'язку T_j – $T_{prev(j)}$ до *HashMap* $< T_j, T_{prev(j)}>$ (факт запису = true). Визначене унікальне значення раннього терміну виконання подій $T_{f(j)}$ на поточній ітерації циклу записується до множини ранніх термінів *HashSet* $\{T_{f(j)}\}$ в події t_{11}. В позиції P_{13} перевіряється чи значення $T_{f(j)}$ успішно записано до множини *HashSet* $\{T_{f(j)}\}$ (факт запису = true). За виконання умови P_{13} виконується перехід до t_3 та початок нової ітерації циклу.

Позиція P_{14} перевіряє наявність визначеного у переході t_8 значення раннього терміну виконання подій $T_{f(j)}$ у множині ранніх термінів $\{T_{fj}\}$ після чергової ітерації розрахунків (наявність $T_{f(j)}$ у множині $\{T_{fj}\}$ можлива, якщо події T_j передують декілька подій T_i, а на попередніх ітераціях циклу відбувалось визначення показника їх раннього терміну виконання). У випадку виконання умови P_{14} $T_{f(j)} \in \{T_{fj}\}$ виконується перехід t_{12}, де визначається максимальне значення серед терміну $T_{f(j)}$ поточної ітерації та попередньо записаного значення $T_{f(j)}$ до множини $\{T_{fj}\}$ (за результатами попередніх ітерацій). На цьому етапі необхідно передбачити можливість коректного розрахунку часу раннього початку для тих подій, яким

передують декілька робіт з врахуванням вимоги $T_{f(j)} \rightarrow \max$, використовуючи вираз:

$$T_{f(j)} = \max \begin{cases} T_{f(k)} + t_{(k \rightarrow j)} \\ \cdots\cdots\cdots \\ T_{f(m)} + t_{(m \rightarrow j)} \end{cases} \quad (3.2)$$

де *k, m* – це індекси подій, що передують події *j*.

У позиції P_{15} відбувається перевірка нерівності $\{T_{f(j)}\} \geq T_{f(j)}$, де встановлюється чи значення раннього терміну виконання подій, яке записано до множини $\{T_{f(j)}\}$ на попередніх ітераціях більше або рівне поточному розрахованому значенню раннього терміну $T_{f(j)}$. Якщо нерівність виконується, то перезапису збереженого у множині значення поточним не відбувається (записане значення більше за поточно розраховане), а система переходить на нову ітерацію циклу t_3. В позиції P_{16} відбувається зворотна перевірка нерівності $\{T_{f(j)}\} < T_{f(j)}$. Якщо поточне розраховане значення раннього терміну $T_{f(j)}$ більше за раніше записане значення до множини $\{T_{f(j)}\}$, запускається процедура перезапису більшого значення до $HashSet\{T_{f(j)}\}$ шляхом виконання переходу t_9 із подальшим проходження черговості описаних етапів.

Представлена на рисунку 3.1 імітаційна модель обходу мережевого графа дала змогу описати процедуру визначення раннього терміну виконання подій $T_{f(j)}$ із урахуванням усіх обмежень у вигляді алгоритму (додаток А, рисунок 2). Імітаційне моделювання процесу обходу мережевого графа дозволило розробити унікальний алгоритм та адаптувати його під методологію гнучкого управління проєктними роботами. Термін раннього виконання подій для останньої події мережевої моделі *n* є терміном виконання усього комплексу робіт спринта.

Далі проведено імітаційне моделювання процесу обходу мережевого графа з метою визначення пізнього терміну виконання робіт T_l. В основу

цієї моделі також закладено принцип циклічного перебору зв'язків $i-j$, проте відмінністю є зворотній порядок обходу (від останньої до першої події). Визначення пізнього терміну виконання подій $T_{l(i)}$ реалізується з метою контролю термінів, до яких події мережевої моделі мають бути повністю виконані задля уникнення випадків збільшення часу на реалізацію усього проєкту. Розглянемо модель цього процесу у вигляді мережі Петрі на рисунку 3.2.

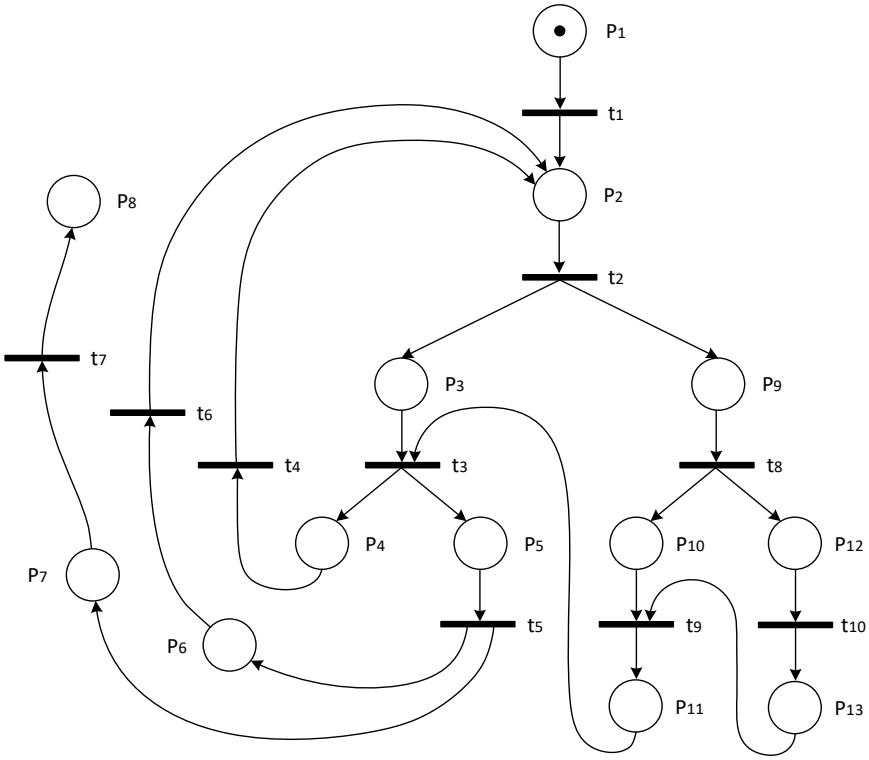


Рисунок 3.2 – Імітаційна модель процесу обходу мережевого графа з метою визначення пізнього часу виконання робіт T_l

Описаний імітаційною моделлю процес розрахунку пізнього терміну виконання події починається із позиції P_1, яка перевіряє введення даних: індексу попередньої події $i=0;$ індексу поточної події $j = n;$ обсягу подій мережевої моделі $n;$ множини робіт мережевої моделі $\{t_{i-j}\}$; множини ранніх термінів виконання подій $\{T_{fj}\}$ (розраховано за результатами обходу графа за попередньою моделлю). За умови виконання P_1 здійснюється перехід t_1 – підготовка параметрів циклічного обходу графа, де пізній час виконання для останньої події рівний ранньому часу виконання цієї події

$T_{l(n)}=T_{f(n)}$, а значення індексу $i = j–1$. Позиція P_2 перевіряє відповідність параметрів циклу на черговій ітерації. Перехід t_2 реалізує процедуру входження у цикл та визначення поточного ребра (роботи) $i – j$. Циклічна процедура обходу мережевого графа організована за прикладом попередньої моделі, відмінністю є зворотній шлях обходу графа (з кінця до початку). Позиція P_3 перевіряє умову чи ребро $i – j$ не належить множині $\{t_{i-j}\}$: $i – j \notin \{t_{i-j}\}$. Перехід t_3 відповідає за визначення поточного значення попередньої події i, адже на кожній ітерації циклу значення індексу попередньої події i буде зменшуватись. В позиції P_4 проводиться перевірка, чи значення i більше або рівне за 1 ($i \geq 1$). Якщо нерівність $i \geq 1$ виконується, здійснюється перехід t_4, де значення індексу i зменшується на 1 (i—). Після чого процедура обходу мережевого графа повертається у позицію P_2. Позиція P_5 перевіряє зворотну умову, коли значення $i < 1$. За умови виконання нерівності $i < 1$, здійснюється перехід t_5, де індекс j зменшується на 1 (j—). В позиції P_6 здійснюється перевірка чи значення індексу поточної події j не досягло 1: $j \neq 1$ (не досягнуто граничного значення циклу). Якщо нерівність виконується, спрацьовує перехід t_6, який відповідає за встановлення нового значення індексу $i = j–1$. Після цього здійснюється циклічне повернення до перевірки умови в позиції P_2. Позиція P_7 перевіряє рівність індексу поточної події $j = 1$. Якщо значення індексу j досягає 1 (досягнуто граничного значення циклу), то виконується перехід t_7, із виводом результатів розрахунку та подальшим завершенням обходу мережевого графу у позиції P_8.

Перейдемо до розгалуження моделі після виконання переходу t_2 (вхід у цикл та перевірка поточного значення $i – j$). В позиції P_9 відбувається перевірка умови, чи поточна робота $i – j$ належить множині $\{t_{i-j}\}$. У разі наявності відповідної роботи у заданій множині робіт $i – j \in \{t_{i-j}\}$ виконується перехід t_8, завданням якого є визначення пізнього терміну з виразу:

$$T_{l(i)} = T_{l(j)} - t_{(i \to j)}, \tag{3.3}$$

де $T_{l(j)}$ – пізній термін виконання наступної (поточної) події; $t_{(i \to j)}$ – обсяг роботи, яка виконується після події i та перед подією j (враховує обмеження у Story Point).

Далі у позиції P_{10} здійснюється перевірка, чи значення розрахованого пізнього терміну $T_{l(i)}$ на поточній ітерації циклу не міститься у множині пізніх термінів виконання подій: $T_{l(i)} \notin \{T_{l(i)}\}$. Якщо поточне значення пізнього терміну $T_{l(i)}$ не міститься у множині, це означає, що розраховане значення отримано вперше. Після чого в переході t_9 відбувається запис значення пізнього терміну виконання цієї події $T_{l(i)}$ до множини $HashSet\{T_{li}\}$. В позиції P_{11} відбувається перевірка факту запису (наявності) $T_{l(i)}$ до $HashSet\{T_{li}\}$ (факт запису = true), після чого відбувається перехід на нову ітерацію циклу в переході t_3. Позиція P_{12} перевіряє наявність визначеного у переході t_8 значення пізнього терміну виконання подій $T_{l(i)}$ у множині пізніх термінів $\{T_{l(i)}\}$ після чергової ітерації розрахунків. У випадку виконання умови $T_{l(i)} \in \{T_{l(i)}\}$ виконується перехід t_{10}, де визначається мінімальне значення серед терміну $T_{l(i)}$ поточної ітерації та попередньо записаного значення $T_{l(i)}$ до множини $\{T_{l(i)}\}$, використовуючи вираз:

$$T_{l(i)} = \min \begin{cases} T_{l(k)} - t_{(i \to k)} \\ \cdots\cdots\cdots \\ T_{l(m)} - t_{(i \to m)} \end{cases} \tag{3.4}$$

де k, m – це індекси похідних подій, від події i.

В позиції P_{13} відбувається перевірка факту визначення мінімального значення $T_{l(i)}$ на поточній ітерації циклу, після чого виконується його перезапис до множини $HashSet\{T_{li}\}$ шляхом повернення до переходу t_9.

Пізній термін T_l для останньої події мережевої моделі рівний часу її раннього виконання події T_f :

$$T_{l(n\ ост.)} = T_{f(n\ ост.)}. \tag{3.5}$$

Значення T_l для першої події мережевої моделі за умови вірного виконання попередніх розрахунків буде дорівнювати часу раннього її виконання T_f:

$$T_{l(n\ поч.)} = T_{f(n\ поч.)}. \tag{3.6}$$

Для усіх подій мережевої моделі, крім першої та останньої, має виконуватись нерівність:

$$T_{l(n)} \geq T_{f(n)}. \tag{3.7}$$

Представлена на рисунку 3.2 імітаційна модель обходу мережевого графа дала змогу описати процедуру ітераційного визначення пізнього терміну виконання подій $T_{l(i)}$ із урахуванням усіх обмежень у вигляді алгоритму (додаток А, рисунок 3).

Реалізовані імітаційні моделі та побудовані на їх основі алгоритми обходу мережевого графа для розрахунку раннього та пізнього термінів виконання подій дають змогу оцінити максимальну тривалість робіт з розроблення спеціалізованого програмного забезпечення зважаючи на наявні ресурси, які необхідні для успішного виконання спринта. Отримані алгоритми на основі імітаційних моделей надають можливість автоматизувати визначення раннього та пізнього термінів виконання робіт, а відтак реалізувати процедуру оперативного перепланування тривалості проєктних робіт в динамічному оточенні (в умовах постійних змін).

Унікальність імітаційних моделей обходу мережевого графа полягає у циклічності процедури встановлення зв'язку між попередньою та поточною подією *i – j*. При обході мережевого графа для визначення раннього терміну за умови фіксованого значення індексу попередньої події *i* здійснюється ітераційне збільшення індексу поточної події *j* до моменту, коли значення *j* буде більше за кількість подій мережевої моделі *n*. На цьому етапі циклічний перебір завершується, індекс попередньої події *i* збільшується на одиницю та реалізується процедура повторного входу у цикл із значенням $j = i+1$. Процедура повторного виконання циклу для обходу мережевого графу виконується до моменту коли значення індексу попередньої події і досягне значення кількості подій мережевої моделі *n*. При обході мережевого графа для визначення пізнього терміну, процедура має аналогічний підхід, проте у зворотньому порядку (від останньої до першої події). За цих умов значення індексу *j* на початковому етапі рівний *n*, а індекс *i* ітераційно зменшується на 1. Застосування цього підходу дозволить почергово перебирати усі ребра (роботи) мережевої моделі із подальшим визначенням необхідних показників.

3.2. Розробка алгоритмів обходу мережевого графа для інформаційної системи підтримки прийняття рішень управління життєвим циклом розроблення спеціалізованих програмних систем

Першим етапом для визначення обсягів проєктних робіт необхідно побудувати мережевий граф. Для цього нумеруємо початкову подію j=1 , попередня подія $(i = 0)$ – відсутня. Від неї проводимо вектор, який позначатиме першу користувацьку історію чи функцію та її вагу у s.p.. Якщо є декілька робіт, які можуть виконуватись паралельно, будується необхідна кількість векторів. Оскільки кожна користувацька історія має завершуватись кінцевою подією, позначаєм подію на графіку і збільшуємо її порядковий номер на $j = i + 1$ (робимо так для всіх паралельних робіт,

при цьому *j* збільшуємо на 1). Якщо потрібно побудувати користувацьку історію, виконання якої можливе тільки після попередньої, шукаємо на графіку кінцеву подію роботи, яка нас цікавить та від неї проводимо вектор потрібної користувацької історії. Якщо ж виконання користувацької історії залежить від завершених декількох попередніх, цю задачу вирішуємо із використанням фіктивної роботи. Від кінця кожної попередньої користувацької історії проводимо вектор, який означає фіктивну роботу вага якої 0 s.p. та завершуємо їх у одній кінцевій події, яка служитиме початком потрібної нам нової користувацької історії. Таким чином будуємо графік мережевого планування для тої кількості користувацьких історій та функцій, які нас цікавлять (в межах обсягу спринта). Розроблений нами алгоритм побудови мережевого графа (мережевої моделі) визначеного обсягу робіт на спринт приведений на рисунку 3.3.

Initialization:
$prevTask[k][p]$ - array of previous works;
$pair < i, j >$, where $i = 1$, $j = 2$;
n- works, $n = 1$;
empty map $MapTask < n, pair < i, j >>$
for $(k = 0;\ k < prevTask.length;\ k++)$ **do**
  for $(p = 0;\ p < prevTask[k].length;\ p++)$ **do**
    if $(pverTask[k][p] == 0)$ **then**
      Add n and $pair < i, j >$ to $MapTask < n, pair < i, j >>$;
      $j++$;
      $n++$;
    end
  end
end
$i++$;
for $(k = 0; k < prevTask.length; k++)$ **do**
  for $(p = 0; p < prevTask[k].length; p++)$ **do**
    if $(prevTask.length == 1)$ **then**
      for $(Map.Entry < n, pair < i, j >> works =$
       $MapTasks.entrySet())$ **do**
        if $(prevTask[k][p] == works.getKey())$ **then**
          for $(Map.Entry < i, j > pairSet : pair.entrySet())$
           do
            $i = paieSet.getValue()$;
            Add n and $pair < i, j >$ to
             $MapTask < n, pair < i, j >>$;
            $j++$;
            $n++$;
          end
        else
          continue;
        end
      end
    else
      $j++$;
      for $(Map.Entry < n, pair < i, j >> works =$
       $MapTasks.entrySet())$ **do**
        if $(prevTask[k][p] == works.getKey())$ **then**
          for $(Map.Entry < i, j > pairSet : pair.entrySet())$
           do
            $i = pairSet.getValue()$;
            Add fict. work 0 and $pair < i, j >$ to
             $MapTask < n, pair < i, j >>$;
            $j++$;
          end
        else
          continue;
        end
      end
      $i = j$;
      $j++$;
      Add n and $pair < i, j >$ to $MapTask < n, pair < i, j >>$;
      $n++$;
    end
  end
end
Return $MapTask < n, pair < i, j >>$.

Рисунок 3.3 – Алгоритм для побудови мережевого графа

Слід нагадати, що основними елементами мережевої моделі є її події та роботи, які оцінюються у Story Point. Такий різновид оцінки дозволяє враховувати часові та ресурсні обмеження процесів розробки специфічного програмного забезпечення.

До основних параметрів мережевої моделі для часової оцінки процесів розробки спеціалізованого програмного забезпечення на стадії

планування, віднесено: ранній термін виконання подій T_f, пізній термін виконання подій T_l, резерви виконання подій $R_{подій}$, резерви виконання робіт $R_{робіт}$. Алгоритм визначення означених параметрів, який закладено в роботу інформаційної системи підтримки прийняття рішень, розглянемо далі.

На першому етапі реалізується визначення раннього терміну виконання подій. На рисунку 3.4 зображений алгоритм визначення раннього часу виконання події, враховуючи розроблену імітаційну модель на рисунку 3.1

Initialization:
i - index of the previous event, $i = 0$;
j - index of the next event, $j = 1$;
n - the volume of network model events;
$\{t_{i \to j}\}$- set of network model works;
$\forall j = 1$ then $T_{e(1)} = 0$;
$i++$;
for $(j = i + 1;\ j \le n;\ j++)$ **do**
  if $(i \to j \in \{t_{i \to j}\})$ **then**
    $T_{e(j)} = T_{e(i)} + \{t_{i \to j}\}$;
    if $(T_{e(j)} \in \{T_{e_j}\})$ **then**
      $max(\{T_{e_j}\}; \{T_{e(j)}\})$;
      if $(\{T_{e_j}\} < T_{e(j)})$ **then**
        $T_{prev(j)} \leftarrow T_i$;
        $HashMap < T_j, T_{prev(j)} > \leftarrow < T_j, T_{prev(j)} >$;
        $HashSet\{T_{e_j}\} \leftarrow T_{e(j)} >$;
      else
        continue for;
      end
    else
      $T_{prev(j)} = T_i$;
      $HashMap < T_j, T_{prev(j)} > \leftarrow < T_j, T_{prev(j)} >$;
      $HashSet\{T_{e_j}\} \leftarrow T_{e(j)}$;
      continue for;
    end
  else
    continue for;
  end
end
$i++$;
if $(i \neq n)$ **then**
  start for;
end
Return:
$\{T_{e_j}\}$ - the set of early term execution of n event of the network model;
$HashMap < T_j, T_{prev(j)} >$ - the set of previous events for the next events in the format $< Key, Value >$.

Рисунок 3.4 – Алгоритм для визначення раннього часу виконання події

Далі з метою контролю термінів, до яких події мережевої моделі мають бути повністю виконані, задля уникнення випадків збільшення часу на реалізацію усього проєкту, необхідно визначити пізній термін виконання подій $T_{l(i)}$, за наступним алгоритмом:

Initialization:
i - index of the previous event, $i = 0$;
j - index of the next event, $j = n$;
n - the volume of network model events;
$\{t_{i \to j}\}$- set of network model works;
$\{T_{e_j}\}$- set of early time execution events;
$\forall j = n$ then $T_{l(n)} = T_{e(n)}$;
for $(i = j - 1; i \geq 1; i - -)$ **do**
  if $(i \to j \in \{t_{i \to j}\})$ **then**
    $T_{l(i)} = T_{l(j)} - \{t_{i \to j}\}$;
    if $(T_{l(i)} \in \{T_{l_t}\})$ **then**
      $min(\{T_{l_t}\}; \{T_{l(i)}\})$;
      if $(\{T_{l_t}\} < T_{l(i)})$ **then**
        $HashSet\{T_{l_t}\} \leftarrow T_{l(i)}$;
      else
        continue for;
      end
    else
      $HashSet\{T_{l_t}\} \leftarrow T_{l(i)}$;
      continue for;
    end
  else
    continue for;
  end
end
$j - -$;
if $(j \neq 1)$ **then**
  start for;
end
Return:
$\{T_{l_t}\}$ - the set of late term execution of n event of the network model;

Рисунок 3.5 – Алгоритм для визначення пізнього часу виконання події

Результати проведених розрахунків дозволяють оцінити максимальні обсяги робіт зважаючи на наявні ресурси, які необхідні для успішного виконання спринта. Максимальні обсяги робіт в мережевій моделі відображає так званий критичний шлях. Визначення критичного шляху організовується мережею робіт $t_{(i \to j)}$ від чергової пізньої події до події, що їй передує (передньої події зазначеної у відповідному секторі вершини мережевого графа). Побудова критичного шляху починається з кінцевої події T_n та завершується початковою подією T_1:

$$T_n \to [t_{j \to n}] \to T_j \to [t_{k \to j}] \to T_k \to \ldots \to T_m \to [t_{i \to m}] \to T_i \to [t_{1 \to i}] \to T_1, \qquad 3.8)$$

де T_n – остання подія мережевої моделі (перша для критичного шляху);

T_j – пізня проміжна подія мережевої моделі;

T_i – рання проміжна подія мережевої моделі;

T_k , T_m – проміжні події мережевої моделі;

T_1 – перша подія мережевої моделі (остання для критичного шляху).

Алгоритмічно процедура визначення критичного шляху за результатами обходу графу мережевої моделі представлено на рисунку 3.6. Вхідними даними для виконання цієї процедури є результати виконання попередніх алгоритмів.

Initialization:
j - index of the event, $j = n$;
k - temporary variable;
$HashMap < T_j, T_{prev} >$ - the set of previous events for the next events T_j in the format $< Key, Value >$.
$Stack.add(T_j)$;
while $(j \geq 1)$ **do**
    $Stack.add(k)$;
    $j - -$;
end
Return:
$Stack$.

Рисунок 3.6 – Алгоритм обходу графа мережевої моделі для визначення критичного шляху

Особливість критичного шляху виконання робіт полягає в тому, що в його межах не міститься жодного ресурсного резерву на досягнення подій (ранній та пізній терміни виконання усіх подій на критичному шляху рівні). Це свідчить про те, що будь яка затримка на виконання подій мережевої моделі, які лежать в межах критичного шляху, стимулюватиме до неповного виконання визначеного обсягу робіт на спринт або перевищення допустимого часу їх реалізації.

Проте інші роботи, що входять до складу мережевої моделі та не належать до критичного шляху, можуть мати певні резерви на досягнення

подій та виконання робіт. Значення цих резервів дозволятиме контролювати межі критичного часу початку та завершення робіт, що не лежать в межах критичного шляху. Розрахунок означених ресурсних резервів проводиться з використанням таких виразів:

$$R_{події}(j) = T_{l(j)} - T_{f(j)}; \tag{3.9}$$

$$R^{п}_{роботи}(t_{i\to j}) = T_{l(j)} - T_{f(i)} - t_{(i\to j)}; \tag{3.10}$$

$$R^{в}_{роботи}(t_{i\to j}) = T_{f(j)} - T_{f(i)} - t_{(i\to j)}; \tag{3.11}$$

$$R^{н}_{роботи}(t_{i\to j}) = T_{f(j)} - T_{l(i)} - t_{(i\to j)}, \tag{3.12}$$

де $R_{події}(j)$ – резерв часу події;

$R^{н}_{роботи}(t_{i\to j})$ – незалежний резерв роботи;

$R^{п}_{роботи}(t_{i\to j})$ – повний резерв роботи;

$R^{в}_{роботи}(t_{i\to j})$ – вільний резерв роботи;

$T_{l(i)}$ – пізній термін попередньої події;

$T_{l(j)}$ – пізній термін наступної події;

$T_{f(i)}$ – ранній термін попередньої події;

$T_{f(j)}$ – ранній термін наступної події;

t – обсяг роботи;

j – наступна подія мережевої моделі;

i – попередня подія мережевої моделі.

Резерв часу події $R_{події}(j)$ характеризує максимально допустимий період часу, на який можливо затримати виконання події n без збільшення критичного шляху та ресурсного обмеження на спринт. Повний резерв роботи $R^{п}_{роботи}(t_{i\to j})$ надає характеристику на який час можливо перенести початок виконання роботи $t_{i\to j}$ або збільшити її тривалість в межах наявних ресурсів та без збільшення загального часу спринта (довжини критичного шляху). Вільний резерв роботи $R^{в}_{роботи}(t_{i\to j})$ це час на який можливо перенести початок виконання роботи $t_{i\to j}$ або розтягнути її тривалість, не порушуючи ранніх термінів виконання подій мережевої моделі.

