Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів та моделей створення мережевої структури робіт ІТ-проєкту веборієнтованої програмної системи
(тема)

Виконав:
здобувач 2 року навчання,
групи УПГІТм-23-2
Дмитро БОРДЮГ
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Управління проектами в галузі ІТ
(повна назва освітньої програми)

Керівник: доц. Тетяна БОРИСЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри ІУС _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Інформаційних управляючих систем

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Управління проектами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"21" квітня 20 25 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Бордюгу Дмитру Євгенійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів та моделей створення мережевої структури робіт ІТ-проєкту веборієнтованої програмної системи

затверджена наказом по університету від "28" березня 2025 р. № 235Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії "03" червня 2025 р.

3. Вихідні дані до роботи науково-технічні публікації та інтернет джерела з тематики з досліджуваної проблеми, матеріали передатестаційної практики .

4. Перелік питань, що потрібно опрацювати у роботі аналіз особливостей управління розробкою веборієнтованих програмних систем; досліження методу мережевого моделювання структури робіт проєкту; використання методу мережевого моделювання для управління ІТ-проєктом; експериментальна перевірка методу мережевого моделювання під час управління ІТ-проєктом вебзастосунку.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз особливостей розробки веборієнтованих програмних систем	21.04.2025 - 26.04.2025	Виконано
2	Аналіз методологій та програмних засобів управління проектом з розробки вебзастосунків	27.04.2025 - 10.05.2025	Виконано
3	Постановка задач магістерської кваліфікаційної роботи	11.05.2025 – 13.05.2025	Виконано
4	Аналіз підходу до опису зв’язків між функціями одного рівня декомпозиції	14.05.2025 – 18.05.2025	Виконано
5	Дослідження методу побудови та аналізу мережевої структури робіт ІТ-проєкту	19.05.2025 – 21.05.2025	Виконано
6	Розробка підходу до використання мережевого моделювання для управління ІТ-проєктом за методологією Scrum	21.05.2025 – 24.05.2025	Виконано
7	Розробка тезаурусу ІТ-проєкта вебзастосунку	25.05.2025 – 27.05.2025	Виконано
8	Експериментальна перевірка методу мережевого моделювання під час управління ІТ-проєктом вебзастосунку	28.05.2025 – 02.06.2025	Виконано
9	Аналіз отриманих результатів	03.06.2025 – 04.06.2026	Виконано
10	Оформлення пояснювальної записки до кваліфікаційної роботи	05.05.2026 – 06.06.2025	Виконано
11	Захист кваліфікаційної роботи в екзаменаційній комісії	07.06.2024	Виконано

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тетяна БОРИСЕНКО
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 89 с., 11 рис., 15 табл., 1 дод., 24 джерел.

ДЕКОМПОЗИЦІЯ, ДІАГРАМА ГАНТА, КРИТИЧНИЙ ШЛЯХ, МЕРЕЖЕВА МОДЕЛЬ, ТЕЗАУРУС, AON, BACKLOG, SCRUM, TASK, USER STORY.

Об’єктом дослідження кваліфікаційної роботи є методи та моделі створення мережевої структури робіт ІТ-проєкту веборієнтованої програмної системи.

Метою дослідження є апробація застосування методу мережевого моделювання для виявлення та опису горизонтальних зв’язків між одиницями робот ІТ-проєкту на кожному рівні декомпозиції в умовах використання методології Scrum для керування проєктом.

Під час дослідження в кваліфікаційній роботі застосовувалися такі методи: спостереження, наукові факти, порівняння, експеримент та аналіз.

Робота містить таке: результати аналізу особливостей управління розробкою веборієнтованих програмних систем; результати дослідження методу мережевого моделювання структури робіт ІТ-проєкту, опис підходу до застосування методу мережевого моделювання для управління ІТ-проєктом в умовах використання методології Scrum; опис проведення та результати експериментальної перевірки методу мережевого моделювання під час управління ІТ-проєктом вебзастосунку.

Результати, які отримані в кваліфікаційній роботі, можуть бути використані для подальших досліджень автоматичного встановлення зв'язків між роботами ІТ-проєкту.

ABSTRACT

Master's thesis: 89 pages, 11 figures, 15 tables, 1 appendices, 24 sources.

AON, BACKLOG, CRITICAL PATH, DECOMPOSITION, GANTT CHART, NETWORK MODEL, SCRUM, THESAURUS, TASK, USER STORY.

The object of research of the qualification work is methods and models for creating a network structure of IT project work for a web-based software system.

In order to test the application of the network modeling method to identify and describe horizontal relationships between IT project work units at each decomposition level in the context of using the Scrum methodology for project management.

During the research in the qualification work, the following methods were used: observation, scientific facts, comparison, experiment, and analysis.

The work contains the following: results of the analysis of the features of managing the development of web-based software systems; results of the study of the network modeling method of the IT project work structure, a description of the approach to applying the network modeling method to manage an IT project in the context of using the Scrum methodology; description of the implementation and results of experimental testing of the network modeling method during the management of an IT project of a web application.

The results obtained in the qualification work can be used for further research into automatically establishing connections between IT project tasks.

ЗМІСТ

С.

Скорочення та умовні познаки 8

Вступ 9

1 Аналіз особливостей управління розробкою веборієнтованих програмних систем 10

1.1 Аналіз особливостей розробки веборієнтованих програмних систем 10

1.2 Аналіз методологій та програмних засобів управління проектом з розробки вебзастосунків 16

1.3 Аналіз мережевих моделей опису структури робіт проєкту 23

1.4 Постановка задач магістерської кваліфікаційної роботи 24

2 Досліження методу мережевого моделювання структури робіт проєкту 26

2.1 Підхід до опису зв'язків між функціями одного рівня декомпозиції 26

2.2 Метод побудови та аналізу мережевої структури робіт ІТ-проєкту .. 28

3 Використання методу мережевого моделювання для управління ІТ-проєктом 38

3.1 Загальні положення 38

3.2 Використання методу на рівні епіків 41

3.3 Використання методу на рівні фіч 42

3.4 Використання епіків на рівні User Story та Task 43

4 Експерементальна перевірка методу мережевого моделювання під час управління ІТ-Проєктом вебзастосунку 48

4.1 Опис особливостей ІТ-проєкту вебзастосунку «Благодійні суботники» 48

4.2 Розробка тезаурусу ІТ-проєкта вебзастосунку 49

4.3 Розробка мережевих моделей робіт вебзастосунку 53

4.4 Аналіз отриманих результатів 71

Висновки 72
Перелік джерел посилання 74
Додаток А Графічний матеріал кваліфікаційної роботи 77

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

AON – Activity on Node

API – Application Programming Interface

HTTP – HyperText Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

JSON – JavaScript Object Notation

PERT – Program Evaluation and Review Technique

UI – User Interface

UML – Unified Modeling Language

ВСТУП

Об’єктом дослідження кваліфікаційної магістерської роботи є методи та моделі створення мережевої структури робіт ІТ-проєкту веборієнтованої програмної системи.