Незалежний резерв роботи $R^{н}_{роботи}(t_{i\to j})$ характеризує час затримки початку виконання роботи $t_{i\to j}$, не збільшуючи загального терміну виконання комплексу робіт спринта та не затримуючи жодну з інших робіт мережевої моделі. Алгоритм обходу графа мережевої моделі для визначення означених резервів представлено на рисунку 3.7.

Initialization:
i - index of the previous event, $i = 0$;
j - index of the next event, $j = n$;
n - the volume of network model events;
$\{t_{i\to j}\}$- set of network model works;
$\{T_{e_j}\}$- set of the early terms executed events;
$\{T_{l_j}\}$- set of the late terms executed events;
Stack - the critical path of network model;
while $(j \geq 1)$ **do**
  $R_{e(j)} = T_{l(j)} - T_{e(j)}$;
  if $(R_{(j)} == 0)$ **then**
    $j--$;
    continue *while*;
  else
    $HashSet\{R_j\} \to R_{(j)}$;
    $j--$;
    continue *while*;
  end
end
$i \to n$;
for $(i = j-1; i \geq 1; i--)$ **do**
  if $(i \to j \in \{t_{i\to j}\})$ **then**
    if $(T_i \notin Stack)$ **then**
      $R_{pw}(t_{(i\to j)}) = T_{l_{(j)}} - T_{e_{(i)}} - t_{(i\to j)}$;
      $R_{fw}(t_{(i\to j)}) = T_{e_{(j)}} - T_{e_{(i)}} - t_{(i\to j)}$;
      $R_{dw}(t_{(i\to j)}) = T_{e_{(j)}} - T_{l_{(i)}} - t_{(i\to j)}$;
      $HashSet\{R_{pw}(t_{(i\to j)})\} \to R_{pw}(t_{(i\to j)})$;
      $HashSet\{R_{fw}(t_{(i\to j)})\} \to R_{fw}(t_{(i\to j)})$;
      $HashSet\{R_{dw}(t_{(i\to j)})\} \to R_{dw}(t_{(i\to j)})$;
      continue *for*;
    else
      continue *for*;
    end
  else
    continue *for*;
  end
end
$j--$;
if $(j \neq 1)$ **then**
  start *for*;
else
  Return:
  $\{R_{e(j)}\}, \{R_{pw}(t_{(i\to j)})\}, \{R_{fw}(t_{(i\to j)})\}, \{R_{dw}(t_{(i\to j)})\}$.
end

Рисунок 3.7 – Алгоритм обходу графа мережевої моделі для визначення часових резервів

3.3. Приклад роботи алгоритмів, для визначення побудови та визначення параметрів мережевої моделі

Алгоритми автоматизованої побудови та визначення параметрів мережевої моделі, при плануванні проєктних робіт апробовані в процесі розроблення безпеко-орієнтованих сервісів на замовлення Державної служби України з надзвичайних ситуацій . Вхідні дані та завдання для розробки були надані від ДСНС у період воєнного стану, що підтверджує роботу в динамічних умовах та обмеженість часових ресурсів. Після отримання завдання Product Owner-ом було сформовано беклог проекту та визначено MVP. Для того, щоб визначити дату першого релізу, команда здійснила оцінку користувацьких історій та подрібнених функцій у story points, визначила залежності між користувацькими історіями, пріоритетність та почерговість виконання. У результаті було сформовано таблицю 1, де task – номер користувацької історії, яка має бути виконана, story point – вага цієї історії, previous task – порядковий номер історії, яка передує даній. Варто зазначити, що даний приклад описує лише один спринт із розробки БОС.

Таблиця 3.1. Набір вхідних даних для побудови мережевого графа

Task	1	2	3	4	5	6	7	8	9	10	11
Story Point	7	15	8	5	10	12	7	3	5	17	9
Previous Task			1	1	2	3,4	5	5	5	7,8,9	6,10

Шляхом імплементації алгоритму (додаток А. рисунок 1) програмно реалізовано процедуру побудови мережевого графа у середовищі розробки IntelliJ IDEA мовою програмування Java.

На рисунку 3.8 представлена мережева модель спринта та процесу розробки програмного забезпечення у вигляді графа.

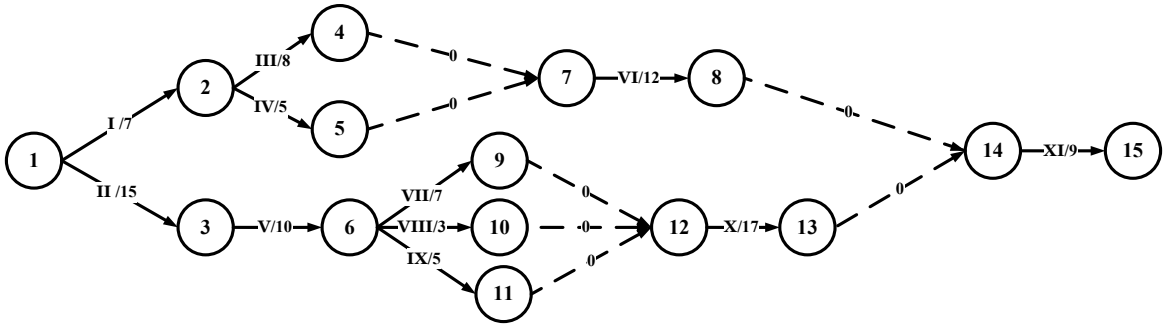


Рисунок 3.8. – Мережева модель процесу розробки програмного забезпечення представлена у вигляді графа

Для належної характеристики усіх процесів розробки програмного забезпечення на етапі планування необхідно визначити усі параметри мережевої моделі спринта. Спочатку імплементовано алгоритми обходу графа для визначення раннього та пізнього часу виконання подій спринта (проєкту). На основі алгоритмів (додаток А, рисунки 2, 3) програмно реалізовано ітераційний процес розрахунку означених параметрів. За результатами розрахунку раннього та пізнього терміну виконання подій програмно визначено критичний шлях (додаток А, рисунок 4).

На рисунку 3.9 представлений результат автоматизованого визначення критичного шляху досліджуваної мережевої моделі, визначено ранній та пізній терміни виконання подій спринта. Отриманий результат вказує на події, які є критично важливі та виконання яких не передбачає часових резервів.

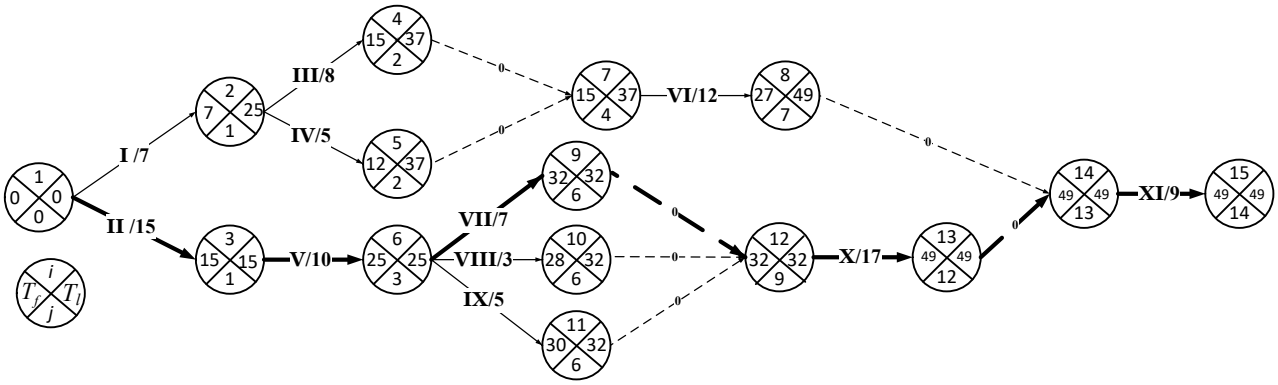


Рисунок 3.9. – Результат виконання програми, яка розраховує критичний шлях розробки ПЗ (критичні події)

Останній крок – визначення резервів подій із використанням алгоритму (додаток А, рисунок 5). Рисунок 3.10 містить результат програмної імплементації означеного алгоритму та його застосування для визначення резервів часу для досліджуваної мережевої моделі.

```
C:\Users\User\.jdks\openjdk-18\bin\java.exe "-javaagent:C:\Program Files
[2, 18], [4, 22], [5, 25], [7, 22], [8, 22], [10, 4], [11, 2]
```

Рисунок 3.10. – Результат виконання програми, яка розраховує часові резерви

На основі одержаних результатів можна зробити висновок, що термін розробки MVP даного програмного забезпечення складає 49 story points. Критично важливий шлях – це шлях, який проходить через події 1, 3, 6, 9, 12, 13, 14 та 15. Події 2, 4, 5, 7, 8, 10, 11 мають часові резерви. За потреби внесення змін до переліку робіт, або коригування часу на виконання певного обсягу робіт, розроблена система оцінки параметрів мережевої моделі на основі представлених псевдокодів дозволить миттєво переоцінити основні характеристики та передбачити можливі відхилення від плану. Ця підсистема покладена в основу системи підтримки прийняття рішень щодо оперативного менеджменту процесом розробки

спеціалізованого програмного забезпечення в динамічних умовах. Вона дозволятиме слідкувати за критичним шляхом виконання робіт, наявними часовими резервами та обсягом невиконаних робіт в режимі реального часу.

Висновки до третього розділу

У динамічних умовах етап планування процесів розроблення спеціалізованого програмного забезпечення є надзвичайно важливим. Як показав досвід подібної розробки, існуючі методи та засоби управління не корелюють з умовами, в яких проводиться розроблення спеціалізованого програмного забезпечення, де окрім змінних вимог принципово важливим є час виконання. Відповідно до результатів дослідження існуючих методів планування життєвого циклу розроблення програмного забезпечення в третьому розділі дисертації отримано такі результати:

1. Шляхом імітаційного моделювання з використанням понятійного апарату мереж Петрі отримано уніфіковану модель процесів обходу мережевих графів, яка полягає у циклічності процедури встановлення зв’язку між попередньою і поточною подіями та уможливлює її адаптивне застосування не залежно від складності та конфігурації мережевих графів.

2. Шляхом алгоритмічного відтворення розроблених імітаційних моделей побудовано чітку процедуру розрахунку раннього та пізнього термінів виконання подій, що дає змогу провести автоматизацію цих процесів для вирішення задач короткострокового планування проєктів з розроблення спеціалізованого програмного забезпечення. А також розроблено алгоритми обходу мережевого гафа та визначення часових резервів.

На основі одержаних у даному розділі наукових результатів та імплементованих алгоритмів отримано експертну комп’ютерну систему,

що дозволяє визначати основні параметри мережевої моделі щодо підтримки прийняття оперативних рішень в процесах короткострокового планування життєвого циклу проєктів розроблення спеціалізованого програмного забезпечення.

РОЗДІЛ 4. ОБҐРУНТУВАННЯ РОЗРОБЛЕНИХ МЕТОДІВ ТА МОДЕЛЕЙ УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ РОЗРОБЛЕННЯ СПЕЦІАЛІЗОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Опис експерименту для прогнозування тривалості виконання спринта для розроби БОС

Для виконання експерименту було залучено команди розробників, які в міру своїх службових обов'язків працювали над розробленням безпеко-орієнтованих сервісів у Львівському державному університеті безпеки життєдіяльності. Для прогнозування тривалості виконання спринта було використано повнофакторний експеримент з подальшим обґрунтуванням із залученням лінійної регресії.

Для побудови рівняння регресії було прийнято використовувати два показники: досвід команди та кількість допустимих змін на спринт. Граничні значення параметрів експериментального дослідження були встановлені шляхом середнього показника цього значення у команді під час виконання спринта.

У таблиці 4.1 продемонстровані результати обчислень середньо-статистичних значень показників за спринт.

Таблиця 4.1 – Граничні значення параметрів експериментального дослідження

№ з/п	Параметр	Мінімальне значення	Максимальне значення	Примітка
1.	Досвід команди	D_{min} = 2,33 роки	D_{max} = 9,66 роки	Середній вік команди для досвідчених та молодих (студентських) команд
2.	Кількість змін на спринт	Z_{min} = 1%	Z_{max} = 13%	Максимальна та мінімальна кількість змін, яку команди розробників готові прийняти за спринт

Для визначення цих показників до експерименту залучались два види команд розробників, перша з яких була студентською і її середній досвід у сфері розробки програмного забезпечення становить 2,33 роки, а друга – команда досвідчених фахівців, викладачів та аспірантів, середній досвід якої – 9,66 років.

Також для прогнозування часу виконання спринта було прийнято застосовувати такий параметр, як кількість змін, які можна вносити під час виконання спринта (оскільки даний показник здійснює значний вплив на швидкість та якість розробки ПП). Його значення представлене у % відношенні і становить 1% (коли змін практично немає, максимум якісь неточності у завданні, які дуже швидко вирішуються) та середній показник допустимої кількості змін – 13% від існуючих завдань. При таких обставинах програмний продукт все ще може розроблятись, не нехтуючи часовими ресурсами та якістю (якщо цей показник значно більший це означає, що частина функціоналу, який розробляється постійно, зазнає змін, а рух у розробці далі просто неможливий).

Відповідно до [25] планування та оброблення результатів повнофакторного експерименту складається з наступних етапів: кодування факторів; побудова план-матриці експерименту; рандомізація дослідів; реалізація плану експерименту; перевірка адекватності моделі.

Отож, проведемо кодування факторів, здійснивши переведення натуральних величин у безрозмірні. Результати зазначені у таблиці 4.2.

Таблиця 4.2 – Рівні зміни факторів

Рівень факторів		*D*, років		*Z*, %	
Назва	Кодоване значення	$\tilde{X}$	$\ln \tilde{X}_1$	$\tilde{X}$	$\ln \tilde{X}_2$
Верхній	+1	9,66	2,2679	0,13	-2,04
Нижній	-1	2,33	0,8458	0,1	-4,6051

Використовуючи методики [11, 25, 133] та таблицю 4.2, побудуємо план-матрицю експериментальних досліджень для математичного

опрацювання експериментально отриманих даних з використанням методу повнофакторного експерименту типу 2^2. Результати досліджень, кожне з яких було проведено двічі, зображені в таблиці 4.4 та на рисунку 4.1.

Таблиця 4.3 – План-матриця експериментальних досліджень

№ досліду	Фактори			
	X_1		X_2	
	код	*D*, роки	код	*Z*, %
1.	-1	2,33	-1	0,01
2.	1	9,66	-1	0,01
3.	-1	2,33	1	0,13
4.	1	9,66	1	0,13

Таблиця 4.4 – Результати експериментальних досліджень

№ дос-ду	Результати дослідів								$\overline{T}$, *s.p.*	$\ln \overline{T}$
	T(1), s.p.	*T(2), s.p.*	*T(3), s.p.*	*T(4), s.p.*	*T(5), s.p.*	*T(6), s.p.*	*T(7), s.p.*	*T(8), s.p.*		
1.	73	69	68	70	72	67	71	74	71	4,262679877
2.	55	53	59	54	53	50	52	54	54	3,988984047
3.	80	76	77	75	79	80	82	78	78	4,356708827
4.	65	63	66	65	63	62	62	67	64	4,158883083

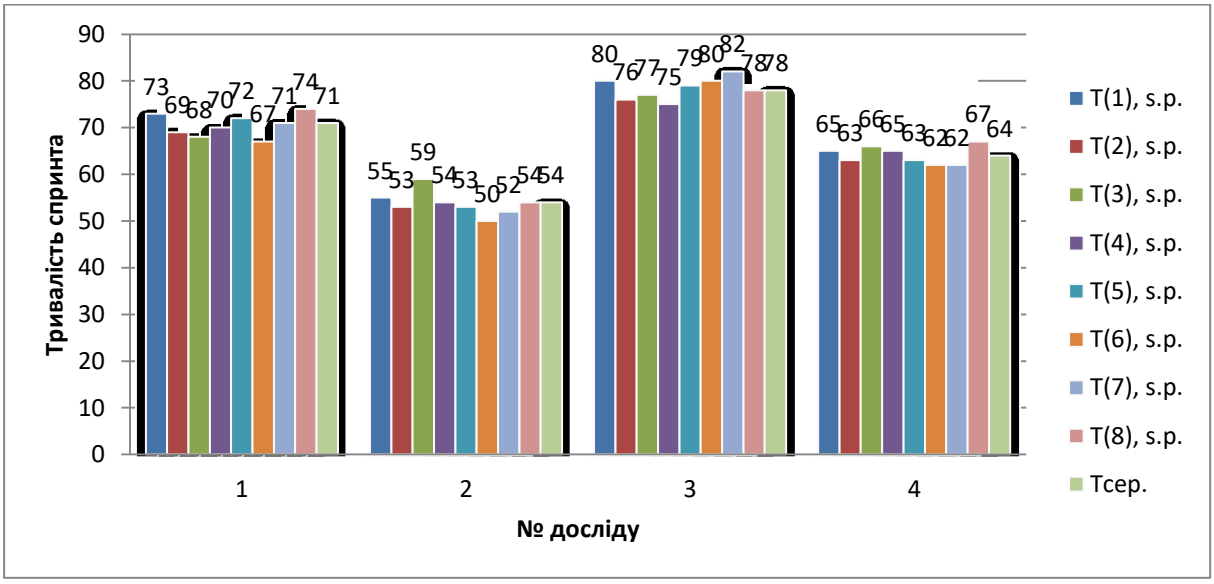


Рисунок 4.1. – Результати експериментальних досліджень

4.2. Математичне опрацювання результатів проведених експериментальних досліджень

Наступним кроком, після виконання експериментів, потрібно здійснити перевірку відтворюваності процесів при однаковому числі паралельних дослідів за критерієм Кохрена. Після цього, за допомогою критерію Фішера, потрібно здійснити перевірку адекватності моделі, а за допомогою критерію Стьюдента – оцінити значущість коефіцієнтів регресії для того, щоб визначити, чи значущий коефіцієнт (якщо ні, то це означає, що він не впливає на параметри оптимізації).

З метою перевірки достовірності результатів прогнозування тривалості розробки програмного забезпечення проведемо логарифмічно-лінійне відтворення функції.

Операції перевірки відтворюваності дослідів, адекватності моделі та оцінювання значущості коефіцієнтів регресії проведемо саме для цього рівняння. Для виконання окреслених завдань незалежні змінні $\widetilde{X}_i$ (таблиця 4.2) перетворено в безрозмірні величини за залежностями [25]:

$$X'_i = \frac{2 \cdot (\ln \widetilde{X}_i - \ln \widetilde{X}_{i\max})}{\ln \widetilde{X}_{i\max} - \ln \widetilde{X}_{i\min}} + 1 \ , \tag{4.1}$$

або:

$$X_i = \frac{2X_i - X_i^+ - X_i^-}{X_i^+ - X_i^-} \ , \quad i = 1; 2; 3 \ . \tag{4.2}$$

Як результат, отримаємо наступні вирази:

$$\begin{aligned} X'_1 &= 1,4 \ln D - 2,18, \\ X'_2 &= 0,78 \ln Z + 2,59. \end{aligned} \tag{4.3}$$

Рівняння регресії, що визначає залежність прогнозованої тривалості розробки програмного забезпечення від двох незалежних чинників (*D, Z*) з кодованими змінними, що враховують взаємодію даних чинників, виглядає наступним чином:

$$\ln T = a'_0 + a'_1 X'_1 + a'_2 X'_2 + a'_{12} X'_1 X'_2 . \quad (4.4)$$

Визначення коефіцієнтів a'_n для рівняння (4.4) з врахуванням експериментально одержаних значень $\overline{T}_i$ проводились за залежністю [25]:

$$a'_n = \frac{1}{N} \sum_{i=1}^{N} X_{in} \ln \overline{T}_i \ , \quad (4.5)$$

де X_{in} – код *n*-го фактора *i*-го досліду (відповідно до табл. 4.3); $\overline{T}_i$ – середнє арифметичне результату *i*-го досліду за певних значень факторів; *N* – кількість дослідів (в цьому випадку – 4).

Результати розрахунку коефіцієнтів a'_n для рівняння (4.4) наступні: $a'_0 = 4{,}191458876$, $a'_1 = -0{,}118869891$, $a'_2 = 0{,}069710775$, $a'_{12} = 0{,}018534544$.

За однакового числа паралельних дослідів *r* (в нашому випадку *r* = 8) на кожному поєднанні рівнів факторів відтворюваність перевіряється за критерієм Кохрена [133]:

$$G = \frac{S^2_{pi\max}}{S^2_B} \le G_{(0{,}05;N;\text{fr})} \ , \quad (4.6)$$

де $S^2_{pi\max}$ – максимальне значення дисперсії розсіювання S^2_{pi} (за формулою (4.7)); S^2_B – значення дисперсії відтворюваності (за формулою (4.8)); *N* – кількість дослідів; f_r – кількість ступенів вільності кожної оцінки (в нашому випадку $f_r = r\text{-}1 = 8\text{-}1 = 7$); $G_{(0{,}05;N;\text{fr})}$ –значення критерію Кохрена табличне.

Для початку необхідно обчислити значення дисперсії розсіювання за формулою:

$$S^2_{pi(1,2)} = (\ln T_{i(1–8)} - \ln \overline{T})^2. \quad (4.7)$$

Для зручності порівняння результатів розрахунків дисперсії розсіювання та знаходження її максимального значення одержані результати зображено у таблиці 4.5.

Таблиця 4.5 – Значення дисперсії розсіювання S^2_{pi}

№ д-ду	S_{pi}	S^2_{pi}	№ д-ду	S_{pi}	S^2_{pi}
1(T1)	0,058335577	0,00340304	3(T1)	0,020521636	0,000421138
1(T2)	-0,025046032	0,000627304	3(T2)	-0,030771659	0,000946895
1(T3)	-0,039644831	0,001571713	3(T3)	-0,017699577	0,000313275
1(T4)	-0,010657294	0,000113578	3(T4)	-0,044016885	0,001937486
1(T5)	0,017513582	0,000306726	3(T5)	0,007942854	6,30889E-05
1(T6)	-0,054459917	0,002965883	3(T6)	0,020521636	0,000421138
1(T7)	0,003527341	1,24421E-05	3(T7)	0,045214248	0,002044328
1(T8)	0,044912557	0,002017138	3(T8)	-0,004796172	2,30033E-05
2(T1)	0,022989518	0,000528518	4(T1)	0,013552966	0,000183683
2(T2)	-0,014051753	0,000197452	4(T2)	-0,017699577	0,000313275
2(T3)	0,093193777	**0,00868508**	4(T3)	0,028820439	0,000830618
2(T4)	0,00464038	2,15331E-05	4(T4)	0,013552966	0,000183683
2(T5)	-0,014051753	0,000197452	4(T5)	0,013552966	0,000183683
2(T6)	-0,072320662	0,005230278	4(T6)	-0,033699918	0,001135685
2(T7)	-0,033099948	0,001095607	4(T7)	-0,033699918	0,001135685
2(T8)	0,00464038	2,15331E-05	4(T8)	0,043858316	0,001923552

Дисперсія відтворюваності обчислюється як сума значень дисперсії розсіювання:

$$S^2_B = \sum_{i=1}^{16} S^2_{pi}. \quad (4.8)$$

Як результат, отримаємо значення $S^2_B = 0,039055488$

Після отримання значень дисперсій розсіювання та відтворюваності можна розрахувати критерій Кохрена та порівняти його із критичним значенням [133] із використанням залежності (4.6):

$$G = \frac{S^2_{pi\max}}{S^2_B} = 0,222377966 < G_{(0,05;4;7)} = 0,637.$$

Відтак, можна зробити висновок, що гіпотеза однорідності дисперсій підтверджується, оскільки $G<G_{кр}$.

Наступний етап – оцінка значущості коефіцієнтів регресії. Дану оцінку здійснюватимемо за допомогою критерію Стьюдента [11, 25, 133], який означає, що коефіцієнт вважається значущим, якщо виконується нерівність з урахуванням половини довжини довірчого інтервалу:

$$\left|a'_i\right| \geq \Delta a'_i = t_{(0,05;f)} \cdot S(a'_i) \ , \tag{4.9}$$

де $t_{(0,05;f)}$ – критичне значення критерію Стюдента для $f=N(r-1)$ ($f=4(8-1)=28$), $\alpha=0,05$; згідно з [25, 133] $t=2,05$),

$$S(a'_i) = \pm\sqrt{\frac{S_B^2}{N^2 \cdot r}} = \pm 0,01746771. \tag{4.10}$$

Отже, половина довжини довірчого інтервалу $\Delta a'_i$ дорівнюватиме 0,035808812.

Враховуючи нерівність 4.9, можна зробити висновок, що коефіцієнти $\left|a'_0\right|$= 4,191458876, $\left|a'_1\right|$= 0,118869891, $\left|a'_2\right|$= 0,069710775 є значущими, а коефіцієнт $\left|a'_{12}\right|$= 0,018534544 – незначущий.

Зважаючи на це, рівняння (4.4) набуде вигляду:

$$\ln T = 4,19 - 0.12X'_1 + 0,069X'_2. \tag{4.11}$$

Відповідно до методики повнофакторного експерименту наступним кроком є перевірка адекватності моделі за критерієм Фішера.

Модель вважається адекватною у випадку виконання нерівності [25]:

$$F = \frac{S_{a\partial}^2}{S_\partial^2} \leq F_{кр_{(0,05;f_1;f_2)}} \ , \tag{4.12}$$

де $S_{a\partial}^2$ – дисперсія адекватності, що визначається за залежністю (4.13); S_∂^2 – похибка досліду, що визначається за залежністю (4.14); $F_{(0,05;f_1;f_2)}$ – критичне значення критерію Фішера за умови α=0,05, $f_1=N-m$, $f_2=N(r-1)$.

В даній ситуації $f_1=4-3=1$ (m – число членів апроксимуючого полінома), $f_2=4(8-1)=28$, за таких умов табличне значення [133] критерію Фішера рівне $F_{кр}$= 4,28.

$$S_{ad}^2 = \frac{1}{N-m}\sum_{i=1}^{N}(\ln \overline{T}_i - \widehat{\ln T}_i)^2, \quad (4.13)$$

де $\widehat{\ln T}_i$ – розрахункове значення параметра за залежністю (4.11) після підстановки значень (–1) та (+1) згідно з план-матрицею експериментальних досліджень.

$$S_д^2 = \frac{S_B^2}{N(r-1)}. \quad (4.14)$$

Проміжні значення для розрахунку дисперсії адекватності занесені в допоміжну таблицю 4.6.

Таблиця 4.6 – Проміжні дані розрахунку дисперсії адекватності

№ досліду	$\ln \overline{T}_i$	$\widehat{\ln T}_i$	$\ln \overline{T}_i - \widehat{\ln T}_i$	$(\ln \overline{T}_i - \widehat{\ln T}_i)^2$
1	4,259152537	4,240617993	0,018534544	0,000343529
2	3,984343667	4,002878211	-0,018534544	0,000343529
3	4,361504999	4,380039542	-0,018534544	0,000343529
4	4,160834303	4,14229976	0,018534544	0,000343529
$\sum_{i=1}^{N}(\ln \overline{T}_i - \widehat{\ln T}_i)^2$.				0,001374117

За формулами (4.13) та (4.14) обчислюємо дисперсію адекватності моделі та похибку досліду: $S_{ad}^2 = 0,001374117,\ S_д^2 = 0,001394839.$ Після цього обчислюємо розрахункове значення критерію Фішера, за формулою (4.12): F = 0,98514405.

Зіставивши розрахункове та критичне значення F = 0,99< $F_{кр}$ = 4,28, можна зробити висновок, що модель (4.11) є адекватною.

Останній етап – визначення коефіцієнту кореляції, для встановлення точності опису експериментальних даних рівнянням регресії. Для цього використаємо наступну залежність:

$$R = \sqrt{1 - \frac{\sum_{i=1}^{N} (\ln \overline{T}_i - \overline{T}_i)}{\sum_{i=1}^{N} (\ln \overline{T}_i - \overline{T}_i^*)^2}}, \qquad (4.15)$$

де $\overline{T}^*$ – середнє значення функції $\ln \overline{T}_i$, яке визначається за залежністю (4.16) та дорівнює 4,1914:

$$\overline{T}^* = \frac{1}{N} \sum_{i=1}^{N} \ln \overline{T}_i \ . \qquad (4.16)$$

Проміжні значення для розрахунку коефіцієнта множинної кореляції занесені в допоміжну таблицю 3.8.

Таблиця 3.8 – Проміжні дані розрахунку коефіцієнта множинної кореляції

№ досліду	$\ln \overline{T}_i$	$\ln \overline{T}_i - \overline{T}^*$	$(\ln \overline{T}_i - \overline{T}^*)^2$
1	4,259152537	0,06769366	0,004582432
2	3,984343667	-0,207115209	0,04289671
3	4,361504999	0,170046122	0,028915684
4	4,160834303	-0,030624573	0,000937864
$\sum_{i=1}^{N} (\ln \overline{T}_i - \overline{T}^*)^2$:			0,07733269

Підставивши отримані значення у формулу (4.15), отримаємо коефіцієнт множинної кореляції R = 0,9822. Розрахункове значення коефіцієнта R наближається до одиниці, а, отже, можемо стверджувати, що рівняння (4.11) майже повністю описує результати експериментальних досліджень.

Для здійснення переходу до моделі в натуральних змінних підставимо рівняння (4.1) в модель (4.17) та проведемо наступний розрахунок:

$$\ln T = 4{,}19 - 0.12 X_1' + 0{,}069 X_2', \qquad (4.17)$$

$$\ln T = 4{,}19 - 0.12 * (1{,}4 \ln D - 2{,}18) + 0{,}069 * (0{,}78 \ln Z + 2{,}59) = 4{,}63 - 0.17 D + 0.054 \ln Z \qquad (4.18)$$

Отже, кінцева модель визначення впливу незалежних чинників D, Z на час виконання спринта T набуде наступного вигляду:

$$T = \exp(4{,}63 - 0.17 \ln D + 0.054 \ln Z). \qquad (4.20)$$

Дана модель може застосовуватись з метою прогнозування тривалості розробки спеціалізованого програмного забезпечення у динамічних умовах. Особливість методу прогнозування полягає у можливості обчислення часу розробки ПП, задаючи досвід команди D та кількість змін Z, які можна вносити за ітерацію.

Використовуючи отриману залежність логарифмічно-логарифмічного відображення функції відгуку, проведемо розрахунок очікуваного значення тривалості розробки програного забезпечення за умови різних способів реалізації. На рисунку 4.2 зображений результат, де кількість змін фіксована і рівна 1%, 5%, 15% та 25%, а досвід команди коливається.

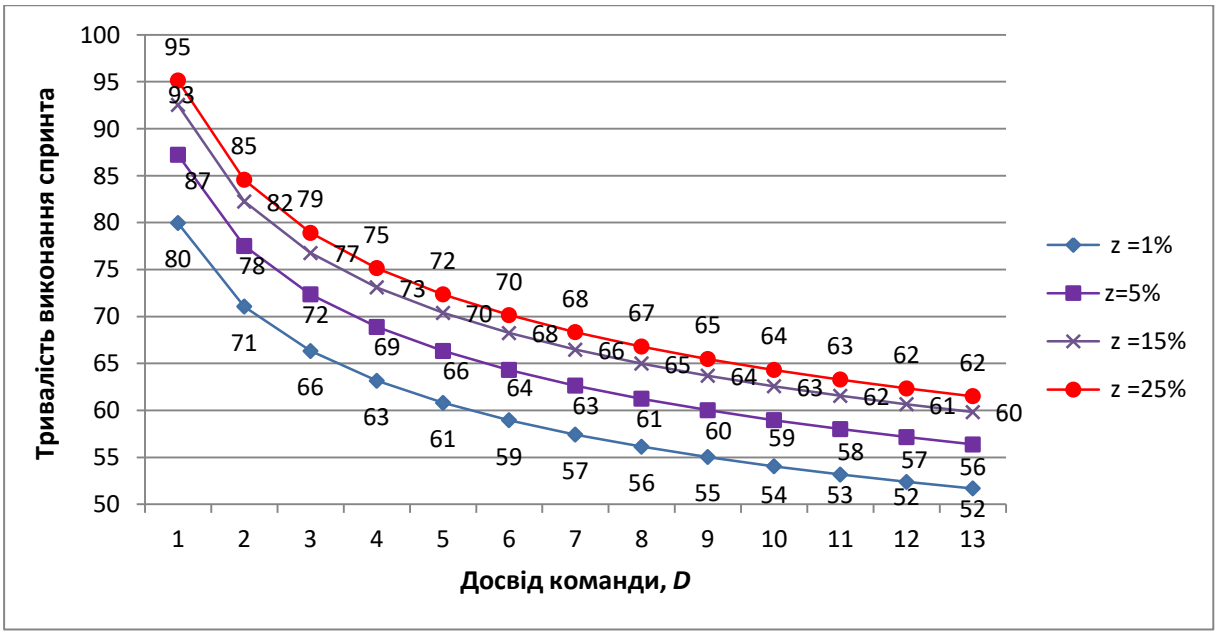


Рисунок 4.2. – Результат прогнозування

Результат, де фіксований досвід команди і рівний 2, 4, 6, 10 та 12 років, а показник кількості змін коливається, зображений на рисунку 4.3.

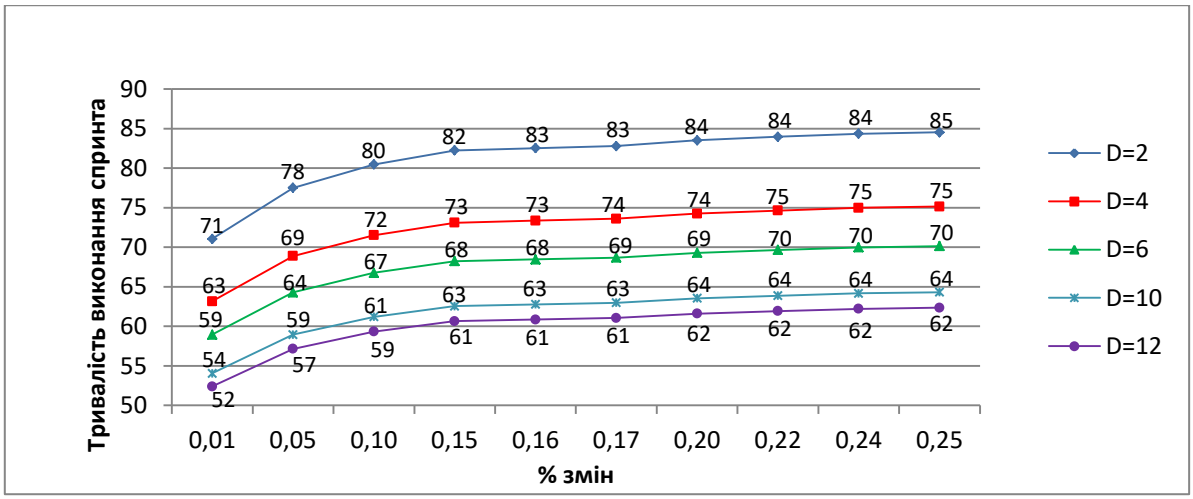


Рисунок 4.3. – Результат прогнозування

Висновки до четвертого розділу

У результаті виконання даного розділу шляхом експериментальних досліджень було розроблено логарифмічно-логарифмічне рівняння лінійної регресії для прогнозування терміну розроблення спеціалізованого програмного забезпечення на один спринт.

Для того, щоб одержати такий результат, було визначено основні параметри, які впливають на час розробки СПЗ, зокрема, це досвід команди та кількість допустимих змін.

Після виконання експериментів було здійснено перевірку відтворюваності процесів при однаковому числі паралельних дослідів за критерієм Кохрена, перевірку адекватності моделі з критерієм Фішера, оцінено значущість коефіцієнтів рівняння за критерієм. Стьюдента. На основі проведених обчислень було розраховано коефіцієнт кореляції, який становить R = 0,9822 і означає, що одержане рівняння регресії майже повністю описує результати експериментальних досліджень.

РОЗДІЛ 5. МОДЕЛЮВАННЯ ТА ВПРОВАДЖЕННЯ АДАПТИВНИХ СИСТЕМ УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ У ДИНАМІЧНИХ УМОВАХ.

Практичне впровадження одержаних результатів полягає, насамперед, у розроблені моделей та методів, а також на їх основі інформаційної системи підтримки прийняття рішень в управлінні життєвим циклом розроблення спеціалізованого програмного забезпечення та аналізу одержаних результатів.

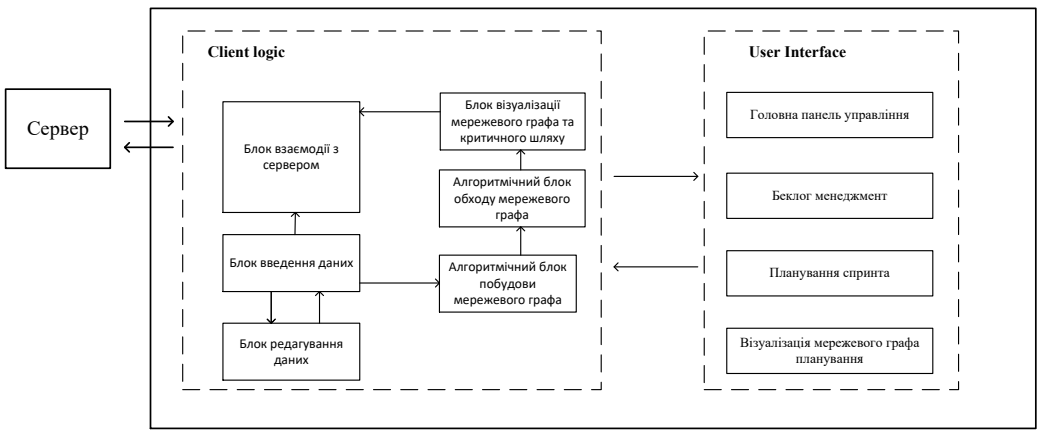
Розроблені у дисертаційній роботі теоретичні та науково-методичні положення впроваджені в освітній процес Львівського державного університету безпеки життєдіяльності, при реалізації студентських R&D проєктів, а також науково-дослідних робіт на замовлення Державної служби України із надзвичайних ситуацій. Зокрема, на базі розроблених методів та моделей було удосконалено процес управління життєвим циклом розроблення наступних безпеко-орієнтованих сервісів: Інформаційно-аналітична система «Інтерактивний інспектор» (на замовлення ДСНС України), Мобільний додаток для обліку та пошуку найближчих об’єктів укриття (студентський R&D проєкт), Веб-сервіс обліку пунктів вакцинації та тестування на COVID-19 (студентський R&D проєкт), інформаційна система «Я-Доброволець» (ініціативна, на замовлення ЛДУ БЖД, ДСНС, МВС України).

5.1. Програмна реалізація основних функцій інформаційної системи підтримки прийняття рішень в управлінні життєвим циклом спеціалізованого програмного забезпечення

Реалізація інформаційної системи підтримки прийняття рішень в управлінні життєвим циклом розроблення спеціалізованого програмного

забезпечення представлена у форматі web-застосунку. Дана система розроблена із використанням мови програмування Java та фреймворка Spring Boot, для розробки зовнішньої частини сайту використано html, CSS (Bootstrap 5), JavaScript та Thymeleaf (серверний механізм Java-шаблонів призначений для обробки HTML, XML, JavaScript, CSS).

Розглянемо концептуальну модель інформаційної системи підтримки прийняття рішень в управлінні життєвим циклом спеціалізованого програмного забезпечення (далі – ІСППР в УЖЦ СПЗ) шляхом візуалізації архітектури клієнтської частини застосунку (рисунок 5.1.). Архітектура цієї частини застосунку відповідає за реакцію на дії користувача та взаємодію із сервером. У ролі потенційного «клієнта» ми розглядаємо проєкт менеджера (PM), власника продукту (PO) або ж скрам майстра (SM), залежно від обсягу проєкту, складу команди тощо. «Клієнт» вводить дані із користувацькими історіями, що бере команда в розробку на спринт у систему, згідно їх пріоритетності та можливостей команди. Далі ці дані зберігаються у беклозі спринта, а також використовуються для побудови мережевого графа планування. Після введення, обробки та зберігання інформації активується алгоритмічний блок обчислень та блок візуалізації мережевого графа планування, які представлені на користувацькому інтерфейсі веб-застосунку. Оскільки користувач має можливість редагувати дані, то між блоками введення та редагування даних існує двосторонній зв'язок.



Рисунокк 5.1 - Архітектура клієнтської частини web-застосунку

Основне призначення роботи серверної частини (рисунок 5.2) в автоматичному режимі – це опрацювання запитів, що надходять з клієнтської частини та зворотне надсилання результатів їх оброблення через блок взаємодії з клієнтом.

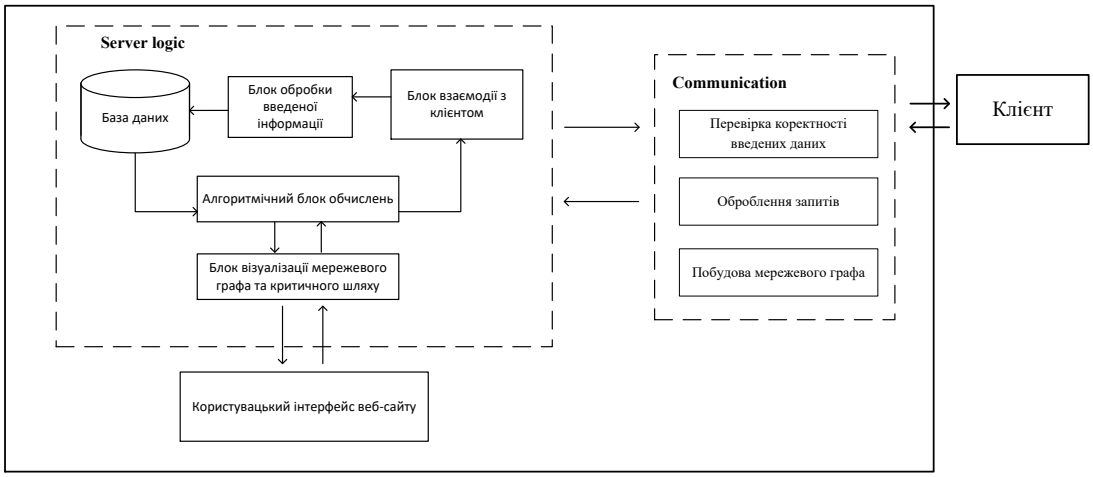


Рисунок 5.2 - Архітектура серверної частини системи

На рисунку 5.3 зображена UML діаграма основних класів даної системи.

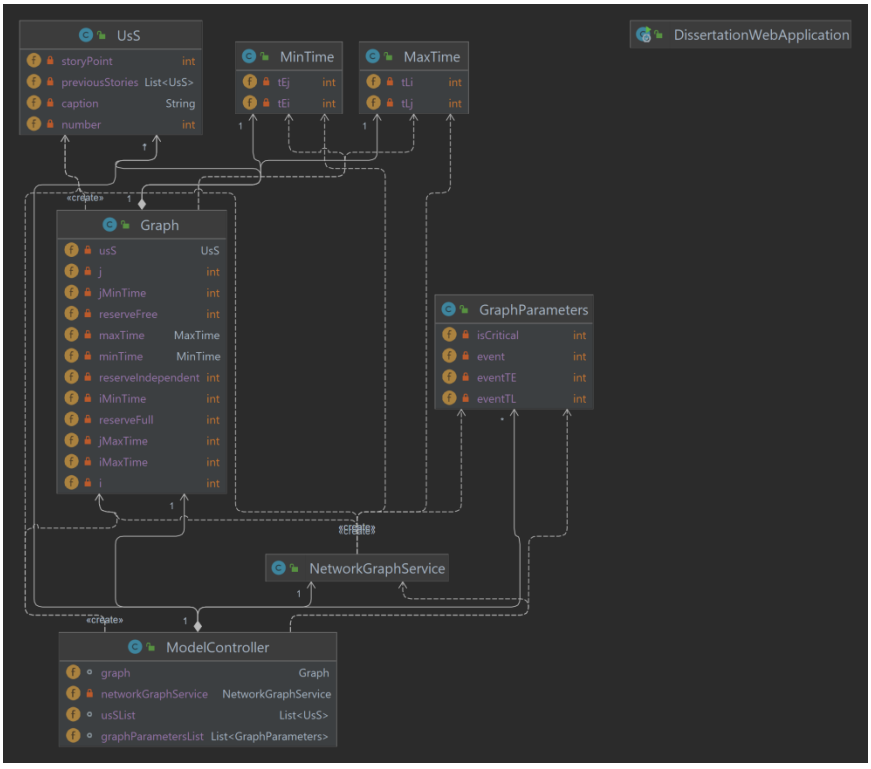


Рисунок 5.3 – UML діаграма класів програмної системи

З даної діаграми можна спостерігати, що для побудови мережевого графа було створено 4 сутності, зокрема Graph, UsS, MinTime та MaxTime.

Для зручності у класі Graph було реалізовано метод із побудови мережевого графа, а інші методи були реалізовані у класі NetworkGraphService та передані ModelController. Детальний код програми описаний у додатку Б.

Якщо ж говорити про користувацький інтерфейс та функціонал, то головна сторінка програми має наступний вигляд (рисунок 5.4):

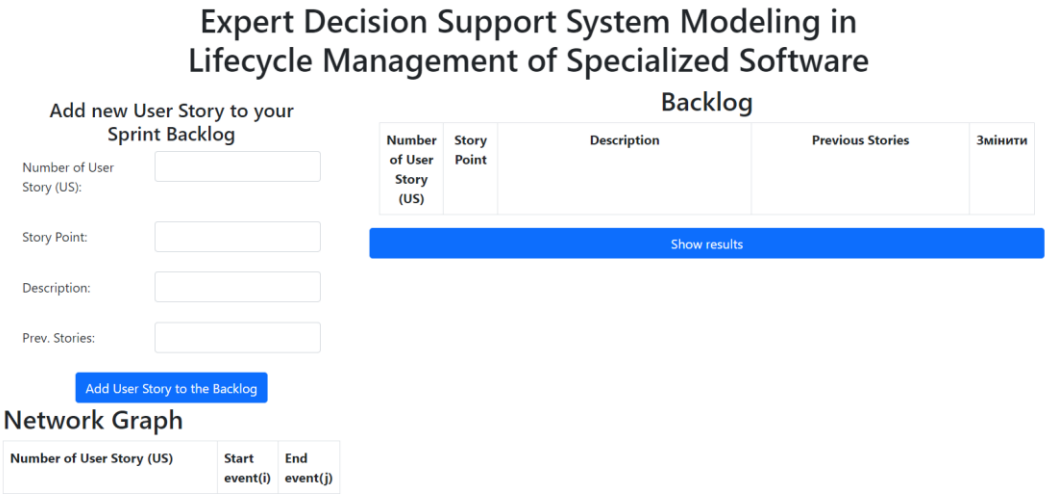


Рисунок 5.4 – Головна сторінка web-застосунку ІСППР в УЖЦ СПЗ

На даній сторінці реалізований функціонал додавання користувацької історії у беклог, перегляд та редагування беклогу. Окрім цього сформована таблиця із побудованим мережевим графом (на основі розробленого алгоритму побудови мережевого графа), де наявними є номер користувацької історії а також початкові та кінцеві події виконання. Приклад реалізації функціоналу головної сторінки зображений на рисунку 5.5.

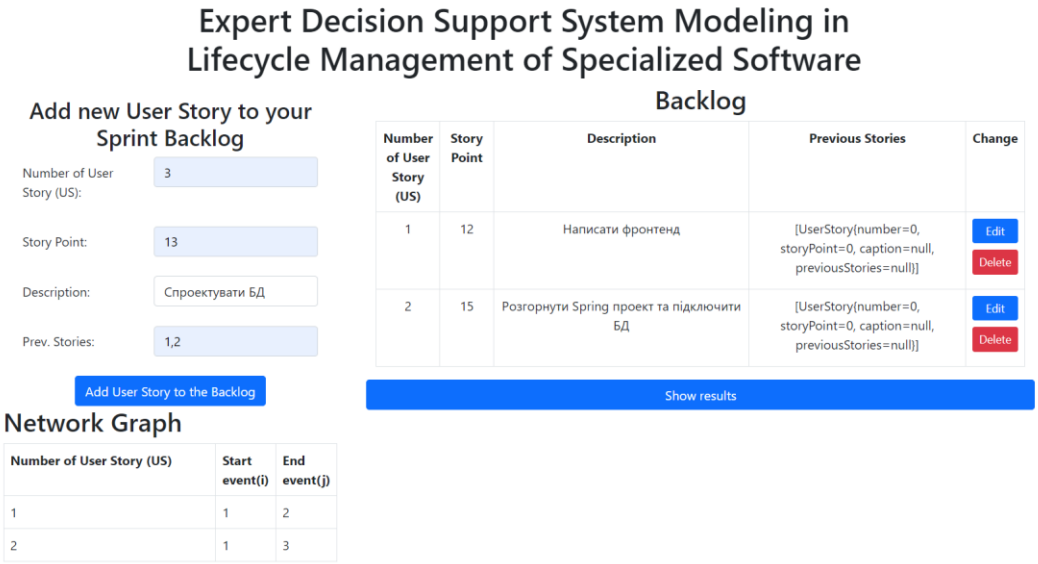


Рисунок 5.5 – Приклад реалізації функціоналу

При натисканні на кнопку «Show results» появиться нове вікно, в якому будуть обчислюватись результати роботи розроблених алгоритмів обходу мережевого графа, зокрема, таблиця результатів роботи алгоритмів обчислення мінімального і максимального часу початку виконання та завершення події та беклог із користувацькими історіями (критичним шляхом та резервами). На рисунку 5.6 представлений приклад даного вікна.