Методи мережевого моделювання проєктів мають багаторічну історію застосування й завоювали визнання у класичному управління складними проєктами. Історія мережевого моделювання починається у середині XX століття. Натомість впродовж останнього десятиліття у сфері інформаційних технологій дедалі більше поширюються гнучкі методи управління проєктами, серед яких один із лідерів – Scrum. Але на цей час застосування традиційних мережевих моделей для управління ІТ-проєктами в умовах використання методології Scrum залишається недостатньо дослідженим.

Актуальність цього дослідження обумовлена необхідністю адаптації перевірених мережевих алгоритмів до ітеративного контексту веборієнтованих ІТ-систем. Це включає обґрунтування методик побудови та аналізу мережевих моделей у рамках спринтового планування, а також визначення їхніх переваг і обмежень у порівнянні з традиційними підходами Scrum. Подібна інтеграція методів сприятиме підвищенню прозорості процесу планування, зменшенню ризику затримок і забезпеченню точнішого контролю за виконанням завдань у динамічному середовищі розробки.

Кваліфікаційну роботу виконано згідно методичних вказівок щодо розробки та оформлення кваліфікаційної роботи [1], ДСТУ 3008:2015 [2] та ДСТУ 8302:2015 [3].

1 АНАЛІЗ ОСОБЛИВОСТЕЙ УПРАВЛІННЯ РОЗРОБКОЮ ВЕБОРІЄНТОВАНИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз особливостей розробки веборієнтованих програмних систем

Розглянемо етапи розробки веборієнтованих програмних систем. Розглядаючи етапи розробки вебзастосунку насамперед, варто зупинитися на основних його відмінностях від звичайного сайту. Вебзастосунок є спеціальною програмою, яка може бути запущена в будь-якому браузері. Всі елементи та компоненти відображаються на екрані стаціонарного комп'ютера, ноутбука, смартфона або іншого гаджета, зовні представляється сукупністю вебсторінок [4].

Важливою характеристикою цього цифрового рішення є наявність не лише статичного контенту, але ще й інтерактивних елементів інтерфейсу, які дозволяють виконувати певний ряд дій.

Розробка вебзастосунку складається за кількох основних етапів, серед яких є планування, реалізація, проєктування, розгортання та тестування. Кожен із цих етапів є ключовим для створення надійного, ефективного, продуктивного та безпечного вебпродукту, а саме:

- проєктування архітектури;
- планування та аналіз вимог;
- розробка фронтенду;
- розробка бекенду;
- дизайн інтерфейсу (UI/UX);
- тестування;
- інтеграція бази даних (БД);
- розгортання (Deployment), оновлення та підтримка.

Над кожною веборієнтованою програмною системою працюють різні спеціалісти, кожен із яких відповідає за свій напрямок розробки (наприклад, бізнес-аналітики, менеджери, UI/UX-дизайнери, розробники фронтенду,

бекенду, тестувальники тощо).

Наступним кроком стане ретельніше дослідження та розглядання особливостей з розробки архітектури веборієнтованих програмних систем.

Архітектура вебзастосунку – це організація компонентів вебзастосунку, як на фронтенді, так і на бекенді, які забезпечують його функціональність, продуктивність і масштабованість. Вона визначає, як різні елементи вебзастосунку взаємодіють, спілкуються і працюють разом, щоб забезпечити безперебійну роботу користувача [5].

Розробка архітектури визначає структуру вебзастосунку, взаємодію між компонентами, а також підходи до забезпечення продуктивності, масштабованості та безпеки.

Архітектурні підходи складаються із:

- клієнт-серверної архітектури – це стандартний підхід, в якому клієнт взаємодіє з сервером через HTTP запити;
- монолітна архітектура – де вся логічна частина застосунку розміщується у єдиній кодовій базі;
- мікросервісна архітектура – коли система розподілена на незалежні малі сервіси, які взаємодіють через API;
- serverless-архітектура – використання хмарних сервісів для виконання конкретних завдань без необхідності управління серверами.

Основні компоненти архітектури вебзастосунку становлять собою структурні елементи, які відповідають за його функціонування, забезпечують взаємодію між клієнтом та сервером, обробляють дані і гарантують безпеку та масштабованість. Ці компоненті включають в себе:

- фронтенд (інтерфейс, який відповідає за взаємодію з користувачем);
- бекенд (реалізує серверну частину та логіку, де відображено управління запитами та роботу з базами даних);
- API-шлюз (обмін даними між частинами фронтеду та бекенду та управління запитами);
- базу даних (зберігання інформації);

− процеси кешування (тимчасове збереження даних у швидко-доступному сховищі (кеші), цим зумовлюється зменшення навантаження на серверну частину за для прискорення обробки запитів);

− балансування навантаження та системи безпеки (процес розподілу трафіку між серверами для масштабованості).

Основні принципи для побудови ефектної архітектури веборієнтованих програмних систем базуються на забезпеченні таких критичних характеристик, як масштабованість, продуктивність, гнучкість системи та її безпека. Одним із головних принципів це є масштабованість, що дає змогу системі адаптуватися до збільшення навантаження без втрати ефективності проєкту. Це може досягатися за шляхом вертикального (підвищення потужностей вже наявних серверів) або горизонтального (додавання нових серверів) масштабування. Також надзвичайно важливим є процес дотримання безпеки, які охоплюють в себе шифрування даних, застосування безпечних протоколів на кшталт HTTPS, а також впровадження механізмів аутентифікації та авторизації користувачів веборієнтованих програмних систем.

Слід зазначити, що важливим аспектом є забезпечення гнучкості та модульності, котрі дають змогу легко змінювати або оновлювати окремі компоненти вебзастосунку без всілякого негативного впливу на весь проєкт. Такий результат досягається завдяки застосуванню мікросервісної архітектури або створенню якісного та структурованого коду в рамках монолітних застосунків. Важливо також враховувати момент відмовостійкості – це процес, в якому здатність системи залишається функціональною навіть у випадках, коли відбуваються збої окремих її елементів. Для цього застосовуються механізми автоматичного відновлення, резервні сервери або балансування навантаження. Крім того, для швидкості обробки запитів і ефективного застосування ресурсів використовуються методи кешування на різних рівнях, таких як фронтенд, бекенд та бази даних.

У вебзастосунках програмних систем зв’язки між функціями можуть

бути – горизонтальними (коли взаємодія відбувається між функціями одного рівня деталізації) так і вертикальними (коли дрібніші функції є складовими батьківських функцій). Горизонтальні зв’язки створюють додаткові труднощі при реалізації проєкту, оскільки вимагають враховувати такі взаємодії під час планування завдань для кожного спринта. Окрім цього, вони ускладнюють належну організацію взаємодії між незалежними функціями, сервісами або модулями.