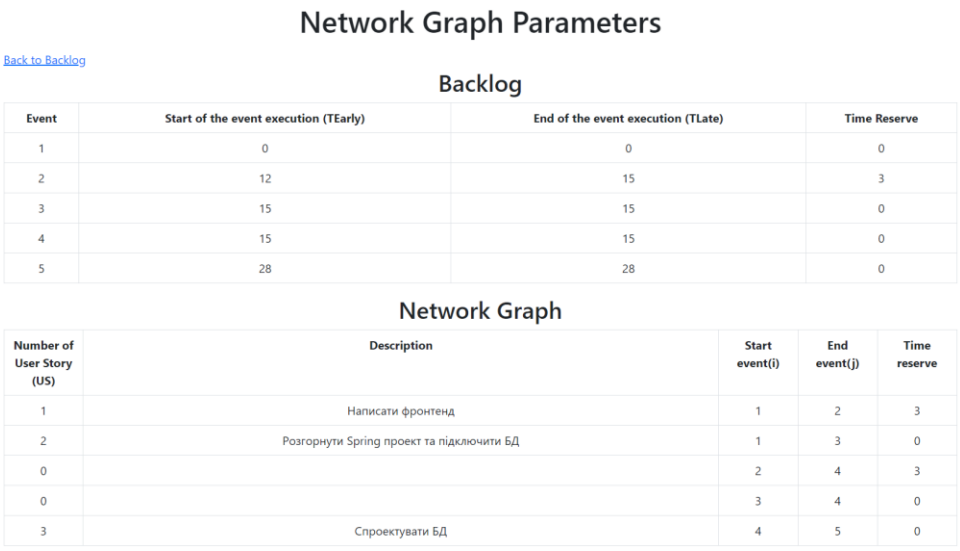


Рисунок 5.6 – Приклад вікна, з відображеними пезультатами роботи розроблених алгоритмів обходу мережевого графа

З наявного вікна можна спостерігати, що подія два має резерв часу її початку у два сторі пойнта, а користувацька історія під номером один має часовий резерв у три сторі пойнта (фіктивну роботу нуль не беремо до уваги, оскільки вона не несе ніякої цінності для проєкта, а лише була необхідна для побудови мережевого графа). Від так, критично важливими для проекту є користувацькі історії, у яких часовий резерв рівний нулю, тобто у прикладі це історії під номером два та три.

На рисунку 5.7 зображена графічна інтерпретація мережевого графу етапу планування спринта.

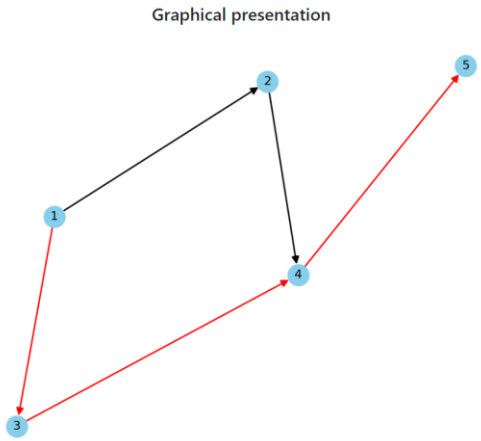
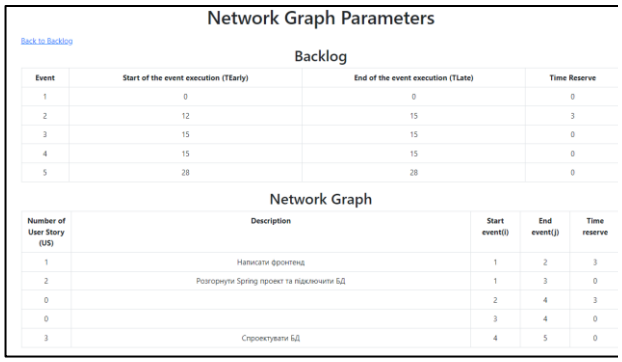


Рисунок 5.7 – Графічне представлення мережевого графа планування (критичний шлях)

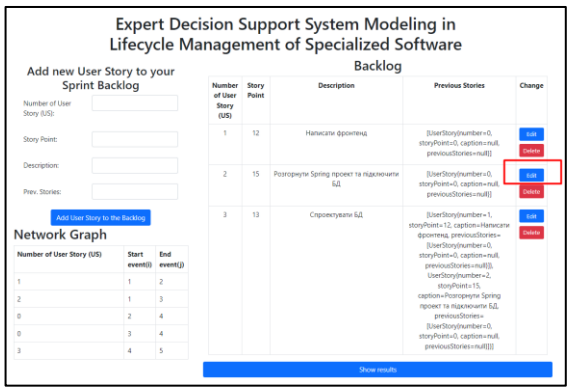
Код для реалізації даного функціоналу був написаний мовою програмування Python, зокрема, із використанням таких бібіліотек як matplotlib, networkx, io та numpy. Спершу був створений невеликий Flask додаток, який обробляв запити, виконував скрипт на Python і повертав зображення графа, а потім за допомогою Python API, зокрема використовуючи RestTemplate для відправки даних до Flask, було отримано зображення графа на веб-сторінці.

Враховуючи отримані результати, а також хід виконання спринта, команда розробників може динамічно змінювати оцінку (що у свою чергу змінює час виконання) користувацької історії. Для цього функціоналом програми передбачено кнопку «Back to Backlog», натиснувши на яку

користувач повертається на головну сторінку і може редагувати беклог спринта. Приклад зображено на рисунку 5.8.

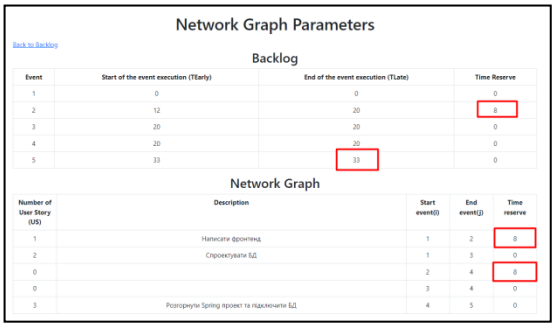


а) – 1 етап (вибір кнопки “Back to Backlog”)



б) 2 етап (вибір кнопки “Edit”)

в) – 3 етап (редагування даних)



г) – 4 етап (перегляд отриманих результатів)

Рисунки 5.8 – Процес перепланування проєкту

Дана інформаційна система дозволяє здійснювати перепланування проекту стільки разів, скільки потрібно команді без втрати часових та матеріальних ресурсів.

5.2. Апробація отриманих результатів

На основі розроблених у дисертаційному дослідженні методів та моделей було створено велику кількість безпеко-орієнтованих сервісів. Розглянемо практичне впровадження деяких із них у розроблену

інформаційну систему підтримки прийняття рішень в управлінні життєвим циклом спеціалізованого програмного забезпечення.

5.2.1. Інформаційно-аналітична система «Інтерактивний інспектор»

Інформаційна система була розроблена на замовлення Державної служби України із надзвичайних ситуацій. Працює вона у форматі чат-боту на платформі Telegram. Призначена для оцінки протипожежного стану об’єкта залежно від параметрів його функціонування (визначення кількості вогнегасників, необхідність обладнання системами протипожежного захисту, визначення ступеню ризику від провадження господарської діяльності тощо). Система орієнтована на автоматизацію окремих процесів щодо перевірки протипожежного стану об’єкта і використовується посадовими особами ДСНС України та суб’єктами господарювання. На рисунку 5.9 зображений приклад роботи такої аналітичної системи.

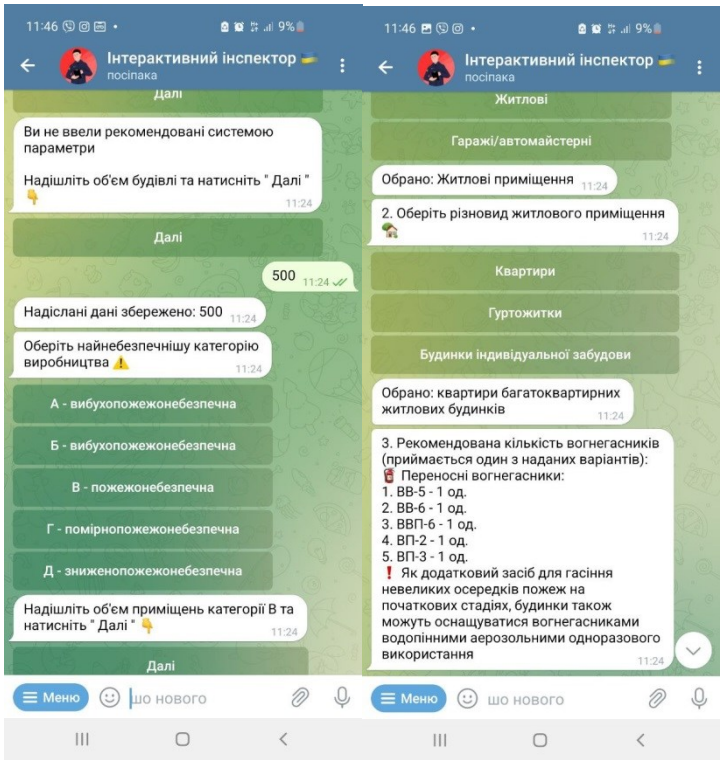
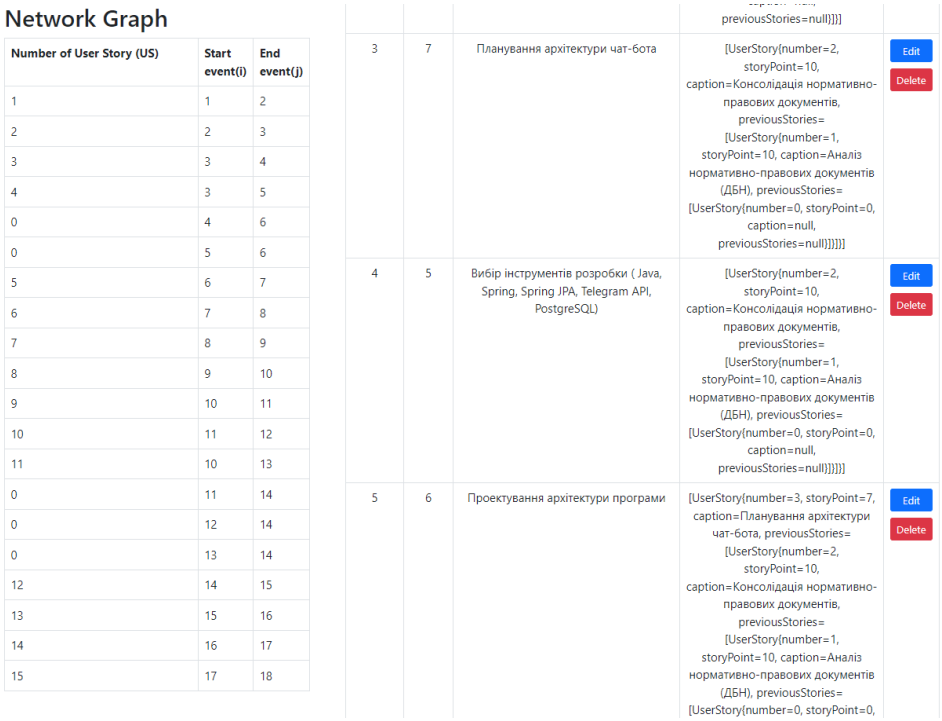


Рисунок 5.9 – Приклад роботи інформаційно-аналітичної системи «Інтерактивний інспектор»

Посилання на систему: **@Interactive_Inspector_Bot**

Вхідні дані та завдання для розробки системи були надані від Державної служби України з надзвичайних ситуацій у період воєнного стану, що підтверджує роботу в динамічних умовах та обмеженість часових ресурсів. На рисунку 5.10 представлений мережевий граф планування одного спринта для розробки даного продукту, котрий занесено до розробленої ІСППР в УЖЦ СПЗ, а на рис 5.11 – приклад його роботи.



Network Graph

Number of User Story (US)	Start event(i)	End event(j)
1	1	2
2	2	3
3	3	4
4	3	5
0	4	6
0	5	6
5	6	7
6	7	8
7	8	9
8	9	10
9	10	11
10	11	12
11	10	13
0	11	14
0	12	14
0	13	14
12	14	15
13	15	16
14	16	17
15	17	18

			previousStories=null}]}]	
3	7	Планування архітектури чат-бота	[UserStory{number=2, storyPoint=10, caption=Консолідація нормативно-правових документів, previousStories= [UserStory{number=1, storyPoint=10, caption=Аналіз нормативно-правових документів (ДБН), previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}]}]	Edit Delete
4	5	Вибір інструментів розробки (Java, Spring, Spring JPA, Telegram API, PostgreSQL)	[UserStory{number=2, storyPoint=10, caption=Консолідація нормативно-правових документів, previousStories= [UserStory{number=1, storyPoint=10, caption=Аналіз нормативно-правових документів (ДБН), previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}]}]	Edit Delete
5	6	Проектування архітектури програми	[UserStory{number=3, storyPoint=7, caption=Планування архітектури чат-бота, previousStories= [UserStory{number=2, storyPoint=10, caption=Консолідація нормативно-правових документів, previousStories= [UserStory{number=1, storyPoint=10, caption=Аналіз нормативно-правових документів (ДБН), previousStories= [UserStory{number=0, storyPoint=0,	Edit Delete

Рисунок 5.10 – Аналітичне представлення мережевого графа планування спринта інформаційно-аналітичної системи «Інтерактивний інспектор»

Network Graph Parameters

Back to Backlog

Backlog

Event	Start of the event execution (TEarly)	End of the event execution (TLate)	Time Reserve
1	0	0	0
2	8	8	0
3	15	15	0
4	15	20	5
5	20	20	0
6	20	20	0
7	26	26	0
8	31	31	0
9	37	37	0
10	44	44	0
11	48	48	0
12	53	53	0
13	48	53	5
14	53	53	0
15	62	62	0
16	69	69	0
17	77	77	0

Рисунок 5.11 – Час виконання подій та резерви подій ПП

На основі одержаних даних було обчислено критично важливі завдання процесу розробки ПП, а також часові резерви для деяких завдань (рисунок 5.12), на рисунку 5.13 графічно продемонстровані результати обчислення критично важливих показників розробки програмного продукту у вигляді графа.

Network Graph

Number of User Story (US)	Description	Start event(i)	End event(j)	Time reserve
1	Аналіз нормативно-правових документів (ДБН)	1	2	0
2	Консолідація нормативно-правових документів	2	3	0
3	Планування архітектури чат-бота	2	4	5
4	Вибір інструментів розробки (Java, Spring, Spring JPA, Telegram API, PostgreSQL)	3	5	0
0		4	6	5
0		5	6	0
5	Проектування архітектури програми	6	7	0
6	Проектування БД	7	8	0
7	Створення схеми, таблиць БД	8	9	0
8	Створення проекту, основних класів, підключення Spring	9	10	0
9	Налаштуваннях з'єднання з БД Postgres за допомогою Spring JPA	10	11	0
10	Реалізація методів для виконання CRUD операцій з БД	11	12	0
11	Створення інтерфейсу для взаємодії з користувачем через Telegram API	10	13	5
0		11	14	5
0		12	14	0
0		13	14	5
12	Реалізація бізнес-логіки для автоматизації процесів оцінки протипожежного стану об'єктів	14	15	0
13	Виявлення помилок та оптимізація роботи чат бота	15	16	0
14	Підтримка системи	16	17	0

Рисунок 5.12 – Беклог спринта та часові резерви

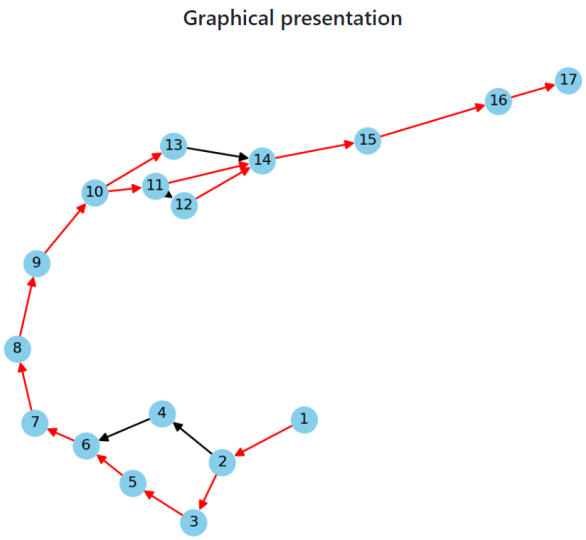


Рисунок 5.13 – Графічно представлені результати обчислення критично важливих показників розробки програмного продукту

Як висновок, команда розробників отримала готовий план, а також могла оцінити, чи зможе виконати його за один спринт, чи потрібно вносити зміни. Варто зазначити, що всі команди розробників, які працювали на даних проєктах є унікальними і, відповідно, кількість сторі пойнтів, які вони можуть взяти на один спринт для них будуть різними.

5.2.2. Веб-сервіс обліку пунктів вакцинації та тестування на COVID-19

Дана програмна система була розроблена у 2021 році студентами спеціальності «Комп'ютерні науки» під менторством здобувача Кордунової Ю.С. Цей R&D проєкт спрямований на систематизацію та облік пунктів вакцинації та тестування на COVID-19.

Веб-застосунок PCR-Surfer спеціалізується виключно на медичних послугах, які є актуальними для боротьби із COVID-19. Це робить його універсальним та вкрай актуальним в часи боротьби із пандемією.

PCR-Surfer має зручний інтерфейс користувача (рисунок 5.14) та велику кількість можливостей.

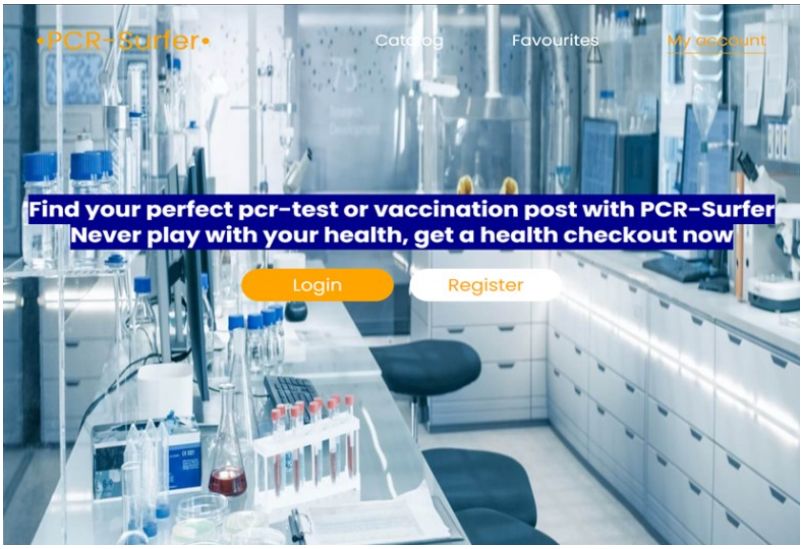
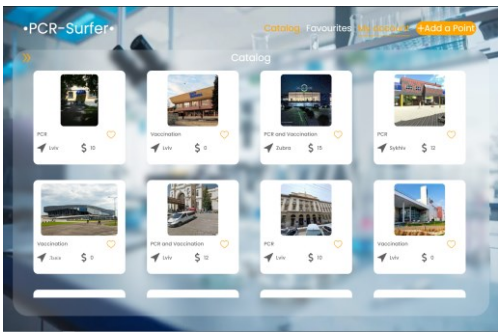


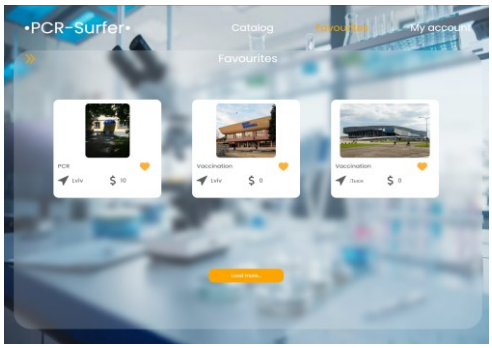
Рисунок 5.14 – Інтерфейс програмної системи PCR-Surfer

Зокрема, серед доступних можливостей користувача є:

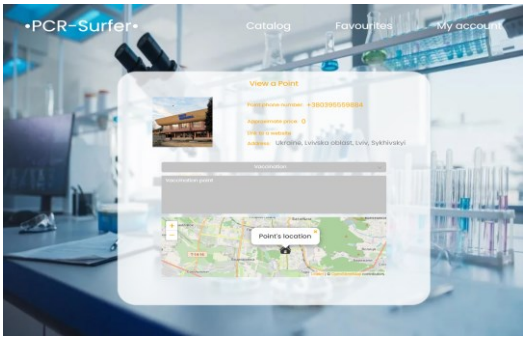
- перегляд великого каталогу пунктів вакцинації та ПЛР-тестування, який постійно оновлюється (рисунок 5.15 а);
- додавання пунктів до уподобаних, з подальшим переглядом (рисунок 5.15 б);
- перегляд точної інформації про пункт: адреса, розташування, номер телефону, опис, посилання на веб-сторінку пункту, ціна, тощо (рисунок 5.15 в);
- власний профіль користувача (рисунок 5.15 г).



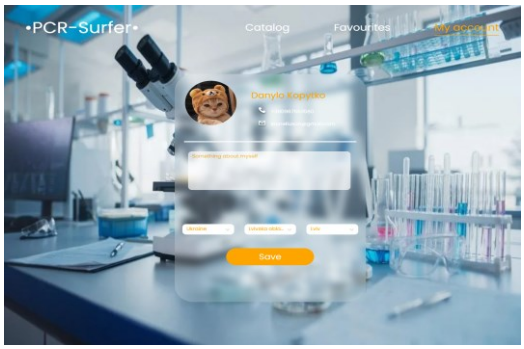
а) – перегляд каталогу



б) додавання до улюблених



в) – детальна інформація про пункт



г) – профіль користувача

Рисунок 5.15 – Функціонал програмної системи PCR-Surfer

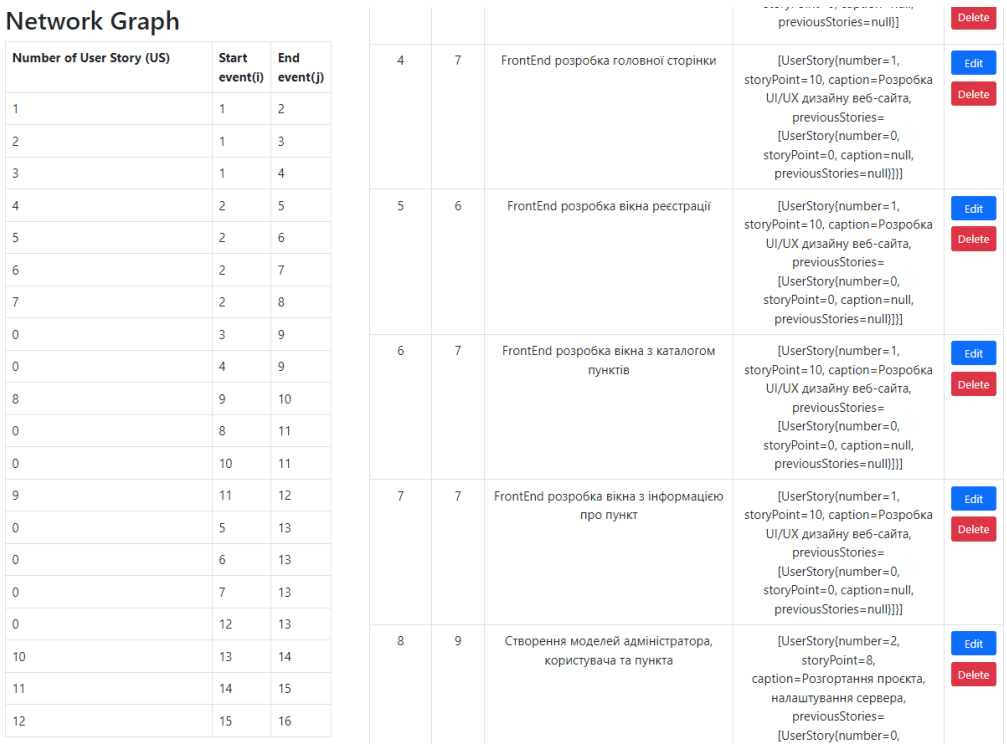
Переваги для компаній, які надають послуги в даній сфері:

- додаткова реклама послуги;
- зручний інтерфейс;
- пряме посилання на сайт, що в свою чергу збільшує конверсію.

Особливо актуальним даний веб-застосунок був на початку війни у лютому-березні 2022 року. Враховуючи велику кількість вимушено переміщених осіб ризик захворіти на COVID-19 був значно більший. Дана автоматизована система забезпечувала легкий та зручний пошук потрібного пункту, відслідковування його на карті та, за потреби, порівняння ціни на послугу у різних локаціях району перебування.

Наразі підтримка цього застосунку не відбувається, проте переглянути інтерфейс веб-сайту можна за посиланням https://master.d2wa196ubyd6uf.amplifyapp.com/.

На рисунку 5.16 зображений мережевий граф беклогу завдань, занесених у розроблену ІСППР в УЖЦ СПЗ а на рисунку 5.17 – час виконання подій та резерви подій ПП.