Ключові виклики горизонтальних зв’язків між функціями за час виконання програмної реалізації:

− залежності між модулями (зміна одного модуля в подальшому може вплинути на роботу та функціонування інших, яке ускладнює процес оновленню системи);

− синхронізація даних (оскільки різні модулі можуть використовувати однакові дані, то важливо уникати конфліктів і забезпечувати актуальність інформації);

− безпека (усі взаємодії які утворюються між модулями мають бути надійно захищені, задля запобіганню витоку даних або несанкціонованого доступу);

− продуктивність (горизонтальні зв’язки можуть збільшувати додаткове навантаження на проєкт із-зі численних викликів між сервісами).

Тепер розглянемо методи вирішення проблем горизонтальних зв’язків між програмними компонентами, зокрема такі, як:

− мікросервісна архітектура (поділ функціональності на незалежні сервіси, котрі обмінюються даними за допомогою API);

− поділ на шари (застосування проміжного рівня для керування та управління запитами між модулями);

− кешування (скорочення кількості запитів між функціями для оптимізації продуктивності);

− балансування навантаження (рівномірний розподіл трафіку між модулями для запобігання збільшення затримок в системі).

Горизонтальні зв’язки відіграють важливу роль у масштабованих системах, де необхідно організувати взаємодію між різноманітними сервісами (наприклад, управління логікою, користувачами, обробкою платежів тощо), також між найменшими одиницями роботи беклогу проєкту, такі як User Story та Task, впливають на пріоритетність та черговість виконання робіт під час спринта. Із-зі цього моменту їх необхідно враховувати під час планування робіт проєкту для чергового спринта.

Одним із ключових елементів для успішної реалізації IT-проєкту вебзастосунку є його тезаурус.

Тезаурус – відіграє важливу роль у процесі розробки архітектури, оскільки забезпечує стандартизацію термінології. Це дозволяє всім членам команди (розробникам, дизайнерам, тестувальникам тощо) спілкуватись єдиною мовою, сприяючи для більш ефективної взаємодії. Тезаурус проєкту являє собою систематизований набір понять, термінів та визначень, які використовуються в межах створення вебзастосунку.

Під час процесу проєктування архітектури вебзастосунку важливо визначити основні компоненти та способи їхньої взаємодії, базу даних, API, протоколи зв’язку та інші ключові аспекти. Відсутність спільного розуміння термінів у команді, може спричинити логічні помилки у роботі проєкту, наприклад, дублювання функцій і створити плутанину в документації.

Створення тезауруса ще на початкових етапах дає змогу уникнути багатьох ускладнень, зробивши архітектуру проєкту більш зрозумілішою та упорядкованою.

Однією з складових частин під час розробки веборієнтованих програмних систем є такі частини, як – фронтенд та бекенд.

Фронтенд – це не просто код і дизайн, а стратегічний інструмент, який допомагає різним проєктам конкурувати, залучати й утримувати клієнтів. Це те, що бачить і відчуває користувач: дизайн, зручність, динаміка. Саме фронтенд визначає, як швидко завантажуються сторінки, як плавно працюють анімації та наскільки легким і приємним є користування сайтом. Оптимізація

скриптів, завантаження зображень і загальна швидкість роботи [6].

Основна задача фронтенду заключається у створенні користувацького інтерфейсу, тобто це те все що бачить користувач і з чим він взаємодіє. Його реалізація здійснюється за допомогою HTML, CSS, JavaScript а також різних фреймворків (наприклад, React, Vue.js, Angular). Основна функція фронтенду полягає в надсиланні запитів до бекенду та отриманні відповідей, які використовуються для відображення необхідної інформації користувачу.

Бекенд – це все, що працює на сервері. Виходячи з цього, бекенд розробка – це робота над програмними засобами, спрямованими на реалізацію логіки ресурсу. Ця частина прихована від очей користувача, оскільки відбувається за межами його браузера або конкретно взятого комп'ютера. Бекенд являє собою процес об'єднання користувача з сервером, який неможливо відстежити неозброєним поглядом [7].

Бекенд відповідає за обробку бізнес-логіки та їх даних у веборієнтованих програмних системах. Для цього використовуються серверні мови програмування (наприклад, Node.js, Python, Java, PHP) а також бази даних (наприклад, MySQL, PostgreSQL, MongoDB) та фреймворки – Express.js, Django, Spring.

Розподіл на два окремі потоки дає можливість командам працювати паралельно, що значно прискорює розробку і спрощує масштабування проєкту.

Для взаємодії між фронтендом та бекендом використовуються HTTP-запити та HTTP-повідомлення через API. Це дозволяє фронтенду, який представляє інтерфейс для користувача, обмінюватися з даними з бекендом (серверною частиною), що відповідає за обробку інформації, забезпечуючи ефективну комунікацію між цими складовими.

Коли користувач відкриває вебзастосунок та, наприклад, натискає на кнопку «Показати профіль», то браузер (фронтенд) надсилає запит до серверу (бекенд), зазнаючи на необхідність отримання даних про обраного користувача. Сервер обробляє цей запит та знаходить відповідну інформацію

у базі даних, де формує відповідь і повертає її у вигляді тексту або структурованих даних (JSON), де фронтенд отримує цю відповідь та відображає відповідну інформацію на екрані. Такий обмін даними відбувається безперервно, наприклад, від введення логіна і пароля до додавання товару в кошик, або надсилання повідомлення у чаті чи перегляду стрічки новин. Усі ці дії є серією HTTP-запитів, що забезпечують передачу інформації між компонентами вебзастосунку.

Особливості розгортання вебзастосунку для експлуатації полягають у тому, що це процес, під час якого готовий продукт стає доступним для користувачів. Спершу код ретельно перевіряють та тестують, або впевнитися у його коректній роботі без помилок, після цього його розгортають на сервері – спеціалізованому комп’ютері, що обробляє запити від користувачів і надає їм необхідну інформацію.

Для реалізації цього процесу використовують хмарні платформи або власні сервери. Для забезпечення безпеки застосовується шифрування HTTPS і відбувається захист даних користувачів. Важливо також налаштувати автоматичні оновлення, щоб оперативно виправляти помилки і впроваджувати нові функції без перерв у роботі вебзастосунку. Після розгортання вебзастосунка необхідно постійно контролювати його роботу, стежити за швидкістю, усувати можливі збої та регулярно оновлювати систему, щоб вона залишалася стабільною, зручною, безпечною для користувачів.

1.2 Аналіз методологій та програмних засобів управління проектом з розробки вебзастосунків

У розробці вебзастосунків найбільш популярними є Agile-методології. Їх популярність пояснюється високою гнучкістю, здатністю оперативно адаптуватися до змін і швидко реагувати на потреби замовників і вимоги

ринку.

Традиційні підходи, наприклад, Waterfall, не завжди є ефективним для вебзастосунків, оскільки передбачають сувору послідовність етапів проєктування і ускладнюють процес внесення змін. У веброзробці часто виникає потреба у швидкому коригуванні, наприклад, змінюються бізнес-вимоги, технології або надходить зворотній зв’язок від користувачів проєкту, який важливо враховувати.

Agile — це методологія гнучкої розробки, яка першочергово сьогодні популярна в IT і дозволяє клієнтам швидше отримувати якісне програмне забезпечення та сукупність підходів і моделей поведінки, орієнтованих на використання ітеративної розробки, time boxes (часових рамок), динамічне формулювання вимог і забезпечення реалізації ПЗ в результаті взаємодії всередині високо самоорганізованої робочої групи із фахівців різних профілів [8].

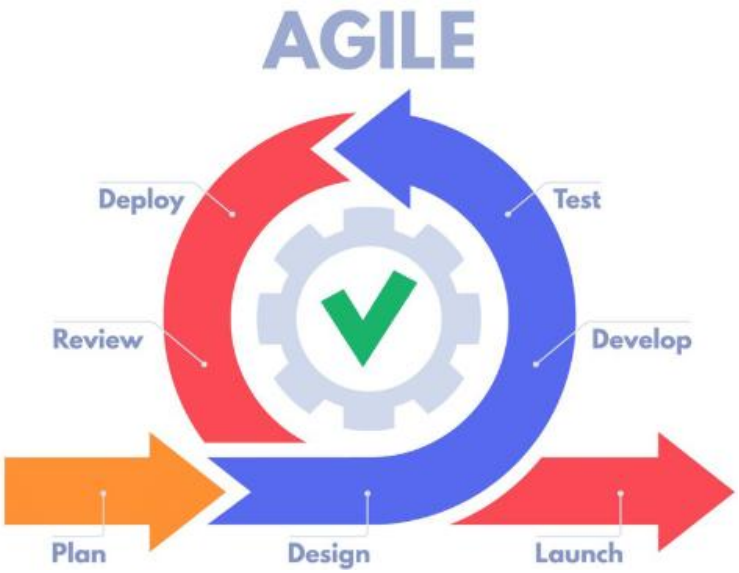

Рисунок 1.1 – Механізм методології Agile

Agile підхід дозволяє розробляти вебзастосунок поступово, розділяючи процес на короткі цикли, тобто спринти, де після кожного спринту досягається проміжний результат, який надає змогу оцінити прогрес та адаптувати подальший розвиток проєкту. Це дає можливість швидко запустити MVP

(мінімально життєздатний продукт) та продовжувати його вдосконаленню поетапно. Такий підхід також сприяє покращенню взаємодії між командами розробників, дизайнерів, тестувальників та замовників завдяки регулярним обговоренням прогресу проєкта й внесенню необхідних змін.

Ще однією та значною перевагою Agile є постійне тестування та раннє виявлення помилок, шо суттєво знижує витрати на їхнє виправлення. Крім того, Agile – допомагає впроваджувати крос-функціональний підхід, коли розробка фронтенду та бекенду здійснюється одночасно. Цей процес дає змогу швидше створювати функціональні компоненти вебзастосунку. Отже, Agile – є оптимальним для вебзастосунків, оскільки забезпечує можливість швидко реагувати на зміни, підвищую ефективність командної взаємодії, сприяє регулярному випуску оновлень та дозволяє створювати проєкти, які відповідають потребам користувачів.

Scrum – це одна з практичних методик Agile. Це не просто модне слово; це чітка схема для управління проектами. За своєю суттю Scrum – це легкий, ітеративний та поступовий підхід до розробки продукту [9]. Це адаптивна методологія управління проєктами, яка передбачає організацію роботи команди і короткі ітерації (спринти), тривалістю від 1 до 4 тижнів.

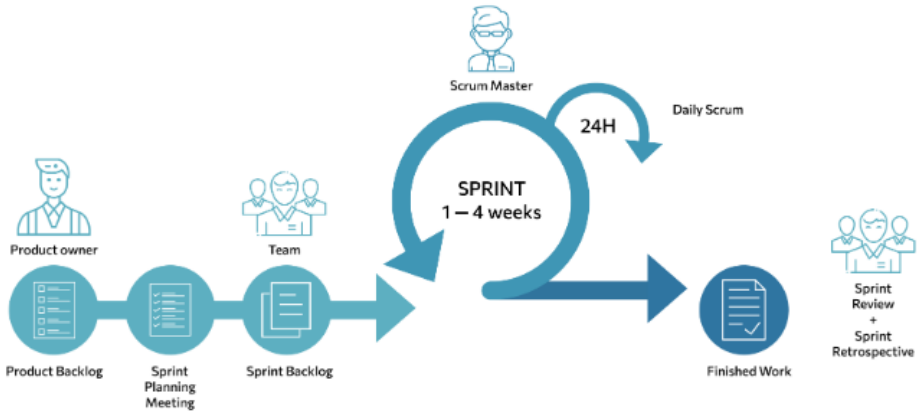

Рисунок 1.2 – Приклад принципової схеми для Scrum-методики

Під час спринту, виконуються найменші одиниці роботи – User Story, Task. User Story формуються шляхом декомпозиції батьківських функцій, коли

Task – є завданням технічної спрямованості. Команда працює автономно та регулярно проводить зустрічі, включно зі щоденними стендапами, сесіями планування та ретроспективами. По завершенню кожного спринту надається готовий та функціональний проєкт.

Ключові ролі Scrum складаються з:

- Scrum Master (координатор процесу);
- Product Owner (відповідає за бачення проєкту);
- Scrum Team (розробники, дизайнери, тестувальники тощо).

Ще одна практична методіка Agile – Nexus. Вона спрямована на вирішення кількох основних проблем у масштабованих середовищах Scrum, включаючи керування залежностями, забезпечення ефективного спілкування та координації між командами та інтеграцію роботи в єдиний продукт [10]. Також призначена для роботи великих команд, коли над одним проєктом одночасно працюють кілька Scrum-команд (зазвичай від 3 до 9). Вона запроваджує додатковий рівень координації, щоб запобігти труднощам з інтеграцією результатів різних команд.

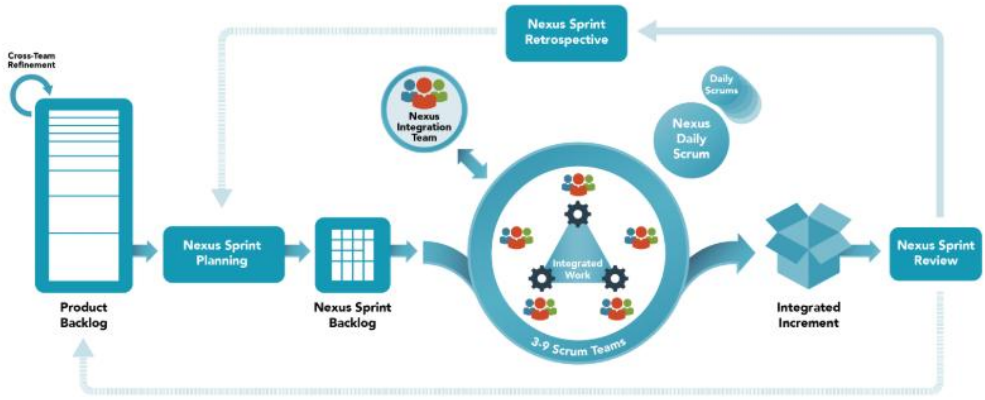

Рисунок 1.3 – Ілюстрація масштабування Scrum за допомогою Nexus

LeSS (Large Scale Scrum) – це набір правил та рекомендацій про те, як застосовувати Scrum у багатокомандному оточенні [11]. Це масштабний підхід Scrum, який дає змогу великій кількості команд (від двох до десятків) працювати над одним продуктом. На відміну від Nexus, LeSS – включає менше додаткових ролей та процесів, зберігаючи ключові елементи

класичного Scrum і забезпечуючи командам більше автономії. Усі команди використовують спільний беклог продукту і разом беруть участь у плануванні спринтів.