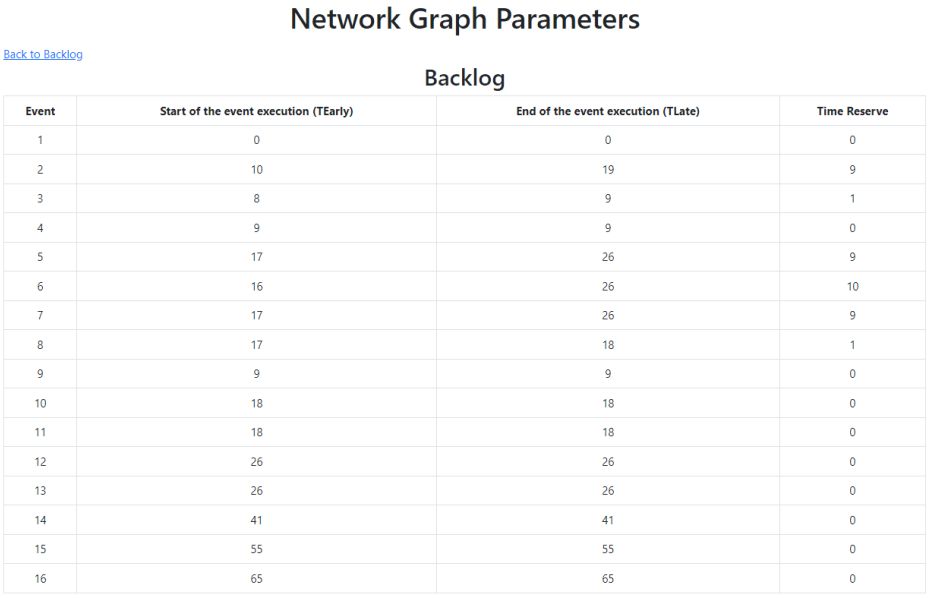


Network Graph

Number of User Story (US)	Start event(i)	End event(j)
1	1	2
2	1	3
3	1	4
4	2	5
5	2	6
6	2	7
7	2	8
0	3	9
0	4	9
8	9	10
0	8	11
0	10	11
9	11	12
0	5	13
0	6	13
0	7	13
0	12	13
10	13	14
11	14	15
12	15	16

			previousStories=null}]	Delete
4	7	FrontEnd розробка головної сторінки	[UserStory{number=1, storyPoint=10, caption=Розробка UI/UX дизайну веб-сайта, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}]	Edit Delete
5	6	FrontEnd розробка вікна реєстрації	[UserStory{number=1, storyPoint=10, caption=Розробка UI/UX дизайну веб-сайта, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}]	Edit Delete
6	7	FrontEnd розробка вікна з каталогом пунктів	[UserStory{number=1, storyPoint=10, caption=Розробка UI/UX дизайну веб-сайта, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}]	Edit Delete
7	7	FrontEnd розробка вікна з інформацією про пункт	[UserStory{number=1, storyPoint=10, caption=Розробка UI/UX дизайну веб-сайта, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}]	Edit Delete
8	9	Створення моделей адміністратора, користувача та пункта	[UserStory{number=2, storyPoint=8, caption=Розгортання проєкта, налаштування сервера, previousStories= [UserStory{number=0,	Edit Delete

Рисунок 5.16 – Аналітичне представлення мережевого графа



Network Graph Parameters

Back to Backlog

Backlog

Event	Start of the event execution (TEarly)	End of the event execution (TLate)	Time Reserve
1	0	0	0
2	10	19	9
3	8	9	1
4	9	9	0
5	17	26	9
6	16	26	10
7	17	26	9
8	17	18	1
9	9	9	0
10	18	18	0
11	18	18	0
12	26	26	0
13	26	26	0
14	41	41	0
15	55	55	0
16	65	65	0

Рисунок 5.17 – Час виконання подій та резерви подій ПП

На основі цих даних було обчислено критично важливі завдання процесу розробки даного ПП, а також часові резерви для деяких завдань (рисунок 5.18) та їх графічне представлення у вигляді графа (рисунок 5.19).

Network Graph

Number of User Story (US)	Description	Start event(i)	End event(j)	Time reserve
1	Розробка UI/UX дизайну веб-сайта	1	2	9
2	Розгортання проєкта, налаштування сервера	1	3	1
3	Проєктування бази даних	1	4	0
4	FrontEnd розробка головної сторінки	2	5	9
5	FrontEnd розробка вікна реєстрації	2	6	10
6	FrontEnd розробка вікна з каталогом пунктів	2	7	9
7	FrontEnd розробка вікна з інформацією про пункт	2	8	1
0		3	9	1
0		4	9	0
8	Створення моделей адміністратора, користувача та пункта	9	10	0
0		8	11	1
0		10	11	0
9	Підключення бібліотеки Leaflet для відображення карт (та маршутів) на сторінці "Інформація про пункт"	11	12	0
0		5	13	9
0		6	13	10
0		7	13	9
0		12	13	0
10	Реалізація CRUD методів для ролі Адміністратор при роботі із пунктами	13	14	0
11	Реалізація функціоналу для користувача: перегляд пункту, додавання до улюбленого, перегляд улюблених пунктів	14	15	0
12	Тестування та підтримка системи	15	16	0

Рисунок 5.18. – Беклог спринта та часові резерви

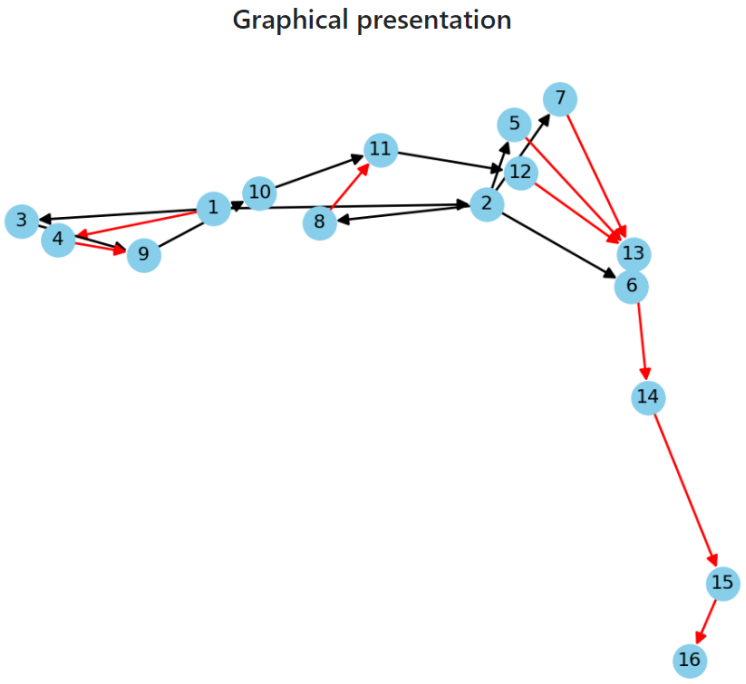


Рисунок 5.19 – Графічне представлення та критично важливі завдання проєкту

5.2.3. Інформаційна система «Я-Доброволець»

Інформаційна система «Я-Доброволець» спрямована на створення інноваційної платформи для організації та координування дій соціально активних громадян, які готові долучитись до ліквідації наслідків надзвичайних подій (на допомогу основним підрозділам).

Мета проєкту: створення безпекового середовища громади шляхом об’єднання зусиль добровольців, а також небайдужих громадян, які мають відповідний досвід та бажання допомогти.

Дана програмна система, наразі, на стадії розробки, тому ознайомитись із функціоналом практично, наразі немає можливості. На рисунку 5.20 можна переглянути інтерфейс системи, який наразі розробляється та на рисунку 5.21 представлене аналітичне зображення мережевого графа для окремого спринта проєктку. На рисунку 5.22 зображений час виконання подій та їх резерви, рисунок 5.23 містить беклог спринта та часові резерви, а на рисунку 5.24 графічно представлені критично важливі завдання проєкту.

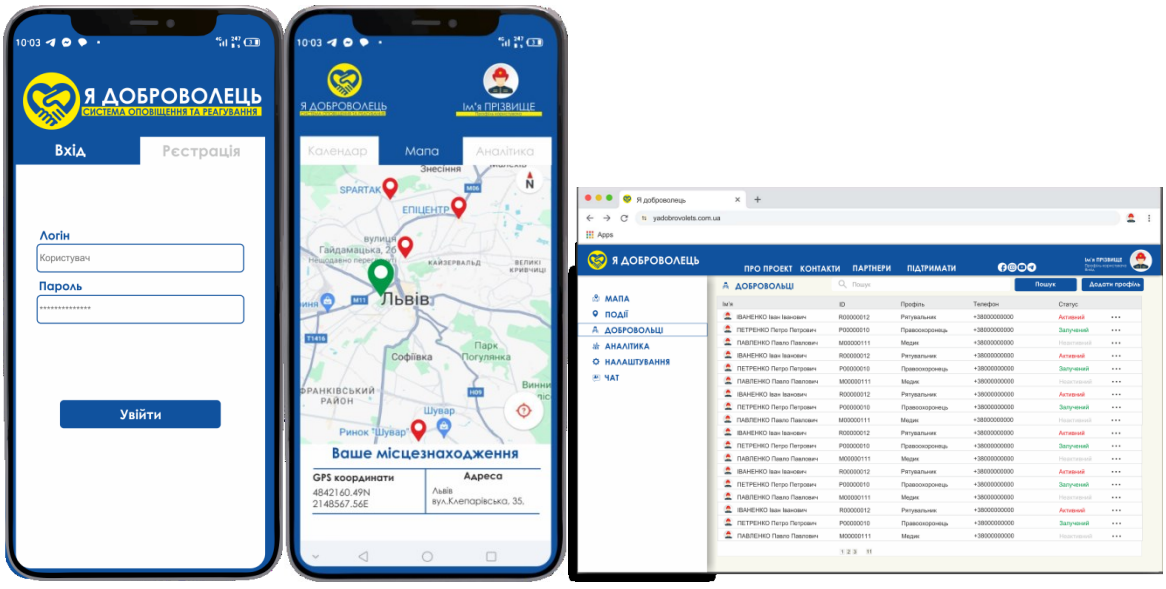


Рисунок 5.20 – Інтерфейс системи «Я-Доброволець»

Network Graph

Number of User Story (US)	Start event(i)	End event(j)
1	1	2
2	1	3
3	1	4
4	1	5
0	2	6
0	3	6
0	4	6
0	5	6
5	6	7
6	7	8
7	8	9
8	8	10
9	8	11
0	9	12
0	10	12
0	11	12
10	12	13
11	13	14
12	14	15

		проєкту	storyPoint=0, caption=null, previousStories=null}]	Delete
4	9	Розробити структуру БД для volunteer-core	[UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]	Edit Delete
5	10	Створити CRUD API для Volunteer, Event	[UserStory{number=1, storyPoint=8, caption=Створити і налаштувати репозиторії, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}, UserStory{number=2, storyPoint=8, caption=Створити скелетон проекту volunteer-core, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}, UserStory{number=3, storyPoint=8, caption=Створити документацію по Архітектурі проекту, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}, UserStory{number=4, storyPoint=9, caption=Розробити структуру БД для volunteer-core, previousStories= [UserStory{number=0, storyPoint=0, caption=null, previousStories=null}]}]	Edit Delete
6	9	Додати OpenAPI volunteer-core	[UserStory{number=5, storyPoint=10, caption=Створити CRUD API для Volunteer, Event,	Edit Delete

Рисунок 5.21 – Аналітичне представлення мережевого графа

Network Graph Parameters

Back to Backlog

Backlog

Event	Start of the event execution (TEarly)	End of the event execution (TLate)	Time Reserve
1	0	0	0
2	8	9	1
3	8	9	1
4	8	9	1
5	9	9	0
6	9	9	0
7	19	19	0
8	28	28	0
9	36	36	0
10	35	36	1
11	35	36	1
12	36	36	0
13	44	44	0
14	54	54	0
15	61	61	0

Рисунок 5.22 – Час виконання подій та резерви подій ПП

Network Graph

Number of User Story (US)	Description	Start event(i)	End event(j)	Time reserve
1	Створити і налаштувати репозиторії	1	2	1
2	Створити скелетон проекту volunteer-core	1	3	1
3	Створити документацію по Архітектурі проекту	1	4	1
4	Розробити структуру БД для volunteer-core	1	5	0
0		2	6	1
0		3	6	1
0		4	6	1
0		5	6	0
5	Створити CRUD API для Volunteer, Event	6	7	0
6	Додати OpenAPI volunteer-core	7	8	0
7	Реалізація процесу пошуку волонтерів для направлення на подію	8	9	0
8	Implement VolunteerRegistryService	8	10	1
9	Implement VolunteerRegistryClient	8	11	1
0		9	12	0
0		10	12	1
0		11	12	1
10	Implement VolunteerRegistryClient findById	12	13	0
11	Fix FirebaseInitializer volunteer-messaging-service	13	14	0
12	Підготувати контроллер для моб апки який буде повертати всю необхфдну інформацію про подію для волонтера	14	15	0

Рисунок 5.23 – Беклог спринта та часові резерви

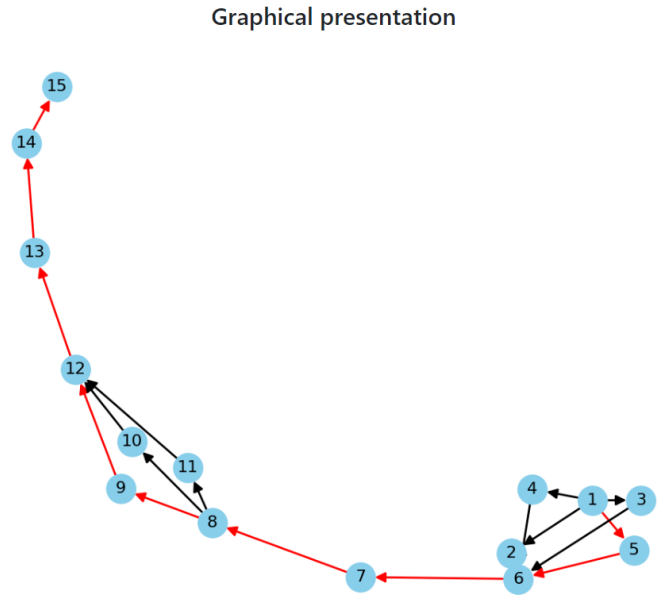


Рисунок 5.24 – Графічне представлення критично важливих завдань проєкту

5.3. Практичне впровадження

Апробацію розробленого методу управління життєвим циклом розроблення спеціалізованого програмного забезпечення проведено шляхом його впровадження у розробку безпеко-орієнтованих сервісів, описаних у розділі 5.2, та порівняння очікуваних значень (розрахованих у розділі 4) із фактичними результатами.

Для досліду було вибрано три команди розробників, які працювали над розробленням безпеко-орієнтованих сервісів на кафедрі інформаційних технологій та систем електронних комунікацій Львівського державного університету безпеки життєдіяльності.

Перша команда займалась розробленням інформаційно-аналітичної системи «Інтерактивний інспектор» та складалась із трьох осіб, досвід розробки у яких два, три та шість років, відповідно середній досвід команди складав 3,6 роки. Під час розробки команда зіткнулась із проблемою, яка полягала у зміні нормативно-правових актів, які впливали на обчислення даних у застосунку, відповідно прийшлось вносити зміни під час розробки системи. Таким чином, кількість змін, яку необхідно було ввести у розробку становила 5, а це 0,33% від загальної кількості завдань. (На рисунку 5.11 зображено беклог продукту, відповідно зміна у нормативно-правових актах внесла свої корективи на користувацькі історії 2, 10, 12, 13, 15).

Якщо ввести ці дані як початкові для прогнозування тривалості виконання проекту у рівняння регресії (4.20), то отримаємо наступний результат: $T=77\ s.p.$.

Друга команда займалась розробкою веб-сервісу обліку пунктів вакцинації та тестування на COVID-19 та складалась із двох осіб, досвід розробки у яких читири, шість років, відповідно середній досвід команди складав п’ять років. Оскільки це був студентський R&D проєкт, то змін під час розробки практично не було. Основна задача – розробити працюючий

веб-сервіс. Проте, враховуючи, що неможливо наперед передбачити всі нюанси системи, врахуємо, що до проекту вносився 2% змін.

Якщо ввести ці дані, як початкові для прогнозування тривалості виконання проекту у рівняння регресії (4.20), то отримаємо наступний результат: *T = 65s.p.*

Третя команда займалась розробкою інформаційної системи «Я-доброволець» та складалась із чотирьох розробників, досвід яких 15, 10, 6 та 3 роки, відповідно середній вік команди складав 8,5 років. Середня кількість змін на цьому проекті становила приблизно 15% на спринт.

Якщо ввести ці дані, як початкові для прогнозування тривалості виконання проекту у рівняння регресії (4.20), то отримаємо наступний результат: *T = 61s.p.*

На основі реалізованих проєктів визначимо похибку, розробленого методу управління життєвим циклом спеціалізованого програмного забезпечення відносно обчисленого прогнозованого значення (таблиця 5.1).

Для цього скористаємось наступною формулою:

$$\Delta = \frac{i_{к(фактичне)} - i_{к(розраховане)}}{i_{к(фактичне)}} \cdot 100\%. \qquad (5.1)$$

Таблиця 5.1 – Похибка відповідності розрахунків фактичному значенню критерію ефективності

№ з/п	Значення i_k (фактичне)	Розподіл i_k (розрахований)	Δ_i, %	$\overline{\Delta}$, %
1.	77	78	1,2	2,5
2.	65	64	1,5	
3.	61	64	4,9	

На рисунку 5.25 зображено діаграму для візуального порівняння фактичного та прогнозованого результатів планування ІТ-проєкту із використанням лінійної регресії.

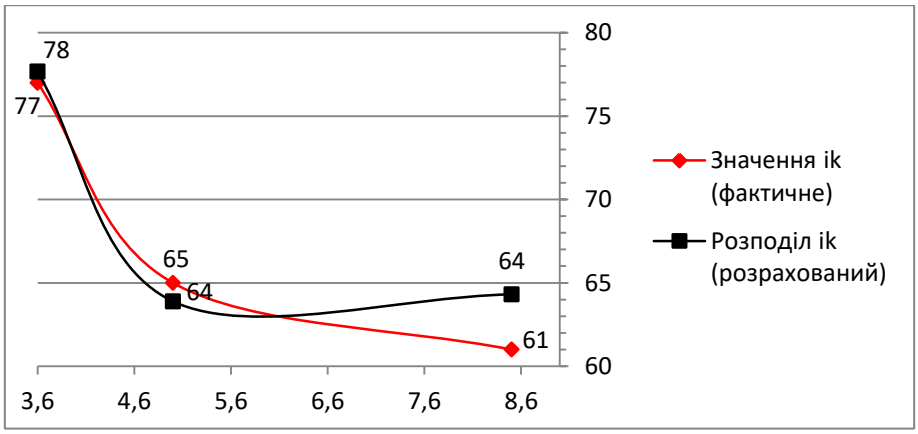


Рисунок 5.25 – Порівняльна залежність фактичного результату від прогнозованого

Згідно з проведеними розрахунками, відносна похибка вимірювань склала 4,9%. Це свідчить про те, що розроблений метод управління життєвим циклом програмного забезпечення працює ефективно та може застосовуватись при розробці безпеко-орієнтованих сервісів для Державної служби України із надзвичайних ситуацій.

Висновки до п’ятого розділу

У п’ятому розділі дисертаційного дослідження розроблено інформаційну систему підтримки прийняття рішень в управлінні життєвим циклом розроблення спеціалізованого програмного забезпечення шляхом впровадження у неї розроблених методів, моделей та алгоритмів для вирішення задачі короткострокового планування розробки безпеко-орієнтованих сервісів. Дана ІС здійснює автоматизацію процесів короткострокового планування, а також швидко і якісно переоцінює тривалість робіт за умови введення будь яких змін до змісту, обсягу, часу виконання окремих спринтів проєкту або складу команди розробників.

У розділі наведено також низку безпеко-орієнтованих сервісів, на яких було апробовано розроблену систему, а також обчислено похибку. Відповідно до проведених розрахунків відносна похибка вимірювань

склала 2,5%. Це свідчить про те, що розроблений метод, а також ІС управління життєвим циклом спеціалізованого програмного забезпечення працює ефективно та може застосовуватись при розробці безпеко-орієнтованих сервісів для Державної служби України із надзвичайних ситуацій.

ЗАГАЛЬНІ ВИСНОВКИ

У дисертаційній роботі на підставі проведених досліджень розв’язано важливу науково-прикладну задачу підвищення ефективності процесу планування розроблення спеціалізованого програмного забезпечення у динамічних умовах, яка базується на обґрунтованих моделях та інформаційній технології.

Основні результати дисертаційної роботи для науки та практики є такими:

1. У результаті проведеного аналізу предметної області та літературних джерел було обгрунтовано вплив гнучких методів на інновації спеціалізованого програмного забезпечення, систематизовано основні концепції методів гнучкої методології управління життєвим циклом програмних систем, розширено існуючі емпіричні дані про можливість та переваги використання гнучких методів у безпековій галузі, що зумовило передумови необхідності розроблення нових методів та моделей управлінням життєвим циклом розроблення спеціалізованого програмного забезпечення.

2. Шляхом моделювання процесу розробки програмних систем із використанням понятійного апарату теорії множин отримано математичний опис їх життєвого циклу, який дає змогу охарактеризувати обсяги робіт на окремих етапах розроблення програмного продукту, встановити взаємозв’язки і взаємозалежності між ними та сформувати фундаментальні принципи автоматизації процедури підтримки прийняття управлінських рішень на різних етапах розроблення програмних систем, зокрема, безпеко-орієнтованого спрямування в динамічному оточенні.

3. Розроблено концептуальну модель процесу управління життєвим циклом спеціалізованого програмного забезпечення (безпеко-орієнтованих сервісів), котра адаптована під специфіку роботи Державної служби України із надзвичайних ситуацій та корелює із принципами гнучкої методології управління життєвим циклом програмного забезпечення.

4. Шляхом імітаційного моделювання з використанням понятійного апарату мереж Петрі отримано уніфіковану модель процесів обходу мережевих графів планування, яка полягає у циклічності процедури встановлення зв’язку між попередньою і поточною подією та може використовуватися не залежно від складності та конфігурації мережевих графів.

5. За результатами оптимізації математичних методів мережевого планування під процеси управління життєвим циклом спеціалізованого програмного забезпечення, а також розроблених імітаційних моделей процесів обходу мережевого графа, розроблено алгоритми побудови та обходу графа мережевої моделі для визначення її основних параметрів, що надають можливість підвищити ефективність планування окремих етапів розробки програмного забезпечення та здійснювати їх коригування в режимі реального часу.

6. Розроблено інформаційну технологію підтримки прийняття рішень в процесі управління життєвим циклом розроблення спеціалізованого програмного забезпечення шляхом впровадження у неї розроблених моделей та алгоритмів для вирішення задачі короткострокового планування розробки безпеко-орієнтованих сервісів. Дана ІС здійснює автоматизацію процесів короткострокового планування, а також швидко і якісно переоцінює тривалість робіт за умови введення будь яких змін до змісту, обсягу, часу виконання окремих спринтів проєкту або складу команди розробників.

7. Наведено низку безпеко-орієнтованих сервісів, в процесі розроблення яких на яких було апробовано дану систему, а також обчислено похибку. Відповідно до проведених розрахунків, відносна похибка вимірювань склала 2,5%. Це свідчить про те, що розроблені моделі та інформаційна технологія управління життєвим циклом розроблення спеціалізованого програмного забезпечення працює ефективно та може застосовуватись при розробці безпеко-орієнтованих сервісів для Державної служби України із надзвичайних ситуацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Agile-маніфест розробки програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: https://agilemanifesto.org/iso/uk/manifesto.html.

2. Алексієв В. О. Визначення методів та засобів для уніфікації процесу розробки веб-ресурсів малого підприємства. *Матеріали VI міжнародної науково-технічної Інтернет-конференції "Автомобіль і електроніка. сучасні технології*", 19-20 листопада 2018 р. Харків. 2018. – С. 48-50.

3. Алексієв В. О., Труш О. М. Аналіз методів та оптимізація процесу розробки програмних продуктів. *Моделювання та інформаційні технології в науці, техніці та освіті: зб. наук. праць міжнар. наук.-практ. Internet-конф*. 21-22 лист. 2018 р. Харків, 2018. С. 45-51.

4. Асєєва А. В., Кулаковська І. В. Аналіз проблем вибору технології для розробки програмного забезпечення. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. Луцьк, 2019. №37. С. 10 – 18.

5. Близнюкова І. О., Семко С. Г., Кійко С. Г. Огляд сучасних методологій управління командами IT-проєктів. *Управління розвитком складних систем*. Київ, 2020. № 43. С. 60 – 66. https://doi.org/10.32347/2412-9933.2020.43.60-66

6. Бобрицька Г.С. Прикладне застосування теорії графів у різних сферах життя суспільства та окремої особистості. *Фізико-математична освіта*. 2017. В. 3(13). С. 26-30.

7. Богач І., Опольський Я. Застосування програмної бібліотеки машинного навчання TensorFlow для реалізації розпізнавання вмісту зображень на мобільних платформах : дис. – ВНТУ, 2018.