SAFe (Scaled Agile Framework) – це онлайн-база знань, що складається з доведених, інтегрованих принципів, практик і компетенцій впровадження Lean, Agile і DevOps при масштабуванні. Вона найбільш формалізована методологія для масштабування Agile у великих організаціях [12]. Вона об'єднує підходи Scrum, Kanban та Lean, створюючи при цьому чітку структуру для організації роботи на всіх рівнях компанії (від команд до керівництва). SAFe розподіляє завдання між Agile-командами і встановлює ключові ролі, такі як Product Management, Release Train Engineer, а також застосовує синхронізовані релізи для управління масштабними проєктами.

У таблиці 1.1 представлене порівняння найсучасних методик Agile.

Таблиця 1.1 – Порівняння методик Agile

Scrum	Nexus	LeSS	SAFe
Застосовується для малих команд, які спільно працюють над єдиним проєктом	Застосовується у випадках, коли кілька команд працюють разом над спільним проєктом, що вимагає залучення інтеграційної команди	Спрощена версія масштабованого Scrum, підходить для великих та середніх команд	Застосовується для масштабованих та великих організацій, які вимагають ефективної координації між командами та управлінськими рівнями

Після ознайомлення з методологією Agile, наступним кроком буде ознайомлення про розуміння систем управління IT-проєктами. Вони мають важливе значення у створенні та розробці вебзастосунків, оскільки сприяють координації роботи команди, відстеження прогресу, ефективному розподілу завдань та дотриманню встановлених строків. Такі інструменти роблять процес розробки прозорим, полегшують співпрацю між учасниками проєкту і

забезпечують успішне впровадження Agile-методологій.

Jira — це система управління проєктами, яка дозволяє закривати майже всі завдання PM-а в рамках одного інструмента: від планування до контролю процесів та результатів [13]. Вона орієнтована на проєкти в рамках Agile та підтримує використання Scrum, Kanban і гібридні підходи. Jira дозволяє формувати беклог завдань, призначати їх членам команди, налаштовувати та візуалізовувати робочі процеси виконання проєкту за допомогою діаграм та дашбордів. Однією з ключових переваг Jira – гнучкість у конфігурації, інтеграції з іншими інструментами (Bitbucket, Slack, Confluence) а також можливість автоматизації рутинних процесів.

Trello – найкращий інструмент для виконання повсякденної роботи. Керування щоденними завданнями, визначення особистих цілей і впорядкування справ у різних сферах життя [14]. Ця система є ефективним і зручним інструментом для управління завданнями, котрі базуються на використанні Kanban-дошок. Її функціонал ідеально підходить для роботи невеликих команд, або для реалізації проєктів з адаптивним та гнучким плануванням, де надзвичайно важливо відстежувати статус виконання завдань проєкту.

Azure DevOps підтримує культуру спільної роботи та набір процесів, які об’єднують розробників, керівників проектів і учасників для розробки програмного забезпечення. Це дозволяє організаціям створювати та вдосконалювати продукти швидше, ніж за допомогою традиційних підходів до розробки програмного забезпечення [15]. Це платформа, яка створена Microsoft, що об’єднує в собі систему керування завданнями, функції контролю версій та інструменти для безперервної інтеграції та розгортання (CI/CD). Вона є популярним вибором для реалізації масштабованих проєктів у корпоративному середовищі.

Після глибшого ознайомлення із системи управління IT-проєктами, варто детально розглянути можливості таких інструментів, як Jira, Trello та Azure DevOps. У подальшому аналізі проводиться їх порівняння (табл 1.2),

зокрема щодо функціоналу, який дозволяє встановлювати та візуалізовувати горизонтальні зв’язки між функціями.

Таблиця 1.2 – Аналіз функціональних можливостей Jira, Trello та Azure DevOps

Jira	Trello	Azure DevOps
Найкращій та оптимальний вибір для реалізації складних проєктів із значною кількістю взаємопов’язаних функцій, що забезпечується завдяки гнучким зв’язкам між задачами та картками	Підходить для невеликих команд, більш простий у використанні	Ідеально підходить для великих IT-команд, надає можливість забезпечувати детальний контроль над зв’язками між функціями та легко інтегрується з DevOps-процесами

Отже, можна зробити такий висновок, що:

– Jira пропонує ефективні інструменти для встановлення горизонтальних зв’язків між функціями та завданнями. Основою цього є система зв’язків між завданнями (Issue Linking), яка дозволяє визначити та задавати типи відносин між задачами. Додатково, Jira підтримує епіки, юзерсторі та підзадачі, що сприяє кращій організації роботи та забезпечує цим самим чітке розуміння взаємодії між функціями;

– Trello пропонує більш простий підхід до відображення зв’язків між функціями. Завдяки можливості створень посилань між картками, завдання проєкту можна легко пов’язувати між собою. Дошки можна налаштувати таким чином, щоб кожна окрема функція вебзастосунку проєкту мала власний список, де всередині карток зберігалися зв’язки із пов’язаними задачами;

– Azure DevOps надає розширений функціонал для встановлення та підтримки взаємозв’язків між різними функціями [16].

1.3 Аналіз мережевих моделей опису структури робіт проєкту

Аналіз мережевих моделей становить важливий аспект в управлінні IT-проєктами. Він надає змогу оцінити логічну структуру, часові характеристики виконання робіт проєкту. Завдяки мережевим моделям можна визначити ключові залежності, знайти критичний шлях та розрахувати резерви часу, що сприяє більш точному плануванню та ефективнішому контролю реалізації проєкту.

Під час планування розробки вебзастосунку надзвичайно важливим є правильний вибір мережевої моделі, яка слугуватиме основою для взаємодії між компонентами проєкту. Від обраної моделі залежить продуктивність системи, її здатність до масштабування, надійність роботи в проєкті та рівень інформаційної безпеки вебзастосунку.

Мережеві моделі для опису структури робіт проєкту базуються на представлені проєкту як орієнтованого графа. У такому графі вузли, які позначають події чи роботи, а також дуги, що відображають залежності між ними – забезпечують наочне уявлення зв’язків і взаємодій між роботами. Є два основні підходи до їх побудови [17]:

– модель із роботами на вузлах (Activity on Node, AON) – є найбільш поширена в сучасних IT-проєктах, де роботи розташовані у вигляді вузлів графа, а самі зв’язки між ними виконують функцію визначення послідовності та порядку реалізації завдань;

– модель із роботами на дугах (Activity on Arrow, AOA або PERT) – належить до більш традиційних методів планування, у якому роботи розташовані на дугах, а вузли символізують події, які позначають початок або завершення етапів проєкту.

Для опису робіт IT-проєктів переважно застосовують AON-моделі через їхню наочність, простоту моделювання та відмінну інтеграцію з інструментами управління (наприклад, MS Project, Jira).

Основними етапами аналізу мережевих моделей становить:

- формалізація структури проєкту;
- розрахунок часових параметрів;
- визначення критичного шляху;
- аналіз резервів часу;
- оцінка можливих ризиків і варіантів оптимізації.

Функції аналізу мережевих моделей у вебзастосунках ІТ-проєкту:

- контроль термінів;
- підтримка прийняття рішень;
- адаптація до змін;
- оптимізація ресурсів.

Можна зазначити, що аналіз мережевих моделей опису структури робіт – це один з основних інструментів для забезпечення контрольованості ІТ-проєкту. Тобто, цей аналіз дозволяє не лише візуально відобразити структуру вебзастосунку, а також глибше зрозуміти взаємозалежності, ризики та потенціал для покращення управління часом і ресурсами, де в будь який час це буде актуально та необхідно для успішної реалізації проєкту [18].

1.4 Постановка задач магістерської кваліфікаційної роботи

Об’єктом дослідження в рамках магістерської кваліфікаційної роботи є методи та моделі створення мережевої структури робіт ІТ-проєкту веборієнтованої програмної системи.

Предметом дослідження є особливості мережевого моделювання робіт для різних рівнів декомпозиції з метою опису та відображення зв’язків між роботами одного і того ж рівня декомпозиції, наприклад між User Story.

Проблемою дослідження є відсутність в системах керування ІТ-проєктами можливостей зручного встановлення горизонтальних зв’язків між

роботами. Виявлення та врахування таких зв’язків є дуже важливим завданням, тому що вони впливають на планування робіт як в масштабі всього проєкту, так і в масштабу чергового спринту. У теперішній час виявлення горизонтальних зв’язків між найменшими одиницями роботи в гнучкому середовищі виконується вручну та між командним органом (відповідно, наприклад, до методики Nexus) з чималими витратами робочого часу [19].

Метою дослідження є апробація застосування методу мережевого моделювання для виявлення та опису горизонтальних зв’язків між одиницями робот на кожному рівні декомпозиції в умовах використання методології Scrum для керування проєктом.

При виконанні магістерської кваліфікаційної роботи для досягнення поставленої мети потрібно вирішити такі задачі дослідження:

− провести аналіз особливостей розробки веборієнтованих програмних систем;

− провести аналіз методологій та програмних засобів управління ІТ-проєктом з розробки вебзастосунків;

− дослідити особливості використання методів мережевого моделювання для опису структури робіт ІТ-проєкту та вибрати тип мережевої моделі і форми її подання;

− розробити підхід до застосування мережевого моделювання для ефективного планування робіт ІТ-проєкту на різних рівнях декомпозиції для умов управління проєктом за методологією Scrum;

− розробити тезаурус для конкретного ІТ-проєкту веборієнтованої програмної системи;

− виконати експериментальну перевірку методу мережевого моделювання під час управління ІТ-проєктом вебзастосунку за методологією Scrum.

2 ДОСЛІЖЕННЯ МЕТОДУ МЕРЕЖЕВОГО МОДЕЛЮВАННЯ СТРУКТУРИ РОБІТ ПРОЄКТУ

2.1 Підхід до опису зв’язків між функціями одного рівня декомпозиції

Для побудови мережевої структури робіт ІТ-проєкту, а саме у рамках дослідження методу мережевого моделювання, ключовим завданням виступає формування ієрархічної структури робіт та встановлення логічних зв’язків між роботами одного рівня декомпозиції. В свою чергу, такі зв'язки забезпечують технологічну та логічну послідовність виконання завдань, що є основою для побудови мережевої коректної моделі реалізації проєкту.

Роботи одного рівня, які є в декомпозиції – це насамперед структурні компоненти, які реалізують частини підсистеми або єдиного модуля, проте не можуть бути підпорядковуватися одна одній в ієрархічному сенсі. Тобто, якщо розглядати у контексті веборієнтованих програмних систем, то це можуть бути, наприклад, окремі задачі, які тісно пов'язані з серверною логікою, з клієнтською частиною або з роботою з базами даних, які можуть реалізовуватись паралельно або з визначеним порядком залежностей [20].

Для набуття завершеної логіки, щоб структура проєкту могла бути у вигляді керованої моделі, усі взаємозв’язки повинні бути формалізовані між функціями. А саме, це означає, що для кожного зв’язку слід визначити таки характеристики:

- взаємозалежні функції;
- тип залежності;
- пріоритет або критичність виконання;
- параметри перекриття або затримки (lag/lead).

В таблиці 2.1, описано типи логічних зв’язків котрі найчастіше використовуються в мережевому моделюванні.

Таблиця 2.1 – Типи логічних зв’язків в мережевому моделюванні

Тип зв’язку	Позначення	Опис
Start to Start	SS	Наступна функція починається одночасно або з затримкою після попередньої
Finish to Start	FS	Функція, яка буде наступною, починається після завершення попередньої
Start to Finish	SF	Можливе завершення функції тільки після початку іншої
Finish to Finish	FF	Можливе завершення функції лише після завершення попередньої

Слід зазначити, що кожен із цих зв'язків може мати випередження (lead) або затримку (lag). Це може статися, наприклад, при плануванні залежностей між бекендом та фронтендом, або між розгортанням проєкту та його тестуванням.

Зв’язки, які були наведені в таблиці 2, фіксують взаємозалежність між окремими елементами структури робіт, але які знаходяться на одному рівні декомпозиції. Самі по собі вони не можуть існувати у відриві один одного. Тобто, це означає, що роботи одного рівня декомпозиції повинні та можуть бути виконі послідовно, паралельно, або пов’язані завершенням чи початком інших функцій. Усі ці варіанти зв’язків потребують чіткої формалізації, а саме опису у вигляді логічних типів зв’язків.

Опис зв’язків між роботами одного рівня декомпозиції є критичним з кількох причин, наприклад:

− однозначність інтерпретації логіки. Це означає, що учасники проєкту повинні розуміти послідовність виконання однаково, що саме буде мінімізувати ризик помилок;

− можливість автоматизації. Зв’язки дозволяють інтегрувати модель проєкту у програмне забезпечення для подальшого управління та моніторингу

(наприклад у середовища Jira, MS Project та ін.);

− побудова мережевої моделі (зв’язки можуть формувати основу для створення сіткового графа (AON), де функції є вузлами, а ті самі зв’язки являються спрямованими дугами);

− аналіз і оптимізація. Це допомагає виявити дублювання та залежності, котрі обмежують гнучкість при створенні логіки реалізації зв'язків.