8. Бушуєв С.Д., Бушуєва В. Б., Бойко О. О. Agile- трансформація підходів в управлінні будівельними проектами, фазах ініціалізації та проєктування. *Управління розвитком складних систем*. Київ, 2020. №41. С. 14 – 20. https://doi.org/10.32347/2412-9933.2020.41.15-20

9. Вавіленкова А. І. Аналіз гнучких методологій розробки програмного забезпечення для реалізації у командних проєктах. *Вісник Національного технічного університету «ХПІ»*. Харків, 2021. № 1(7). С. 39 – 46. https://doi.org/10.20998/2413-4295.2021.01.06

10. Ванін В.В., Залевська О.В., Яблонський П.М. Застосування теорії графів для удосконалення та візуалізації алгоритму пошуку найкоротшого шляху в математичній моделі відео ігри. *Прикладна геометрія та інженерна графіка*, 2020 №97, с. 23-28

11. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Розробка архітектури програмного засобу для управління мережевим плануванням реінжинірингу програмного проекту. *Сучасний стан наукових досліджень та технологій в промисловості*. 2019. № 2 (8). С. 25–35. DOI: https://doi.org/10.30837/2522-9818.2019.8.025

12. Возняк А. Науково-дослідницька діяльність студентів як важливий напрям роботи з обдарованою молоддю. *Вища школа. Гуманізація навчально-виховного процесу*. 2011. №7. С. 77–82

13. Герасимчук О. Алгоритми розпізнавання символів для систем штучного інтелекту. *Матеріали X студентської науково-технічної конференції «Природничі та гуманітарні науки. Актуальні питання»*. 2007. С. 117-117.

14. Данченко О. Б. Методи та засоби аналізу проектних ризиків. *Вісник Черкаського державного технологічного університету* . Черкаси. 2004, № 1. – С. 87-92.

15. Димова Г.О., Ларченко О. В. Розробка комп’ютерної програми розв’язання задач мережевої оптимізації. *Науковий журнал "Комп’ютерно-інтегровані технології: освіта, наука, виробництво"*. Луцьк, 2020. Випуск № 41 С. 143 – 151

16. Єчина Ю. С. Науково-дослідницька діяльність студентів як підгрунтя науково-технічного розвитку. *Вісник КНУТД*. 2012. №5 С. 341-347.

17. Кім О. О., Козлова В. В. Перспективи застосування методології Agile менеджменту в управлінні ІТ-проектами. *Соціальна економіка*. Харків. 2019. № 58. С. 95 – 99. https://doi.org/10.26565/2524-2547-2019-58-12

18. Клочко, Н. Б., Чеховський, С. А., & Слабінога, М. О. Дослідження фізичної моделі процесу вимірювання об'єму газу турбінними лічильниками методом Монте-Карло. *Математичне моделювання*. №2, 2016. С. 77-79.

19. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення. *Системи управління, навігації та зв'язку*. Полтава, 2018. Т. 3(49). С. 116-125. https://doi.org/10.26906/SUNZ.2018.3.116

20. Коваль М. С., Литвин А. В. Завдання та властивості інформаційно-освітнього середовища закладу вищої освіти ДСНС України. *Модернізація змісту професійної освіти в умовах євроінтеграції України : матеріали Всеукраїнської науково-практичної конференції*. Київ, 2021. Ч.1. С. 39-43.

21. Ковальчук О. І., Зачко О. Б. Моделі життєвого циклу розвитку проєктних команд в системі цивільного захисту. *Вісник Львівського державного університету безпеки життєдіяльності*. Львів, 2022. №25. С. 71-78. https://doi.org/10.32447/20784643.25.2022.08

22. Козир І. С. Фактори впровадження Agile-менеджменту в практику управління. *I International Scientific and Practical Conference «Problemas y perspectivas de la aplicación de la investigación científica innovadora»*. Кембридж. 2021. Т. 1. С. 78-79. https://doi.org/10.36074/logos-19.03.2021.v1.26

23. Козяр М. М., Козловський Ю. М., Стечкевич О. О. Реалізація можливостей Stem-освіти засобами інтеграції креативних методів навчання. *Наукові записки. Педагогічні науки.* Кропивницький, 2020. № 191. С. 20-23.

24. Колянко О. В., Озимок Г. В., Використання жорсткої "Waterfall" та гнучкої "Agile" моделей управління проектами. *Вісник Львівського торговельно-економічного університету. Економічні науки*. Львів, 2017. Вип. 52. С. 177 – 182.

25. Кордунова Ю. С., Придатко О. В., Смотр О. О. Переваги використання Agile- методології під час розробки програмного забезпечення в умовах сучасного ринку. *Інформаційна безпека та інформаційні технології : зб. наук. праць IV Всеукр. наук.-практ. конф. молодих учених, студентів і курсантів.* м. Львів 27 листопада 2020 р. Львів, 2020. С. 206 – 207

26. Кордунова Ю. С., Смотр О. О. Визначення ефективності використання Agile методології в сучасних організаціях. *Проблеми та перспективи забезпечення цивільного захисту: матеріали міжнародної науково-практичної конференції молодих учених*. Харків: НУЦЗУ, 2021. С. 166.

27. Кордунова Ю. С., Смотр О. О. Сенс Agile-маніфесту для сучасного проєкт-менеджменту. *Проблеми та перспективи розвитку системи безпеки життєдіяльності: зб. наук. праць XVI Міжнар. наук.-практ. конф. молодих вче1них, курсантів та студентів*. – Львів: ЛДУ БЖД, 2021. С. 247-248

28. Кордунова Ю. С., Смотр О. О., Кокотко І. Я., Малець Р. Б. Аналіз традиційного та гнучкого підходів до створення програмного забезпечення в динамічних умовах. *Управління розвитком складних систем*. Київ, 2021. № 47. С. 71 – 77, https://doi.org/10.32347/2412-9933.2021.47.71-77

29. Кордунова Ю.С., Придатко О.В., Смотр О.О. Переваги використання Agile-методології під час розробки програмного забезпечення в умовах сучасного ринку. *Інформаційна безпека та інформаційні технології*, Вип.4, 206-207.

30. Кордунова, Ю., Фелтіновскі, М., Придатко, О., Смотр, О. Математичне моделювання процесу розробки спеціалізованих програмних систем безпеко-орієнтованого спрямування. *Вісник Львівського державного університету безпеки життєдіяльності*. Львів, 2023. № 27, С. 23-31. https://doi.org/https://doi.org/10.32447/20784643.27.2023.03

31. Кузь М. В, Пікуляк М. В. Остафійчук Т. Д. Модель системи управління якістю процесу розробки програмного забезпечення. *Proceedings of the 2019 Scientific Seminar on Innovative Solutions in Software Engineering*. Івано-Франківськ, 2019. С. 19-21, https://doi.org/10.5281/zenodo.4091480

32. Кузьменко А. І., Коциловський М. П. Удосконалення організації руху вантажопотоків у транспортній системі України. Системи та технології, № 2 (54), 2015 с 81 – 89

33. Кузьменок та ін. Алгоритми пошуку найкоротших шляхів та їх застосування в комп'ютерних іграх. Комп'ютерні системи та інформаційні технології. 2021. № 2. С. 78-84.

34. Латанська Л. О., Балашов М. О. Удосконалення регресійного рівняння для оцінювання трудомісткості розробки програмного забезпечення, створеного за гнучкою методологією. *Матеріали XI-ої Міжнародної науково-практичної конференції «Free and Open Source Software»*, Харків, 19-21 листопада 2019 р. – Харків: Харківський національний університет будівництва та архітектури, С. 46.

35. Муравецький С. А., Крамський С. О. Планування процесів забезпечення якості у великих та географічно розподілених гібридних IT-проектах. *Вісник НТУ «ХПІ»*. Харків, 2016. №1(1173). С. 106 – 109. https://doi.org/10.20998/2413-3000.2016.1173.21

36. Павлова О.О., Лопатто І.Ю., Говорущенко Т.О. Метод діяльності та структура інтелектуального агента на основі онтологічного підходу для оцінювання початкових етапів життєвого циклу програмного

забезпечення. *Вісник Хмельницького національного університету*, Хмельницький, 2020. №3. С.61-64

37. Пасічник А. М., Кутирєв В. В., Бугерко К. М. Розрахунок максимального потоку вантажів у транспортній мережі шляхом застосування алгоритму Форда–Фалкерсона. *Вісник АМСУ. Серія: "Технічні науки"*, № 2 (52), 2014 с 18 – 31

38. Праворська Н., Мартинюк В. Конструювання програмного забезпечення за допомогою синхронного підходу: основні процеси та інструменти для ефективної реалізації Devops. *Вісник Хмельницького національного університету*, Том 1, №5, 2023 С. 182-191

39. Придатко О. В., Кордунова Ю.С, Кокотко І. Я., Головатий Р. Р. Обгрунтування методології управління студентськими R&D проєктами (на прикладі освітньої програми комп'ютерні науки) *"Інформаційно-комунікаційні технології в сучасній освіті: досвід, проблеми, перспективи"* Львів, 2021.

40. Придатко О. В., Придатко В. В., Борзов Ю. О., Дзень В. Є. Інтеграція новаційного методу мобільного навчання в освітні проекти підготовки розробників програмного забезпечення. *Вісник ЛДУБЖД*, Львів: ЛДУ БЖД, 2018. – №18. – С.70-80.

41. Приймак В. Гнучкі моделі управління командною роботою інжинірингових проектів. *Вісник Київського національного університету мені Тараса Шевченка. Економіка*. Київ, 2019. №6 (207). С. 21-27. https://doi.org/10.17721/1728-2667.2019/207-6/3

42. Прошкін В. В. Стимулювання студентських наукових пошуків як засіб інтеграції науки й освіти. *Професійна освіта. Педагогіка*. 2010. № 1. С.39 – 44.

43. Семенов С. Г., Халифе Кассем, Захарченко М. М. Удосконалений спосіб масштабування гнучкої методології розробки програмного забезпечення. . Вісник НТУ «ХПІ». Харків, 2017. Т. 1, № 1. С. 79 – 84. https://doi.org/10.20998/2522-9052.2017.1.15

44. Сєдих О.Л., Чобану В.В. Оптимізація мережевого графіка комплексу робіт. *Modern engineering and innovative technologies.* № 03-01, 2018 С. 61-67 https://doi.org/10.30890/2567-5273.2018-03-01-009 10.

45. Сидорчук Н.Г. До питання про організацію науково-дослідної роботи студентів педагогічних навчальних закладів. *Зб.наук.пр. Сучасні інформаційні технології та інноваційні методики навчання у підготовці фахівців: методологія, теорія, досвід, проблеми*. Київ-Вінниця, 2002. С. 408-413.

46. Студентський R&D проєкт [Електронний ресурс] Режим доступу: https://www.slideshare.net/GlobalLogicUkraine/rd-236853435

47. Теорія планування експериментів: Виконання розрахунково-графічної роботи [Електронний ресурс] : навч. посіб. для студ. спеціальності 131 «Прикладна механіка», спеціалізації «Технологія машинобудування» – Київ : КПІ ім. Ігоря Сікорського,2020. – 86 с.

48. Токарєв В. В., Славтіч Д. О. Застосування алгоритма Дейкстри при проектуванні «розумних зупинок». *Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення: збірник тез доповідей*, 16 лист. 2020р, Тернопіль, 2020. В. 53. Ч.1. С.100- 101.

49. Шапошнікова О.П., Кірвас В.В. Застосування методології Agile в практиці проектного навчання при підготовці ІТ спеціалістів. *Системи обробки інформації*. Харків. 2020. № 4(163). С. 94-100. https://doi.org/10.30748/soi.2020.163.10

50. Якубенко І. М. Agile-менеджмент, як дієве управління проектами для цілеспрямованих команд. *Економіка. Менеджмент. Бізнес*. 2017. №4(22). С. 167 – 172.

51. A guide to the Project Management Body of Knowledge. PMBOK guide SIXTH EDITION – USA: Project Management Institute, 2017.

52. Adhikary, S. Images Within Images? A Multi-image Paradigm with Novel Key-Value Graph Oriented Steganography. *Intelligent Computing &*

Optimization. ICO 2021. Lecture Notes in Networks and Systems, 2022. vol 371. Springer, Cham. https://doi.org/10.1007/978-3-030-93247-3_83

53. Anand R. Vijay, Dinakaran M. Improved scrum method through staging priority and cyclomatic complexity to enhance software process and quality. *International Journal of Internet Technology and Secured Transactions*, 8 (2), 2018. p.p. 150-166. https://doi.org/10.1504/IJITST.2018.093342

54. Avasthi A. & Mishra G. A New Framework for the Agile Software Development Method. *Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, p.p. 436-438, https://doi.org/10.1109/ICECA.2018.8474737.

55. Barbareschi M., Barone S., Carbone R. et al. Scrum for safety: an agile methodology for safety-critical software systems. *Software Qual J.* 2022. №30, pp. 1067–1088 https://doi.org/10.1007/s11219-022-09593-2

56. Barraood S. O., Mohd H., Baharom F. A Comparison Study of Software Testing Activities in Agile Methods. *Knowledge Management International Conference (KMICe) Virtual Conference*. Malaysia, 2021. pp. 130-137

57. Belkasmi, M.G., Bougroun, Z., Farissi, I.E., Emharraf, M., Belouali, S., Chadli, S., Saber, M.: Global IT project management: An agile planning assistance. *Advances in Smart Technologies Applications and Case Studies*, pp. 575–582. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-53187-4_63

58. Benedicenti L., Messina A. & Sillitti A., IAgile: Mission Critical Military Software Development. *International Conference on High Performance Computing & Simulation (HPCS)*, 2017. p.p. 545-552, https://doi.org/10.1109/HPCS.2017.87

59. Benedicenti, L., Cotugno, F., Ciancarini, P., Messina, A., Pedrycz, W., Sillitti, A., Succi, G. Applying scrum to the army: a case study. *International Conference on Software Engineering Companion IEEE*, 2016. p.p. 725-727. https://doi.org/10.1145/2889160.2892652

60. Bertling M., Caroli H., Dannapfel M., Burggraf P. The minimal viable production system (mvps) – an approach for agile (automotive) factory planning in a disruptive environment. *Advances in Production Research. Springer International Publishing.* 2019. pp. 24–33, https://doi.org/10.1007/978-3-030-03451-1_3

61. Boral, S. Domain V: Adaptive Planning. *Ace the PMI-ACP® exam. Apress, Berkeley*, 2016. CA. https://doi.org/10.1007/978-1-4842-2526-4_6

62. Carbone, R., Barone, S., Barbareschi & M., Casola, V. Scrum for Safety: Agile Development in Safety-Critical Software Systems. Quality of Information and Communications Technology. *Communications in Computer and Information Science*, 2021. P. 1439. https://doi.org/10.1007/978-3-030-85347-1_10

63. Casola, V., De Benedictis, A., Rak, M., Villano, U. A novel security-by-design methodology: Modeling and assessing security by slas with a quantitative approach. *Journal of Systems and Software*, 2020. P. 163. https://doi.org/10.1016/j.jss.2020.110537

64. Cawley, O., Wang, X., Richardson, I. Lean/agile software development methodologies in regulated environments-state of the art. *Lean Enterprise Software and Systems. LESS 2010. Lecture Notes in Business Information Processing*, 2010. P. 65. https://doi.org/10.1007/978-3-642-16416-3_4

65. Cole R., Scotcher E. Brilliant Agile Project Management: A Practical Guide to Using Agile, Scrum and Kanban. Edinburg: Pearson, 2015. 187 p.

66. Cotugno, F.R., Messina, A. Adapting SCRUM to the Italian Army: Methods and (Open) Tools. Open Source Software: Mobile Open Source Technologies. *OSS 2014. IFIP Advances in Information and Communication Technology*, 2014. P. 427. https://doi.org/10.1007/978-3-642-55128-4_7

67. De Sá, F.R., Vieira, R.G., da Cunha, A.M. Lessons Learned from the Agile Transformation of an Aeronautics Computing Center. Agile Methods. *WBMA 2019. Communications in Computer and Information Science*, 2019. P. 1106. https://doi.org/10.1007/978-3-030-36701-5_7

68. Dur´an, M., Ju´arez-Ram´ırez, R., Jim´enez, S., Tona, C. User story estimation based on the complexity decomposition using bayesian networks. *Programming and Computer Software* 46, 2020. p.p. 569–583 https://doi.org/10.1134/S0361768820080095

69. Dymova, H., Larchenko, O.: Development of a computer program for solving network optimization problems. *Computer-integrated technologies: education, science, production.* 41, 2020. p.p. 143–151 https://doi.org/10.36910/6775-2524-0560-2020-41-23

70. Edison H., Wang X. and Conboy K. (2022) Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. Transactions on Software Engineering, 48(8), 2020. p.p. 2709-2731. https://doi.org/10.1109/TSE.2021.3069039

71. Gozhyj A., Kalinina I., Gozhyj V., Vysotska V., Web Service Interaction Modeling with Colored Petri Nets. *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Metz, France, 2019, pp. 319-323, doi: 10.1109/IDAACS.2019.8924400.

72. Gozhyj, A., Kalinina, I., Gozhyj, V., Danilov, V. Approach for Modeling Search Web-Services Based on Color Petri Nets. *Data Stream Mining & Processing. DSMP 2020. Communications in Computer and Information Science*, vol 1158, 2020. Springer, Cham, https://doi.org/10.1007/978-3-030-61656-4_35

73. Guidance on the use of AGILE practices in the development of medical device software (2012). Retrieved from: https://webstore.ansi.org/standards/aami/aamitir452012r2018

74. Hajou, A., Batenburg, R., Jansen, S. How the pharmaceutical industry and agile software development methods conflict: A systematic literature review. *International Conference on Computational Science and Its Applications IEEE*, 2014. p.p. 40-48, https://doi.org/10.1109/ICCSA.2014.19

75. Hallmann, D. ”i don’t understand!”: Toward a model to evaluate the role of user story quality. *Agile Processes in Software Engineering and Extreme Programming*. 2020. pp. 103–112. Springer International Publishing https://doi.org/10.1007/978-3-030-49392-9_7

76. Hanssen, G., Stålhane, T., & Myklebust, T. SafeScrum – Agile Development of Safety-Critical Software. New York: Springer. 2020.

77. Hovorushchenko T., Herts A.; Hnatchuk Y. Concept of Intelligent Decision Support System in the Legal Regulation of the Surrogate Motherhood. *International Workshop on Informatics & Data-Driven Medicine*, 2019, pp: 57-68.

78. Islam G., Stoner T. A case study of agile software development for safety-Critical systems projects. *Reliability Engineering & System Safety*. Vol. 200. 2020. https://doi.org/10.1016/j.ress.2020.106954

79. ISO/IEC 12207:2008 «Systems and software engineering — Software life cycle processes» [Електронний ресурс] – Режим доступу до ресурсу: https://standards.ieee.org/ieee/12207/5672/

80. Jain P., Sharma A., Ahuja L. The Impact of Agile Software Development Process on the Quality of Software Product. *International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. Noida, India, 2018, pp. 812-815, https://doi.org/10.1109/ICRITO.2018.8748529

81. Jansen U., Schulz W. FlowChart Tool for Decision Making in Interdisciplinary Research Cooperation. Digital Human Modeling. Applications in Health, Safety, Ergonomics, and Risk Management: Health and Safety. *Lecture Notes in Computer Science*, 2017. vol 10287. Springer, Cham. https://doi.org/10.1007/978-3-319-58466-9_24

82. Karras, O., Kl¨under, J., Schneider, K.: Is task board customization beneficial? *International Conference on Product-Focused Software Process Improvement*. 2017. pp. 3–18. Springer https://doi.org/10.1007/978-3-319-69926-4_1

83. Khmel, M., Prydatko, O., Popovych, V., Tkachenko, T., Kovalchuk, V.: Students r&d projects as a tool for achieving program competencies. *Information and communication technologies in modern education: experience, problems, prospects*. 2021.

84. Kordunova Y., Prydatko O., Smotr O., Golovatyi R. Expert Decision Support System Modeling in Lifecycle Management of Specialized Software. *Lecture Notes on Data Engineering and Communications Technologies*, *Springer*, Switzerland. Vol. 149, 2022, pp. 367-383, https://doi.org/10.1007/978-3-031-16203-9_22

85. Kordunova Yu., Prydatko O., Smotr O., Kokotko I. The network graph traversal method for solving the problem of short-term planning of safety-oriented services development. Monografia powstała w ramach Projektu dofinansowanego przez Ministra Edukacji i Nauki ze środków budżetu państwa w ramach programu „Doskonała Nauka", Warszawa 2022. p. 172 – 181.

86. Kovalchuk O., Zachko O. and Kobylkin D. Criteria for intellectual forming a project teams in safety oriented system. *IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2022, pp. 430-433

87. Kovalchuk O., Kobylkin D. and Zachko O. HR Decision-Making Support System Based On The CBR Method. *IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*, Lviv, Ukraine, 2023, pp. 1-4, doi: 10.1109/CSIT61576.2023.10324169.

88. Kuhrmann M. et al., Hybrid Software Development Approaches in Practice: *A European Perspective. IEEE Software.* 2019. vol. 36, no. 4, pp. 20-31, https://doi.org/10.1109/MS.2018.110161245

89. Kula E., Greuter E., Deursen A., Gousios G. Factors Affecting On-Time Delivery in Large-Scale Agile Software Development. *IEEE Transactions on Software Engineering*, vol. 48, no. 9. 2022. pp. 3573-3592. https://doi.org/10.1109/TSE.2021.3101192.

90. Lamsellak H, Belkasm M. G. Global software development agile planning model:challenges and current trends. Indonesian Journal of Electrical Engineering and Computer Science, Vol. 32, No. 3. 2023. pp. 1774-1784

91. Lamsellak H., Khalil A., Belkasmi M.G., Saber M. Agile Practices in Iteration Planning Process of Global Software Development. International Conference on Advanced Intelligent Systems for Sustainable Development. AI2SD 2022. Lecture Notes in Networks and Systems, vol 712, 2023. Springer, Cham, https://doi.org/10.1007/978-3-031-35251-5_29

92. Lenarduzzi V., Lunesu I., Matta M., Taibi D. Functional Size Measures and Effort Estimation in Agile Development: A Replicated Study. Agile Processes in Software Engineering and Extreme Programming. XP 2015. *Lecture Notes in Business Information Processing*, 2015. vol 212. https://doi.org/10.1007/978-3-319-18612-2_9

93. Liaskovska, S., Martyn, Y., Malets, I., Prydatko, O.: Information technology of process modeling in the multiparameter systems. *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP),* Lviv, Ukraine, 2018, pp. 177–182 (08 2018). https://doi.org/10.1109/DSMP.2018.8478498

94. Malets I., Prydatko O., Popovych V., Dominik A. Interactive Computer Simulators in Rescuer Training and Research of their Optimal Use Indicator. *2018 IEEE Second Conference on Data Stream Mining & Processing*. Lviv. №2. 2018. p.p. 558-562.

95. Martyn, Y., Smotr, O., Burak, N., Prydatko, O., Malets, I.: Software for shelter's fire safety and comfort levels evaluation. *Communications in Computer and Information Science*. 2020. pp. 457–469. Springer International Publishing https://doi.org/10.1007/978-3-030-61656-4_31

96. Martyn Ye., Liaskovska S., Gregus M., Izonin I., Velyka O. Optimization of Technological's Processes Industry 4.0 Parameters for Details *Manufacturing via Stamping: Rules of Queuing Systems. Procedia Computer Science*. 2021. Vol. 191, 290-295

97. McCaffery, F., Trektere, K., Ozcan-Top, O. Agile – Is it Suitable for Medical Device Software Development? *Software Process Improvement and Capability Determination*, 2016. P. 609. https://doi.org/10.1007/978-3-319-38980-6_30

98. Meier, A., Ivarsson, J.C.: Agile software development and service science. GSTF *Journal on Computing (JoC)* 3(3), 2013. p.p. 1–5 https://doi.org/10.7603/s40601-013-0029-6

99. Messina, A., Fiore, F., Ruggiero, M., Ciancarini, P., Russo, D. (2016). A new agile paradigm for mission-critical software development. *CrossTalk*. 29, 2016. p.p. 25-30.

100. Notander, J. P., Runeson, P., Höst, M. A model-based framework for flexible safety-critical software development: a design study. *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. 2013. p.p. 1137-1144, https://doi.org/10.1145/2480362.2480575

101. Paez N., Fontdevila D., Oliveros A. On the Influence of Agile in the Usage of Software Development Practices. I*EEE Congreso Bienal de Argentina (ARGENCON),* Resistencia Argentina. 2020. p.p. 1-7, https://doi.org/10.1109/ARGENCON49523.2020.9505407.

102. Papadopoulos G. Moving from traditional to agile software development methodologies also on large, distributed projects. *Procedia – Social and Behavioral Sciences.* 2015. № 175. pp. 455 – 463. https://doi.org/10.1016/j.sbspro.2015.01.1223

103. Prydatko, O., Popovych, V., Malets, I., Solotvinskyi, I.: Algorithm of rescue units logistic support planning in the process of regional life safety systems development. *MATEC Web of Conferences*. 294, 04002 (01 2019). https://doi.org/10.1051/matecconf/201929404002

104. Rajkumar S., Pradeep K., Partha P. R., Umapada P. Modeling local and global behavior for trajectory classification using graph-based algorithm. *Pattern Recognition Letters*. 2021, volume 150, pp 280-288

105. Ribeiro S., Schmitz E., Alencar A., Silva M. Literature Review on the Theory of Constraints Applied in the Software Development Process. *IEEE Latin America Transactions*. 2018. vol. 16, no. 11, pp. 2747-2756, https://doi.org/10.1109/TLA.2018.8795116

106. Sachdeva V. Requirements Prioritization in Agile: Use of Planning Poker for Maximizing Return on Investment. Information Technology - New Generations. *Advances in Intelligent Systems and Computing*, 2017. vol 558. Springer, Cham. https://doi.org/10.1007/978-3-319-54978-1_53

107. Schwaber, K., & Sutherland, J. (2020). The scrum guide (МетодР16)

108. Seidykh, O., Chobanu, V.: Optomozation of the network graphics of the complex of works. *Modern engineering and innovative technologi*. 2018. 1(3), p.p. 61–67. https://doi.org/10.30890/2567-5273.2018-03-01-009

109. Singh B. and Gautam S. The Impact of Software Development Process on Software Quality: A Review. *8th International Conference on Computational Intelligence and Communication Networks (CICN)*. Tehri, India, 2016. pp. 666-672, https://doi.org/10.1109/CICN.2016.137

110. Smith J., Bradbury J., Hayes W., Deadrick W. Agile Approach to Assuring the Safety-Critical Embedded Software for NASA's Orion Spacecraft. *IEEE Aerospace Conference, Big Sky*, 2019. p.p. 1-10, https://doi.org/10.1109/AERO.2019.8742095.