Опис зв’язків між роботами одного рівня декомпозиції не другорядною процедурою, а, навпаки, – виявляється головним компонентом процесу перетворення структури робіт в ту саму мережеву модель котра нам потрібна. Щоб здійснити правильне та ефективне прогнозування тривалості, або ефективне планування чи розрахунок критичного шляху, потрібно встановити зв’язки між роботами одного рівня декомпозиції. Ці всі дії будуть забезпечувати збалансовану синхронізацію проєкту, де приймаються обґрунтовані управлінські рішення на основі об’єктивних даних проєкту.

2.2 Метод побудови та аналізу мережевої структури робіт IT-проєкту

У процесі побудови мережевої структури робот IT-проєкту, слід зазначити, що вона має допомогти в вирішенні таких завдань:

− встановлення послідовності виконання завдань;

− виявлення критичних точок;

− прогнозування затримок та організація розкладу проєкту;

− розрахунок тривалості проєкту.

Усі ці завдання можна вирішити за допомогою методу мережевого моделювання, який передбачає представлення проєкту у вигляді орієнтованого графа.

В нашому випадку це буде модель Activities on Node (AON), де дані відображаються у вигляді вузлів, а зв’язки між ними – у вигляді стрілок [21].

Цей метод передбачає використання в орієнтованому графу таких позначень:

- кожна робота, тобто activity зображується у вигляді вузла (прямокутника);
- стрілка між вузлами означає логічну залежність між роботами;
- кожен зв’язок має типи, які було наведено в таблиці 2, а саме FS, SS, FF, SF і може мати затримку (lag).

Дана модель Activities on Node відрізняється від класичної моделі PERT (AOA) тим, що робота зображується стрілкою, а самі події – вузлами. Для кращого розуміння різниці між моделями типу AON та AOA був зроблений їх детальний порівняльний аналіз (див. таблицю 2.2):

Таблиця 2.2 – Порівняльний аналіз моделей AON та AOA

Критерій	Модель Activity on Node (AON)	Модель PERT (AOA)
Основний елемент	Робота у вузлі	Робота на стрілці
Вузол	Робота (задача або дія)	Подія (початок або кінець)
Стрілка	Відображає залежність	Відображає саму роботу
Гнучкість типів залежностей	FS, SS, FF, SF та lag і lead	Зазвичай тільки FS, та обмежена підтримка інших залежностей
Логіка моделі	Зв’язки між роботами	Відображення події як зміна стану проєкту
Підтримка програмного забезпечення	Повна (MS project, Jira тощо)	Часткова або в більшості систем відсутня
Наочність	Зручна для використання командами, інтуїтивно зрозуміла	Менш зрозуміла та часткова наочна, потребує досвіду команди
Підходить для	Паралельних, гнучких, змінних середовищ	Інженерні, виробничі або промислові проєкти

Кінець таблиці 2.2

Критерій	Модель Activity on Node (AON)	Модель PERT (AOA)
Використання в IT-проєктах	Популярне та поширене	Дуже рідко використовується
Складність побудови	Простіше для новачків, початківців	Вимагає більше формальностей та складніше
Потреба у фіктивних роботах	Немає	Часто потребує

Після відображення таблиці 2.2 зверху, можна зробити невеликі висновки основних відмінностей, що у моделі AON (Activities on Node) – роботи зображуються як вузли, а самі стрілки вказують на логічні зв’язки між ними. Ця модель мережевого моделювання структури IT-проєкту є найбільше поширеніша в останній час та має більш сучасний підхід з підтримкою MS Project, Jira тощо.

Метод PERT/AOA переважно застосовується здебільше в виробничих, інженерних та великих будівельних проєктах, де важлива тільки подієва логіка.

У сфері веборієнтованих програмних систем для управління IT-проєктами метод AON є більш універсальним та зручним, тому що він:

- краще підтримує та відображає складну логіку залежностей;
- дозволяє без всяких важких зусиль інтегрувати модель у популярні інструменти управління IT-проєктами;
- гнучкий у плануванні;
- легко масштабується (деталізує модель на різних рівнях, від високорівневої декомпозиції (епіки, фічі) до рівня окремих задач);
- відображає послідовність робіт.

Для моделі типу AON дуже важливими є позначення , які використовуються у вузлах моделі. Вони є фундаментальними інструментами

аналізу, управління проєктом та планування. Перелік позначень для вузлів моделі наведений у таблиці 2.3.

Таблиця 2.3 – Умовні позначення вузлів моделі AON

Позначення	Назва	Опис
ID	Ідентифікатор роботи	Код завдання (наприклад A1, B3)
Назва	Опис роботи	Короткий опис суті функції
FF	Free Float	Вільний резерв часу (наскільки часу можна зсунути роботу, щоб не вплинувши на наступну)
TF	Total Float	Повний резерв часу (LF – EF) або (LS – ES)
LF	Late Finish	Найпізніше можливе завершення
LS	Late Start	Найпізніший початок, котрий допустимий без впливу на графік
EF	Early Finish	Найраніше можливе завершення (EF = ES + Dur)
ES	Early Start	Найраніший можливий початок
Dur	Duration	Тривалість роботи (години/дні)

Ці позначення дозволяють перетворити просту схему залежностей на більш потужну математичну модель, яка здатна точно та без похибки оцінювати строки, розраховувати резерви часу, виявляти вузькі місця та формувати графік виконання завдань. Без цього мережева структура робіт IT-проєкту втрачає свою аналітичну силу та згодом перетворюється на подальший неструктурований перелік завдань.

Мережева модель типу AON може бути представлена в різних формах подання. Перелік та опис цих форм наведено в таблиці 2.4.

Таблиця 2.4 – Форми подання мережевої моделі AON

Форма подання	Опис	Переваги
Графічна (мережевий графік)	Візуальне зображення стрілок і вузлів у вигляді мережевого графа	Візуалізація, інтуїтивність, виявлення критичного шляху
Таблична (аналітична)	Дані подаються у вигляді таблиці з усіма параметрами: EF, ES, LS, TF,LF, тривалість, тип зв’язку	Має зручності для документування та зручно для розрахунків
Матрична (таблиця суміжності)	Показує наявність зв’язку, має матрицю залежностей 0 або 1 для кожної пари	Має зручності для автоматизованої обробки
Програмна (машинно-читабельна)	Використовуються формати XML, CVS, JSON для автоматизації, імпорту даних з ПЗ, обміну даними	Має підтримку великих проєктів
Календарно-діаграмна (діаграма Ганта)	Показує тривалість, резерви часу, зв’язки, має часову шкалу для графіку виконання задач	Має можливість показати реальні строки проєкту, Є змога щоб управляти ресурсами
Онтологічна, концептуальна (UML)	UML діаграми, котрі відображають логіки проєкту в контексті бізнес процесу або системного аналізу (діаграма діяльності, компонентів, послідовності)	Можливість використовувати архітектурні рішення для моделювання процесів

Форми подання моделі AON існують для того, щоб в різних способах можна було відобразити візуально, таблично або у вигляді цифрового представлення логіку виконання робіт у самому проєкті. Кожна форма подання унікальна та неповторима із-за того, що має свою специфічну мету і буде застосовуватись залежно від потреб проєкту, наприклад:

- аналіз даних;
- комунікація в команді роботи проєкту;

- документування процесу планування проєкту;
- автоматизація або презентація проєкту.