111. Steghöfer, JP., Knauss, E., Horkoff, J., Wohlrab, R. Challenges of Scaled Agile for Safety-Critical Systems. In: Franch, X., Männistö, T., Martínez-Fernández, S. (eds) Product-Focused Software Process Improvement. PROFES 2019. *Lecture Notes in Computer Science*, 2019. P. 11915. Springer, Cham. https://doi.org/10.1007/978-3-030-35333-9_26

112. Stellman A., Greene J. Learning Agile: Understanding Scrum, XP, Lean, and Kanban. 1st Edition, USA: O'Reilly Media, 2013. 420 c.

113. Stioca M., Ghlic-Micu B., Mircea M., Uscatu C. Analyzing Agile Development – from Waterfall Style to Scrumban. *Informatica Economică*. 2016. №4. С. 5–14. https://doi.org/10.12948/issn14531305/20.4.2016.01

114. Thesing T., Feldmann C., Burchardt M. Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project. *Procedia Computer Science*. 2021. № 181. pp. 746 – 756.

115. Tordesillas, J., Lopez, B.T., Carter, J., Ware, J., How, J.P.: Real-time planning with multi-fidelity models for agile flights in unknown environments. 2019. pp. 725–731 https://doi.org/10.1109/ICRA.2019.8794248

116. Uzun, I., Lobachev, I., Gall, L., Kharchenko, V.: Agile Architectural Model for Development of Time-Series Forecasting as a Service Applications. 2021. pp. 128–147. Springer International Publishing https://doi.org/10.1007/978-3-030-82014-5_9

117. Velykodniy, S., Burlachenko, Z., Zaitseva-Velykodna, S.: Architecture development of software for managing network planning of software project reengineering. *Innovative technologies and scientific solutions for industries*. 2019. № 2(8), p.p. 25–35 https://doi.org/10.30837/2522-9818.2019.8.025

118. Wang Linlin, Fang Xiaoyu, Hong Tao, Liu Chang, Liu Shilan Image Recognition Based on the Depth-Wise Separable Convolution and Softpool. *International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, 2022. pp.147-152

119. Wang H., Ma Z. Application and Improvement of Agile Development in Intelligent Health Hut Software Project. *International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS)*, 2023. p.p. 14-18, https://doi.org/10.1109/ICMSS56787.2023.10118316.

120. Zachko O. B., Kobylkin D. S. Structural Models of Safety-Oriented Management of Infrastructure Projects Decomposition. *International Scientific and Technical Conference on Computer Sciences and Information Technologies*. 2020. № 2. P. 131–134.

121. Zachko O. B., Kovalchuk O. I. Models of the life cycle of forming project teams in a security-oriented system. *International Scientific and Technical Conference on Computer Sciences and Information Technologies*, 2020. Vol. 1, P. 235-239.

ДОДАТКИ

Додаток А. Графічне представлення алгоритмів побудови та обходу мережевого графа

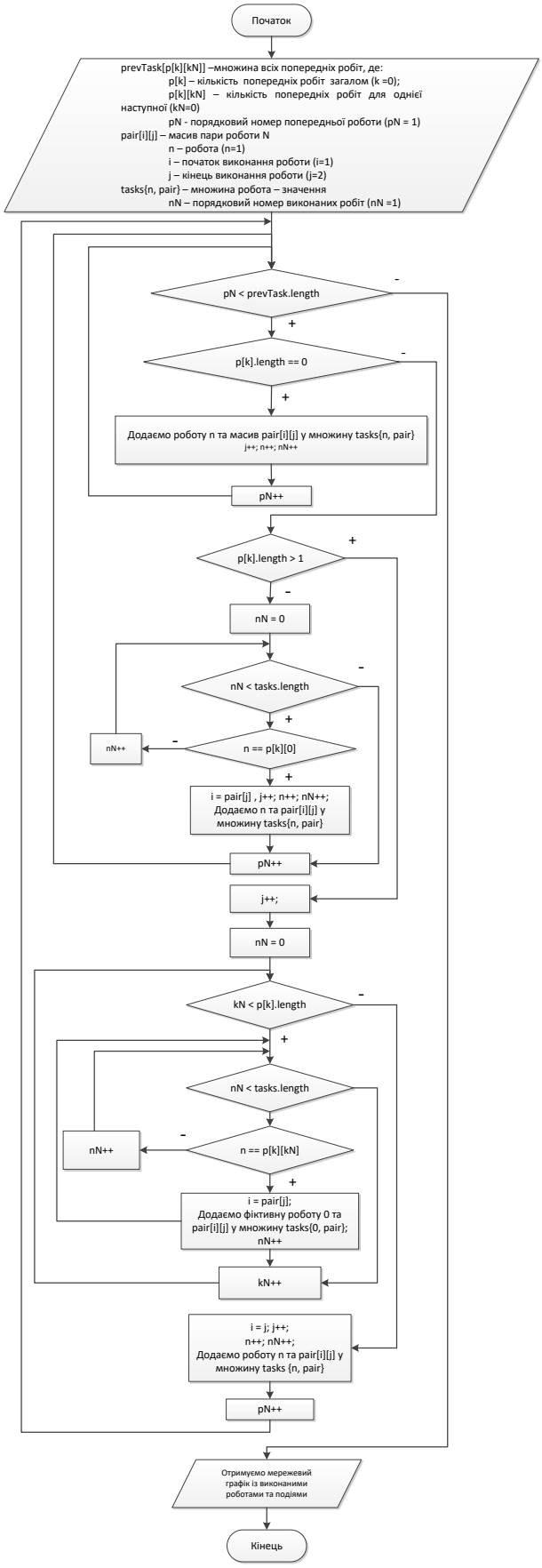


Рисунок 1 – Алгоритм побудови мережевого графа

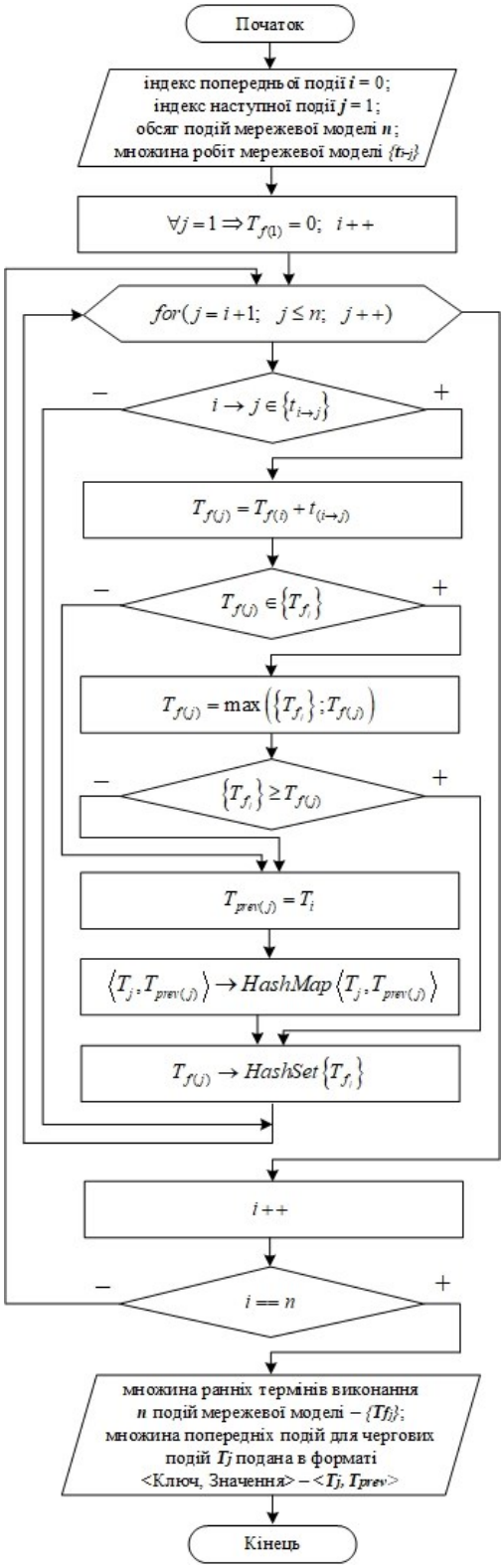


Рисунок 2 – Алгоритм обходу графа мережевої моделі задля визначення показника $T_{f(j)}$

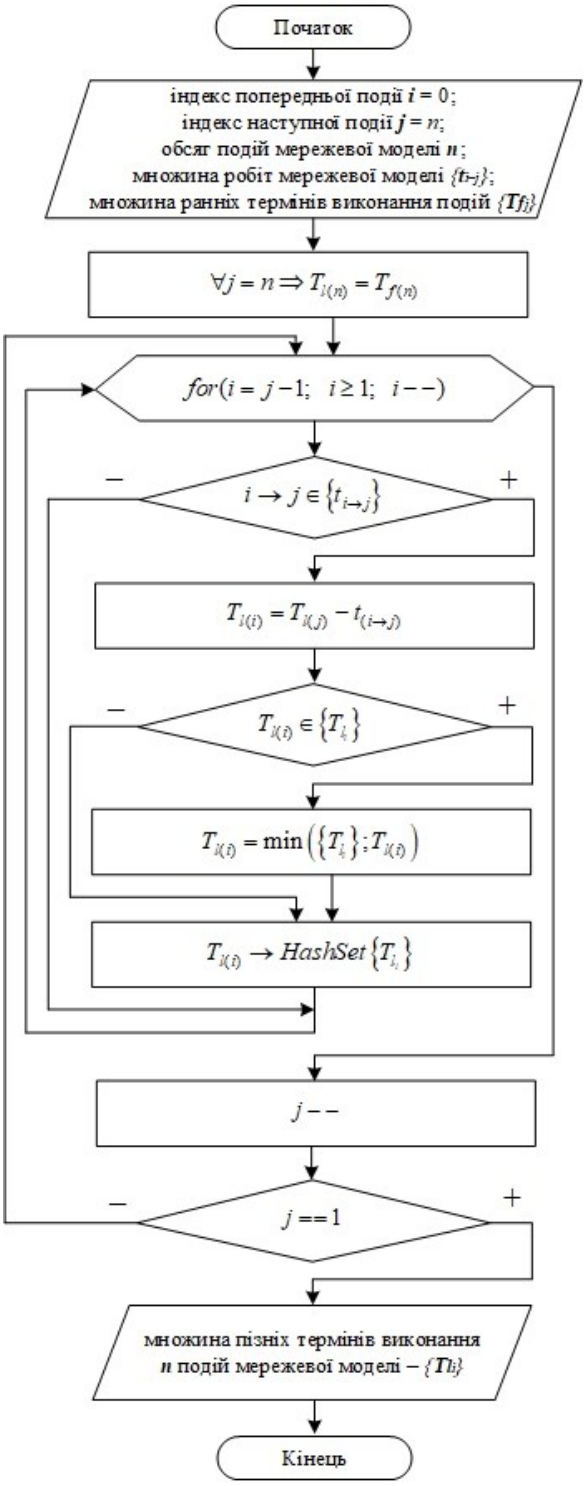


Рисунок 3 – Алгоритм обходу графа мережевої моделі для визначення показника $T_{l(i)}$

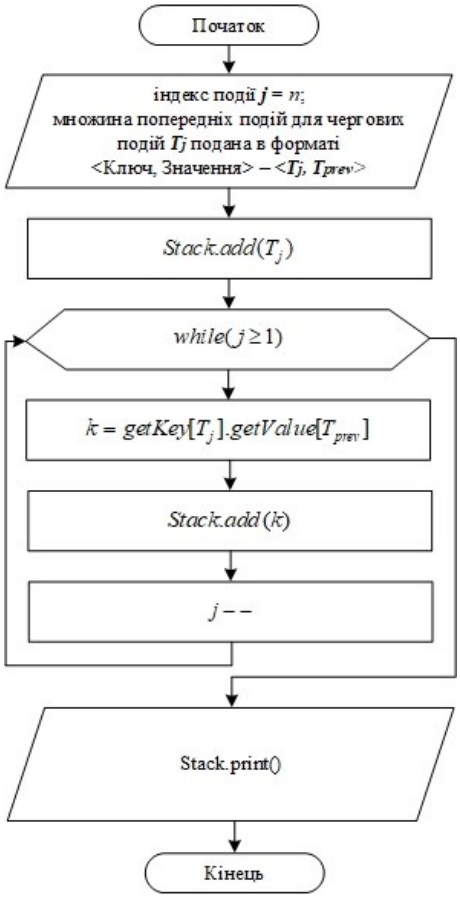


Рисунок 4 – Алгоритм обходу графа мережевої моделі для визначення критичного шляху

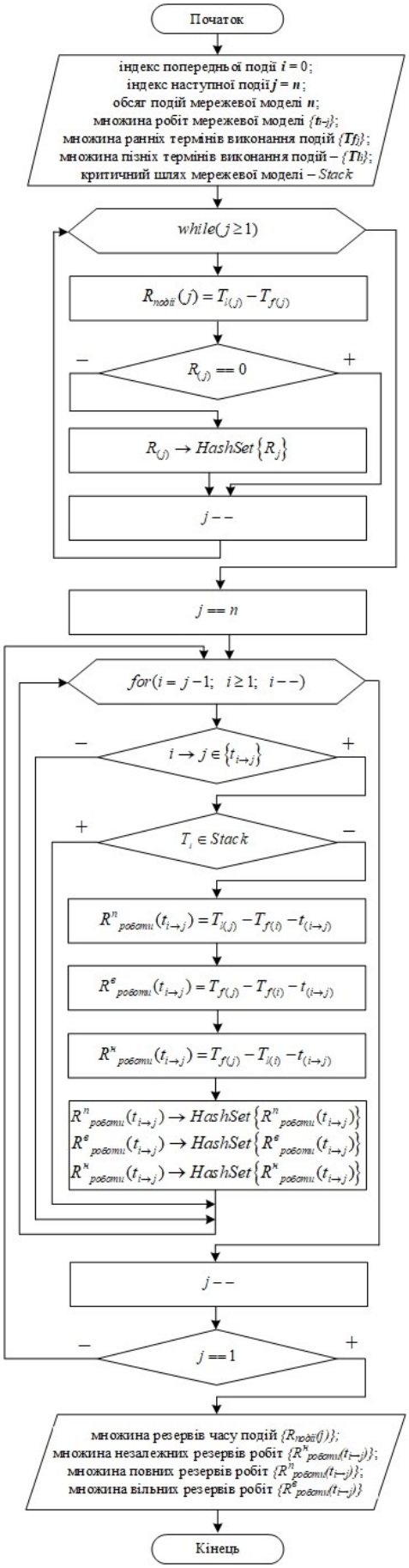


Рисунок 5 – Алгоритм обходу графа мережевої моделі для визначення означених резервів

Додаток Б. Лістинг програмного коду

ModelController.java

```
package com.example.dissertationweb.controllers;
import com.example.dissertationweb.entity.*;
import com.example.dissertationweb.service.DataService;
import com.example.dissertationweb.service.GraphService;
import com.example.dissertationweb.service.NetworkGraphService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;
import java.util.*;

@Controller
public class ModelController {

    List<UsS> usSList = new ArrayList<>();
    Graph graph = new Graph();
    List<GraphParameters> graphParametersList;
    private final RestTemplate restTemplate;

    @Autowired
    private NetworkGraphService networkGraphService;

    @Autowired
    public ModelController(RestTemplate restTemplate) {
        this.restTemplate = restTemplate;
    }

    @GetMapping("/")
    public String showForm() {
        return "usSForm";
    }

    @GetMapping("/form")
    public String showForm(Model model) {
        model.addAttribute("usS", new UsS());
        model.addAttribute("usSList", usSList);
        model.addAttribute("networkGraph", graph.buildNetworkGraph(usSList));
        //
        return "usSForm";
    }

    @PostMapping("/add")
    public String addObject(@ModelAttribute("uSS") UsS usS, String
previousStoriesString) {
        usS.setPreviousStoriesStr(previousStoriesString, usSList);
        usSList.add(usS);
        return "redirect:/form";
    }

    @GetMapping("/showUpdateForm/{id}")
    public String showUpdateForm(@PathVariable("id") int id, Model model) {
        UsS usS = usSList.get(id - 1);
        model.addAttribute("usS", usS);
        //model.addAttribute("usSList", usSList);
        return "updateUsS";
    }

    @PostMapping("/updateForm")
    public String updateForm(@ModelAttribute("uSS") UsS usS, String
```

```
previousStoriesString) {
        usS.setPreviousStoriesStr(previousStoriesString, usSList);
        usSList.set(usS.getNumber() - 1, usS);
        graphParametersList = new
ArrayList<>(networkGraphService.getGraphParametersMethod(graph.buildNetworkGrap
h(usSList)));
        return "redirect:/form";
    }

    @GetMapping("/delete/{id}")
    public String delete(@PathVariable("id") int id) {
        usSList.remove(id - 1);
        graphParametersList = new
ArrayList<>(networkGraphService.getGraphParametersMethod(graph.buildNetworkGrap
h(usSList)));
        return "redirect:/form";
    }

    @GetMapping("/networkGraphParameters")
    public String networkGraphPar(Model model) {
        graphParametersList = new
ArrayList<>(networkGraphService.getGraphParametersMethod(graph.buildNetworkGrap
h(usSList)));
        model.addAttribute("graphParametersList", graphParametersList);
        model.addAttribute("networkGraph",
networkGraphService.resultsOfManagementSystem(graphParametersList,
graph.buildNetworkGraph(usSList)));
        return "networkParameters";
    }

    public byte[] generateGraphMethod() {
        String url = "http://10.138.132.147:5000/generate_graph";
        Map<String, List<Map<String, String>>> data = new HashMap<>();
        List<Map<String, String>> edges = new ArrayList<>();

        for (Graph graph :
networkGraphService.resultsOfManagementSystem(graphParametersList,
graph.buildNetworkGraph(usSList))) {
            Map<String, String> edge = new HashMap<>();
            edge.put("id", String.valueOf(graph.getIsOnCriticalPath()));
            edge.put("from", String.valueOf(graph.getI()));
            edge.put("to", String.valueOf(graph.getJ()));
            edges.add(edge);
            System.out.println(graph);
        }

        data.put("edges", edges);

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<Map<String, List<Map<String, String>>>> request = new
HttpEntity<>(data, headers);
        ResponseEntity<byte[]> response = restTemplate.exchange(url,
HttpMethod.POST, request, byte[].class);
        return response.getBody();
    }

    @GetMapping("/generate-graph")
    public ResponseEntity<byte[]> generateGraph() {
        byte[] image = generateGraphMethod();
        return
ResponseEntity.ok().contentType(MediaType.IMAGE_PNG).body(image);
    }
}
```

Graph.java

```
package com.example.dissertationweb.entity;

import lombok.Data;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

@Data
public class Graph {

    private UsS usS;
    private int i;
    private int j;
    private MinTime minTime;
    private MaxTime maxTime;
    public int iMinTime;
    public int iMaxTime;
    public int jMinTime;
    public int jMaxTime;
    private int reserveIndependent;
    private int reserveFull;
    private int isOnCriticalPath;


    public Graph(UsS usS, int i, int j) {
        this.usS = usS;
        this.i = i;
        this.j = j;
    }

    public Graph() {
    }

    public Graph(int i) {
    }

    @Override
    public String toString() {
        return "UserStory=" + usS.getNumber() +
                ": [" + i + " (" + iMinTime + ", " + iMaxTime + ")" +
                ", " + j + " (" + jMinTime + ", " + jMaxTime + ")" +
                ", reserveFull: " + reserveFull +
                ", isOnCriticalPath: " + isOnCriticalPath;
    }

    public Set<Graph> buildNetworkGraph(List<UsS> userStories) {
        Set<Graph> networkGraph = new LinkedHashSet<>();
        int i = 1;
        int j = 1;

        Set<Graph> networkGraph2 = new LinkedHashSet<>();

        for (UsS usS : userStories) {
            if (usS.getPreviousStories().size() == 1 &&
usS.getPreviousStories().get(0).getNumber() == 0) {
                j++;
                Graph graph = new Graph(usS, i, j);
                networkGraph.add(graph);
                graph.setJ(j);
            } else if (usS.getPreviousStories().size() == 1 &&
usS.getPreviousStories().get(0).getNumber() != 0) {
                for (Graph graph : networkGraph) {
                    if (usS.getPreviousStories().get(0).getNumber() ==
graph.usS.getNumber()) {
                        j++;
                        Graph graph1 = new Graph(usS, graph.getJ(), j);
```

```
                        networkGraph2.add(graph1);
                        break;
                    }
                }
                networkGraph.addAll(networkGraph2);
            } else if (usS.getPreviousStories().size() > 1) {
                j++;
                int k = 0;
                while (k < usS.getPreviousStories().size()) {
                    for (Graph graph : networkGraph) {
                        if (graph.usS.getNumber() ==
usS.getPreviousStories().get(k).getNumber()) {
                            UsS fict = new UsS(0, 0);
                            Graph graphFict = new Graph(fict, graph.getJ(), j);
                            networkGraph2.add(graphFict);
                        }
                    }
                    k++;
                }
                i = j;
                j++;
                Graph graph4 = new Graph(usS, i, j);
                networkGraph2.add(graph4);

                for (Graph graph : networkGraph) {
                    if (graph.usS.getNumber() == usS.getNumber()) {
                        graph.setJ(j);
                        break;
                    }
                }
                networkGraph.addAll(networkGraph2);
            }
        }

        for (Graph graph : networkGraph) {
            System.out.println(graph.toString());
        }
        return networkGraph;
    }
}
```

GraphParameters.java

```
package com.example.dissertationweb.entity;
import lombok.Data;

@Data
public class GraphParameters {
    private int event;
    private int eventTE;
    private int eventTL;

    private int isCritical;

    public GraphParameters(int event, int eventTE) {
        this.event = event;
        this.eventTE = eventTE;
    }

    public GraphParameters(int event, int eventTE, int eventTL) {
        this.event = event;
        this.eventTE = eventTE;
        this.eventTL = eventTL;
    }

    public GraphParameters() {
    }
```

```
    @Override
    public String toString() {
        return event +
                " (" + eventTE +
                ", " + eventTL +
                ") is critical = " + isCritical;
    }
}
```

MaxTime.java

```
package com.example.dissertationweb.entity;
import lombok.Data;

@Data
public class MaxTime {

    private int tLi;
    private int tLj;

    public MaxTime(int tLi, int tLj) {
        this.tLi = tLi;
        this.tLj = tLj;
    }
}
```

MinTime.java

```
package com.example.dissertationweb.entity;
import lombok.Data;

@Data
public class MinTime {

    private int tEi;
    private int tEj;

    public MinTime(int tEi, int tEj) {
        this.tEi = tEi;
        this.tEj = tEj;
    }

    public MinTime(int tEj) {
        this.tEj = tEj;
    }
}
```

UsS.java

```
package com.example.dissertationweb.entity;
import lombok.Data;
import java.util.*;
import java.util.stream.Collectors;

@Data
public class UsS {
    private int number;
    private int storyPoint;
    private String caption;
    private List<UsS> previousStories;

    public UsS(int number, int storyPoint) {
        this.number = number;
        this.storyPoint = storyPoint;
        this.previousStories = new ArrayList<>();
    }
```

```
    public UsS(int number, int storyPoint, String caption) {
        this.number = number;
        this.storyPoint = storyPoint;
        this.caption = caption;
        this.previousStories = new ArrayList<>();
    }

    public UsS() {
    }

    @Override
    public String toString() {
        return "UserStory{" +
                "number=" + number +
                ", storyPoint=" + storyPoint +
                ", caption=" + caption +
                ", previousStories=" + previousStories +
                '}';
    }

    public void setPreviousStoriesStr(String previousStoriesString, List<UsS>
u) {
        if (previousStoriesString != null && !previousStoriesString.isBlank())
{
            List<UsS> previousStoriesList =
Arrays.stream(previousStoriesString.split(","))
                    .map(Integer::parseInt)
                    .map(number -> {
                        UsS previousUsS = new UsS();
                        if (number == 0) {
                            previousUsS.number = 0;
                            previousUsS.storyPoint = 0;
                        } else {
                            for (UsS usS : u) {
                               if (number == usS.getNumber()) previousUsS =
usS;

                            }
                        }
                        return previousUsS;
                    })
                    .collect(Collectors.toList());
            this.setPreviousStories(previousStoriesList);
        } else {
            System.err.println("!!!!!!!!!Error!!!!!");
        }
    }
}
```

NetworkGraphService.java

```
package com.example.dissertationweb.service;
import com.example.dissertationweb.entity.*;
import org.springframework.stereotype.Service;
import java.util.*;

@Service
public class NetworkGraphService {

    public Set<GraphParameters> getGraphParametersMethod(Set<Graph>
networkGraph) {
        List<Graph> networkGraphList = new ArrayList<>(networkGraph);
        Set<Graph> newGraphList = new LinkedHashSet<>();
        Set<GraphParameters> arrayListWithGraphParameters = new
LinkedHashSet<>();

        int n;
        int numberOfLastEvent = networkGraph.size() - 1;
        n = networkGraphList.get(numberOfLastEvent).getJ();
        int tEi;
```

```
        int tEj;

        Graph graph0 = new Graph();
        graph0.setUsS(new UsS(0, 0));
        graph0.setI(0);
        graph0.setJ(1);
        MinTime minTime0 = new MinTime(0);
        graph0.setMinTime(minTime0);
        MaxTime maxTime0 = new MaxTime(0, 0);
        graph0.setMaxTime(maxTime0);
        newGraphList.add(graph0);

        for (int k = 0; k < networkGraphList.size(); k++) {
            if (k == 0) {
                tEi = 0;
                Graph graphk = networkGraphList.get(k);

                tEj = tEi + graphk.getUsS().getStoryPoint();
                MinTime minTimeOb = new MinTime(tEi, tEj);
                graphk.setMinTime(minTimeOb);
                MaxTime maxTime = new MaxTime(0, 0);
                graphk.setMaxTime(maxTime);
                newGraphList.add(graphk);
            } else {
                for (int i = 1; i <= n; i++) {
                    for (int j = 2; j <= n; j++) {
                        if
(networkGraphList.get(k).getUsS().getPreviousStories().size() == 1) {
                            int a1 =
networkGraphList.get(k).getUsS().getPreviousStories().get(0).getNumber(); // №
попередньої роботи
                            if (i == networkGraphList.get(k).getI() &&
networkGraphList.get(k).getJ() == j && i == 1) {
                                tEi =
networkGraphList.get(a1).getMinTime().getTEi();
                                tEj = tEi +
networkGraphList.get(k).getUsS().getStoryPoint();
                                MinTime minTime2 = new MinTime(tEi, tEj);
                                networkGraphList.get(k).setMinTime(minTime2);
                                newGraphList.add(networkGraphList.get(k));
                            } else if (i == networkGraphList.get(k).getI() &&
networkGraphList.get(k).getJ() == j && i > 1) {
                                UsS a =
networkGraphList.get(k).getUsS().getPreviousStories().get(0);
                                for (Graph graph : networkGraphList) {
                                    if (graph.getUsS().equals(a)) {
                                        int max = graph.getMinTime().getTEj();
                                        tEj = max +
networkGraphList.get(k).getUsS().getStoryPoint();
                                        MinTime minTime3 = new MinTime(max,
tEj);

networkGraphList.get(k).setMinTime(minTime3);

newGraphList.add(networkGraphList.get(k));
                                    }
                                }
                            }
                        } else if
(networkGraphList.get(k).getUsS().getPreviousStories().size() > 1) {
                            UsS a =
networkGraphList.get(k).getUsS().getPreviousStories().get(0);
                            if (i == networkGraphList.get(k).getI() &&
networkGraphList.get(k).getJ() == j && i > 1) {
                                for (Graph graph : networkGraphList) {
                                    if (graph.getUsS().equals(a)) {
                                        int eI = graph.getJ();
                                        int eJ =
```

```
networkGraphList.get(k).getI();
                                        int max = graph.getMinTime().getTEj();

                                        for (int h = 1; h <
networkGraphList.get(k).getUsS().getPreviousStories().size(); h++) {
                                            UsS a1 =
networkGraphList.get(k).getUsS().getPreviousStories().get(h);
                                            for (Graph graph1 :
networkGraphList) {

                                                if (graph1.getUsS().equals(a1))
{

                                                    int eI1 = graph1.getJ();
                                                    int max1 =
graph1.getMinTime().getTEj();

                                                    if (max1 > max) {
                                                        max = max1;
                                                        eI = eI1;
                                                    }
                                                }
                                            }
                                        }
                                        MinTime minTime4 = new MinTime(max,
max);
                                        Graph newGraph = new Graph(new UsS(0,
0), eI, eJ);
                                        newGraph.setMinTime(minTime4);
                                        newGraphList.add(newGraph);

                                        tEj = max +
networkGraphList.get(k).getUsS().getStoryPoint();
                                        MinTime minTime5 = new MinTime(max,
tEj);


networkGraphList.get(k).setMinTime(minTime5);

newGraphList.add(networkGraphList.get(k));
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
        System.out.println();

        for (Graph g : newGraphList) {
            System.out.println(g.toString() + " " + g.getMinTime().getTEi() + "
" + g.getMinTime().getTEj());
        }
        Set<Graph> graphWithLateEvents = new LinkedHashSet<>();
        Collections.reverse(networkGraphList);
        networkGraphList.get(0).setMaxTime(new
MaxTime(networkGraphList.get(0).getMinTime().getTEj(),
networkGraphList.get(0).getMinTime().getTEj()));
        graphWithLateEvents.add(networkGraphList.get(0));
        int tLi;
        int tLj;
        for (int k = 0; k < networkGraphList.size(); k++) {
            for (int s = 1; s < networkGraphList.size(); s++) {
                if (networkGraphList.get(k).getI() ==
networkGraphList.get(s).getJ()) {
                    tLi = networkGraphList.get(k).getMaxTime().getTLi() -
networkGraphList.get(k).getUsS().getStoryPoint();
                    tLj = tLi -
networkGraphList.get(s).getUsS().getStoryPoint();
                    MaxTime maxTime = new MaxTime(tLi, tLj);
```

```
                    networkGraphList.get(s).setMaxTime(maxTime);
                    graphWithLateEvents.add(networkGraphList.get(s));
                }
            }
        }

        System.out.println(graphWithLateEvents.size());
        for (Graph gL : graphWithLateEvents) {
            for (Graph gE : newGraphList) {
                if (gL.getJ() == gE.getJ()) {
                    gE.setMaxTime(gL.getMaxTime());
                }
            }
        }
        System.out.println();

        for (Graph g : newGraphList) {
            System.out.println(g.toString() + " " + g.getMinTime().getTEi() + "
" + g.getMaxTime().getTLi());
        }

        for (Graph g : newGraphList) {
            GraphParameters graphParameters = new GraphParameters(g.getJ(),
g.getMinTime().getTEj(), g.getMaxTime().getTLi());
            arrayListWithGraphParameters.add(graphParameters);
        }
        System.out.println();

        for (GraphParameters g : arrayListWithGraphParameters) {
            if (g.getEventTE() == g.getEventTL()) {
                g.setIsCritical(0);
            } else {
                g.setIsCritical(g.getEventTL() - g.getEventTE());
            }
        }

        for (GraphParameters graphParameters : arrayListWithGraphParameters) {
            System.out.println(graphParameters.toString());
        }
        return arrayListWithGraphParameters;
    }

    public List<Graph> resultsOfManagementSystem (List<GraphParameters>
arrayListWithGraphParameters, Set<Graph> newGraphList){
        for (GraphParameters gp: arrayListWithGraphParameters){
            for (Graph g: newGraphList){
                if (gp.getEvent() == g.getI()){
                    g.setIMinTime(gp.getEventTE());
                    g.setIMaxTime(gp.getEventTL());
                }
                if (gp.getEvent() == g.getJ()){
                    g.setJMinTime(gp.getEventTE());
                    g.setJMaxTime(gp.getEventTL());
                }
            }
        }

        for (Graph g: newGraphList){
            System.out.println(g);
        }

        List<Graph> newFullGraphList = new ArrayList<>(newGraphList);
        for (Graph graph:  newFullGraphList){
            graph.setReserveIndependent(graph.getJMinTime() -
graph.getIMaxTime() - graph.getUsS().getStoryPoint());
            graph.setReserveFull(graph.getJMaxTime() - graph.getIMinTime() -
graph.getUsS().getStoryPoint());
        }
```

```
        for (Graph g: newFullGraphList){
            if (g.iMinTime == g.iMaxTime && g.jMinTime == g.jMaxTime){
                g.setIsOnCriticalPath(0);
            }else g.setIsOnCriticalPath(1);
        }

        System.out.println();
        for (Graph g: newFullGraphList){
            System.out.println(g);
        }
        return newFullGraphList;
    }
}
```

DissertationWebApplication.java

```
package com.example.dissertationweb;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class DissertationWebApplication {
    public static void main(String[] args) {
        SpringApplication.run(DissertationWebApplication.class, args);
    }
}
```

usSForm.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/html">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
          integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <title>UsS Form</title>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.
js"
            integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
            crossorigin="anonymous"></script>
</head>
<body>
<div class="container text-center mt-5">
    <h1>Expert Decision Support System Modeling in<br/>
        Lifecycle Management of Specialized Software</h1>
</div>
<div class="container">
    <div class="row">
        <div class="col-4">
            <div class="container text-center mt-3">
                <h3>Add new User Story to your Sprint Backlog</h3>
            </div>
            <form class="container text-left " action="/add" method="post">
                <div class="text-left mb-4 row">
                    <label for="number" class="col-sm-5 col-form-label">Number
of User Story (US): </label>
                    <div class="col-sm-7">
                        <input type="text" id="number" name="number"
class="form-control" required>
                    </div>
```

```
                </div>
                <div class="mb-4 row">
                    <label for="storyPoint" class="col-sm-5 col-form-
label">Story Point: </label>
                    <div class="col-sm-7">
                        <input type="text" id="storyPoint" name="storyPoint"
class="form-control" required>
                    </div>
                </div>
                <div class="mb-4 row">
                    <label for="caption" class="col-sm-5 col-form-
label">Description: </label>
                    <div class="col-sm-7">
                        <input type="text" id="caption" name="caption"
class="form-control" required>
                    </div>
                </div>
                <div class="mb-4 row">
                    <label for="previousStoriesString" class="col-sm-5 col-
form-label">Prev. Stories: </label>
                    <div class="col-sm-7">
                        <input type="text" id="previousStoriesString"
name="previousStoriesString" class="form-control"
                               required>
                    </div>
                </div>
                <div class="container text-center mt-3">
                    <button type="submit" class="btn btn-primary">Add User
Story to the Backlog</button>
                </div>
            </form>
            <h2>Network Graph</h2>
            <table class="table table-bordered">
                <tr>
                    <th class="col-sm-8">Number of User Story (US)</th>
                    <th>Start event(i)</th>
                    <th>End event(j)</th>
                </tr>
                <tr th:each="n : ${networkGraph}">
                    <td th:text="${n.usS.number}"></td>
                    <td th:text="${n.i}"></td>
                    <td th:text="${n.j}"></td>
                </tr>
            </table>
        </div>
        <div class="col-8">
            <div class="container text-center">
                <div class="row">
                    <div class="col">
                        <h2>Backlog</h2>
                        <table class="table table-bordered">
                            <tr>
                                <th class="col-sm-1"> Number of User Story
(US)</th>
                                <th class="col-sm-1">Story Point</th>
                                <th>Description</th>
                                <th class="col-sm-4">Previous Stories</th>
                                <th class="col-sm-1">Change</th>
                            </tr>
                            <tr th:each="usS : ${usSList}">
                                <td th:text="${usS.number}"></td>
                                <td th:text="${usS.storyPoint}"></td>
                                <td th:text="${usS.caption}"></td>
                                <td th:text="${usS.previousStories}"></td>
                                <td><div class="d-grid gap-2">
                                    <a
th:href="@{/showUpdateForm/{id}(id=${usS.number})}" class="btn btn-primary btn-
sm">Edit</a>
```

```
                                        <a
th:href="@{/delete/{id}(id=${usS.number})}" class="btn btn-danger btn-sm"
style="margin-bottom: 10px;">Delete</a>
                                    </div>
                                    </td>
                                </tr>
                            </table>
                        </div>
                        <a th:href="@{/networkGraphParameters}" type="button"
class="btn btn-primary">Show results</a>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
</div>
</body>
</html>
```

updateUsS.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org"
xmlns:sec="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Log In Form</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet"
          integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
</head>
<body>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"
        integrity="sha384-
oBqDVmMz9ATKxIep9tiCxS/Z9fNfEXiDAYTujMAeBAsjFuCZSmKbSSUnQlmh/jp3"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js"
        integrity="sha384-
IDwe1+LCz02ROU9k972gdyvl+AESN10+x7tBKgc9I5HFtuNz0wWnPclzo6p9vxnk"
        crossorigin="anonymous"></script>

<div class="container text-center mt-5">
    <h2> Edit information: </h2>
</div>
<div class="container text-left">
    <div class="row">
        <div class="col">
            <div class="container text-left mt-5">
                <div class="container text-left mx-5">
                    <form th:action="@{/updateForm}" th:object="${usS}"
                          method="post">
                        <div class="mb-4">
                            <label for="number" class="form-label"> Number of
US </label>
                            <input type="text"
                                   th:field="*{number}"
                                   class="form-control"
                                   id="number"/>
                        </div>
                        <div class="mb-3">
                            <label for="storyPoint" class="form-label">Story
```

```
Point</label>
                                <input type="text"
                                       th:field="*{storyPoint}"
                                       class="form-control"
                                       id="storyPoint"
                                       name="storyPoint"/>
                            </div>
                            <div class="mb-3">
                                <label for="caption" class="form-
label">Caption</label>
                                <input type="text"
                                       th:field="*{caption}"
                                       class="form-control"
                                       id="caption"
                                       name="caption"/>
                            </div>
                            <div class="mb-3">
                                <label for="previousStoriesString" class="col-sm-5
col-form-label">Prev. Stories: </label>
                                <div class="col-sm-7">
                                    <input type="text" id="previousStoriesString"
name="previousStoriesString" class="form-control"
                                           required>
                                </div>
                            </div>
                            <div class="container text-center my-1">
                                <button type="submit" class="btn btn-success"> Send
</button>
                            </div>
                            <a th:href="@{/indexUser}" class="btn btn-secondary">
Back to Information</a>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

networkGraphParameters.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <title>Title</title>
</head>
<body>
<div class="container text-center mt-5">
    <h1>Network Graph Parameters</h1>
</div>
<div class="container text-left mt-3">
    <a href="/form">Back to Backlog</a>
</div>
<div class="container text-center">
    <div class="row">
        <div class="col">
            <h2>Backlog</h2>
            <table class="table table-bordered">
                <tr>
                    <th>Event</th>
```

```
                    <th>Start of the event execution (TEarly)</th>
                    <th>End of the event execution (TLate)</th>
                    <th>Time Reserve </th>
                </tr>
                <tr th:each="gP : ${graphParametersList}">
                    <td th:text="${gP.event}"></td>
                    <td th:text="${gP.eventTE}"></td>
                    <td th:text="${gP.eventTL}"></td>
                    <td th:text="${gP.isCritical}"></td>
                </tr>
            </table>
            <h2>Network Graph</h2>
            <table class="table table-bordered">
                <tr>
                    <th class="col-sm-1">Number of User Story (US)</th>
                    <th>Description</th>
                    <th class="col-sm-1">Start event(i)</th>
                    <th class="col-sm-1">End event(j)</th>
                    <th class="col-sm-1">Time reserve</th>
                </tr>
                <tr th:each="n : ${networkGraph}">
                    <td th:text="${n.usS.number}"></td>
                    <td th:text="${n.usS.caption}"></td>
                    <td th:text="${n.i}"></td>
                    <td th:text="${n.j}"></td>
                    <td th:text="${n.reserveFull}"></td>
                </tr>
            </table>
        </div>
    </div>
</div>
<div class="container text-center">
    <h2>Graphical presentation</h2>
    <img th:src="@{/generate-graph}" alt="Graph Image"/>
</div>
</body>
</html>
```

app.py

```
from flask import Flask, request, send_file
import matplotlib.pyplot as plt
import networkx as nx
import io
import numpy as np
app = Flask(__name__)
@app.route('/generate_graph', methods=['POST'])
def generate_graph():
    data = request.json
    G = nx.DiGraph()
    edge_colors = []
    for edge in data['edges']:
        G.add_edge(edge['from'], edge['to'], id=edge['id'])
        if int(edge['id']) == 0:
            edge_colors.append('red')
        else:
            edge_colors.append('black')
    print("vuvuyvyiu")
    plt.figure(figsize=(10, 8))
    pos = nx.spring_layout(G)
    nx.draw(G, pos, with_labels=True, node_color='skyblue', node_size=800,
font_size=16, font_color='black', edge_color=edge_colors, width=2, arrowsize=20)
    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    return send_file(img, mimetype='image/png')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Додаток В. Акт проведених досліджень

ЗАТВЕРДЖУЮ
Т.в.о. проректора з науково-дослідної роботи Львівського державного університету безпеки життєдіяльності
полковник служби цивільного захисту
_____ Василь ПОПОВИЧ
“___” _____ 2024 р.

АКТ ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

здобувачки третього (освітньо-наукового) рівня освіти із спеціальності 122 «Комп’ютерні науки» Назар Юлії Сергіївни за темою «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах»

Цим актом затверджується, що під час написання дисертаційної роботи Назар Юлії Сергіївни на тему: «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах» були проведені дослідження для розрахунку тривалості розробки спеціалізованого програмного забезпечення. Для дослідів було обрано 4 безпеко-орієнтовані сервіси (Чат бот консультаційної підтримки «Вступнику ЛДУ БЖД», Мобільний додаток для обліку та пошуку найближчих об’єктів укриття, Інформаційно-довідкова система «Unibell», Інформаційна система визначення оптимальної кількості ліній зв’язку «101» та чисельності диспетчерського складу), які розроблялись на базі кафедри інформаційних технологій та систем електронних комунікацій. Загалом було проведено по 8 спринтів та отримано наступні результати:

Назва СПЗ	Досвід команди, р.	Спринт 1		Спринт 2		Спринт 3		Спринт 4	
		Зміни, %	Тривалість,s.p	Зміни, %	Тривалість,s.p	Зміни, %	Тривалість,s.p	Зміни, %	Тривалість,s.p
Чат бот консультаційної підтримки «Вступнику ЛДУ БЖД»	1,5	0,01	73	0,01	69	0	68	0	70
Мобільний додаток для обліку та пошуку найближчих об’єктів укриття	3	0,15	80	0,1	76	0,1	77	0,1	75
Інформаційно-довідкова система «Unibell»	6.6	0,01	55	0,02	53	0,05	59	0,01	54
Інформаційна система визначення оптимальної кількості ліній зв’язку «101» та чисельності диспетчерського складу	12.6	0,12	65	0,1	63	0,2	66	0,15	65

Назва СПЗ	Досвід команди, р.	Спринт 5		Спринт 6		Спринт 7		Спринт 8	
		Зміни, %	Тривалість,s.p	Зміни, %	Тривалість,s.p	Зміни, %	Тривалість,s.p	Зміни, %	Тривалість,s.p
Чат бот консультаційної підтримки «Вступнику ЛДУ БЖД»	1.5	0,01	72	0	67	0,01	71	0,01	74
Мобільний додаток для обліку та пошуку найближчих об'єктів укриття	3	0,17	79	0,17	80	0,15	82	0,12	78
Інформаційно-довідкова система «Unibell»	6.6	0,01	53	0	50	0,01	52	0,02	54
Інформаційна система визначення оптимальної кількості ліній зв'язку «101» та чисельності диспетчерського складу	12.6	0,1	63	0,1	62	0,12	62	0,18	67

Науковий керівник
к.т.н., доцент — Олександр ПРИДАТКО

Науковий керівник
к.т.н., доцент — Ольга СМОТР

Ад'юнкт — Юлія НАЗАР

Додаток Г. Акти впровадження дисертаційного дослідження

ЗАТВЕРДЖУЮ
Т.в.о. першого проректора Львівського державного університету безпеки життєдіяльності
_____ Андрій ОСТАП'ЮК
"04" 06 2024 р.

АКТ ВПРОВАДЖЕННЯ
результатів дисертаційної роботи
здобувачки третього (освітньо-наукового) рівня освіти із спеціальності 122 «Комп'ютерні науки» Назар Юлії Сергіївни за темою «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах» в освітній процес кафедри інформаційних технологій та систем електронних комунікацій

Цим актом засвідчується, що в освітньому процесі Львівського державного університету безпеки життєдіяльності у рамках викладання дисциплін «Управління IT-проектами», «Об'єктно-орієнтоване програмування» для здобувачів першого (бакалаврського) рівня освіти зі спеціальності 122 «Комп'ютерні науки», використовуються результати дисертаційного дослідження Назар Юлії Сергіївни за темою «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах». Також в освітньому процесі означених дисциплін використовувались результати виконання науково-дослідних робіт «Методи аналізу життєвого циклу програмних систем безпеко-орієнтованого спрямування в динамічному оточенні» (д. р. н. 0123U101699), «Дослідження параметрів оцінки протипожежного стану об'єктів та побудова на їх основі комп'ютерної аналітичної системи «Комп'ютерна аналітична система» (д. р. н. 0122U200729), а також дослідно-конструкторської роботи «Розроблення середовища та програмної системи консультаційної допомоги на базі мобільних платформ з інтегрованою функцією інформування», які входять до переліку наукових досліджень університету та виконувались в рамках написання дисертаційного дослідження. Освітній процес з означених дисциплін забезпечується на кафедрі інформаційних технологій та систем електронних комунікацій (лектори к.т.н., доцент, доцент кафедри Смотр О.О. та к.т.н., доцент, начальник кафедри Придатко О. В.). Окремі наукові положення дисертаційного дослідження відображені в силабусах дисциплін та методичних матеріалах окремих тем курсів.

Начальник кафедри інформаційних
технологій та систем електронних комунікацій
к.т.н., доцент Олександр ПРИДАТКО

ЗАТВЕРДЖУЮ
Директор ТОВ «Емброкс»

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Назар Юлії Сергіївни за темою «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах»

Одержані наукові результати Назар Юлії Сергіївни у дисертаційному дослідженні «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умова» мають вкрай важливе практичне значення при розробці програмного забезпечення у динамічних умовах. Зокрема, розроблена інформаційна технологія підтримки прийняття рішень в управлінні життєвим циклом програмного забезпечення була використана на етапі планування розроблення UI/UX дизайну інформаційної системи «Я доброволець», яким займалась наша компанія.

Дана технологія забезпечила якісний менеджмент розробки дизайну системи, а також можливість у реальному часі реагувати на зміни та корективи, які вносили замовники під час розробки. Таким чином, можна стверджувати, що одержані моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення є вкрай актуальними під час розроблення програмних систем у динамічних умовах, а також рекомендовані до використання у проєктах, які полягають у розробці спеціалізованого програмного забезпечення.

06.06.2024 підпис/печатка Гірак Н.Б.

ЗАТВЕРДЖУЮ
Директор Департаменту запобігання
надзвичайним ситуаціям ДСНС України
Олександр ЧЕКРИГІН
06 2024 р.

АКТ ВПРОВАДЖЕННЯ
результатів дисертаційної роботи
здобувачки третього (освітньо-наукового) рівня освіти із спеціальності 122 «Комп'ютерні науки» Назар Юлії Сергіївни за темою «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах» під час розроблення безпеко-орієнтованих сервісів для потреб ДСНС України

Цим актом засвідчується, що розроблені у рамках виконання дисертаційного дослідження Назар Юлії Сергіївни моделі та інформаційна технологія підтримки прийняття рішень використані під час менеджменту окремих етапів розроблення інформаційно-аналітичної системи «Інтерактивний інспектор», яка розроблена у Львівському державному університеті безпеки життєдіяльності на замовлення Департаменту запобігання надзвичайним ситуаціям ДСНС України. Інформаційно-аналітична система включає 14 блоків та дозволяє автоматизувати процеси оцінки протипожежного стану об'єктів залежно від параметрів їх функціонування. Інформаційно-аналітична система використовується посадовими особами ДСНС України та суб'єктами господарювання.

Отримані наукові положення та побудована на їх основі інформаційна технологія в рамках написання дисертаційного дослідження Назар Юлії Сергіївни були використані при організації командної роботи та менеджменту процесів розроблення інформаційно-аналітичної системи. Застосовані моделі та інформаційна технологія дозволили продемонструвати високу ефективність менеджменту розробки спеціалізованого програмного забезпечення в динамічних умовах. Інформаційна технологія управління життєвим циклом розроблення програмного забезпечення дозволила своєчасно та на якісно-належному рівні організувати розроблення інформаційно-довідкової системи «Інтерактивний інспектор». Отримані результати в рамках написання дисертаційного дослідження Назар Юлії Сергіївни за темою «Моделі та інформаційна технологія управління життєвим циклом розроблення програмного забезпечення в динамічних умовах» мають вагоме практичне значення та рекомендовані для подальшого використання відповідними підрозділами Державної служби України із надзвичайних ситуацій під час розроблення спеціалізованого програмного забезпечення безпеко-орієнтованого спрямування у динамічному оточенні.

Заступник начальника управління
пожежної безпеки –
начальник відділу профілактичної роботи
управління пожежної безпеки
Департаменту запобігання
надзвичайним ситуаціям ДСНС України Степан ЮДІН