Правильне використання цих форм є найважливішою та необхідною однієї із складовою для ефективного проектування та управління ІТ-проєктом, зокрема для розробки веборієнтованих програмних систем.

Математичний апарат для аналізу та оптимізації виконання робіт, який лежить в основі мережевого моделювання типу Activities on Node, має забезпечити виконання таких дій:

- будування візуальної структури проєкту;
- визначення критичних елементів (вузлів, які не мають резервів часу і формують цим критичний шлях);
- приймання управлінських рішень для оптимізації плану виконання ІТ-проєкту;
- виконання глибоких та аналітичних розрахунків часових характеристик;
- оцінювання резервів часу.

Важливу роль в цій мережевій моделі відіграє формалізація. Формалізація мережевої моделі складається з того, що з логічного та математичного перетворення структури робіт проєкту котрі в нас є, ми отримуємо орієнтований ациклічний граф. Структура такої моделі описується формулою (1).

$$\text{Граф } G = (V, E), \tag{1}$$

де V – множина вершин, які відповідають роботам;

E – множина дуг (ребер), які відображають залежності між роботами.

Для того щоб цей граф був коректним, він повинен задовольняти таким умовам:

- мати визначені кінцеві та початкові вершини;
- усі роботи повинні виконуватися в логічній послідовності, тобто бути

топологічно впорядкованими;

- не мати циклів, а саме не можливо щоб робота залежала сама від себе;
- мати забезпечувати декомпозиції (вершини графа можуть відповідати дочірнім мережевим елементам моделі).

При розгляданні математичного апарату важливим є також алгоритм розрахунку часових параметрів, тому що він є ключовим інструментом в управління часом для проєкту. Математична основа цього алгоритму, базується на тому, що при аналізі та розрахунку орієнтованого ациклічного графу він дозволяє відображати взаємозв'язки, послідовність та часові обмеження між роботами проєкту, якщо вони існують. Основна перевага цього алгоритму складається в тому, що можливі зміни керування проєктом в реальному часі. Завдяки цьому отримуємо повну прозорість під час планування проєкту, тобто дозволяючи виявити не тільки найраніші але й найпізніші допустимі терміни виконання проєкту та визначити резерви часу, які є допустимі при необхідності та також критичні моменти роботи.

Наступним кроком, перерахуємо основні причини, чому саме цей алгоритм необхідний у використанні під час аналізу та оптимізації виконання робіт:

- визначення мінімальної тривалості проєкту (кожна робота, тобто вузол в AON, який відображається позначками ES і EF (найраніші дати та початок завершення проєкту) має обчислюватись для встановлення строку завершення проєкту);
- виявлення резервів часу проєкту (позначки FF та TF, які вказують на гнучкість у графіку, цим самим можуть забезпечувати при необхідності перепланування без впливу на загальний строк проєкту);
- побудова критичного шляху (виявлення робіт, де позначка TF = 0 (нульовий резерв часу) дає нам змогу у визначенні найвразливіших етапів планування, де затримка проєкту не є допустимою);
- планування за допомогою оптимізації проєкту;
- цифрове управління за допомогою автоматизації (можливість

використання MS Project, Jira та інших інструментів для управління проєктами).

З цього списку, який відобразили вище, розглянемо більш детально резерв часу та поняття критичного шляху проєкту.

Резерв часу – це та кількість часу в проєкті, котру можна відкласти на початок роботи або її завершення без всілякого негативного впливу на інші роботи проєкту. Це потрібно для того, наприклад, щоб знати які роботи в проєкті можна буде гнучко перепланувати, або зробити перерозподіл ресурсів без ризику, якщо є така необхідність. Також процес визначення резерву часу дає таку можливість, при якій можливо відокремити критичні роботи від менш важливих.

Існують дві формули для розрахунку резерву часу проєкту. Перша формула для визначення повного резерву часу (Total Float), яка має такий вигляд:

$$TF = LF - EF = LS - ES, \tag{2}$$

де LF = Late Finish (пізнє завершення);

EF = Early Finish (раннє завершення);

LS = Late Start (пізній початок);

ES = Early Start (ранній початок).

Друга формула призначена для розрахунку вільного резерву часу (Free Float), та має такий вигляд (3):

$$FF = \min\left(ES_{\text{наступника}}\right) - EF, \tag{3}$$

де $ES_{\text{наступника}}$ = Early Start (ранній початок);

EF = Early Finish (раннє завершення).

Критичний шлях проєкту – це найдовший за тривалістю маршрут з послідовністю робіт у мережевій моделі, що визначає мінімально можливий

час для завершення всього проєкту.

Критичний шлях розраховується за формулою (4):

$$TF = LF - EF = 0 \text{ або } TF = LS - ES = 0, \quad (4)$$

Під час розглядання алгоритму для розрахунку часових параметрів, потрібно приділити увагу цим двом фундаментальним процесам – Прямий прохід (Forward Pass, FS) та Зворотній прохід (Backward Pass, BP).

Прямий прохід (Forward Pass, FP) – цей процес важливий тим, що дозволяє робити обхід мережі проєкту зліва направо (від початкових робіт до завершальних) із метою розрахунку найраніших можливих дат до самого завершення кожної роботи проєкту. Цей процес потрібен для таких задач:

- визначення часових меж проєкту, коли робота має можливість початися без очікування;
- для розрахунку мінімальних можливостей тривалості усього проєкту;
- для бачення паралельності та послідовності виконання задач.

Зворотній прохід (Backward Pass, BP) – це процес обходу мережі справа наліво (від завершальних до початкових робіт) із метою визначення найпізніших допустимих строків проєкту та завершення кожної роботи. Цей процес потрібен для таких задач:

- для виявлення того, наскільки роботу в проєкту можна буде відкласти, щоб не порушувати загального графіка;
- для розрахунку резервів часу (Total Float, Free Float);
- для підготовки за потреби, якщо потрібно підготуватись до можливих затримок або варіантів перепланування проєкту.

При необхідності під час розрахунку часових параметрів, можна використати метод оптимізації, де описується стискання строків проєкту. Це необхідно для того, якщо замовник хоче змінити свої вимоги до проєкту, або сталися технічні ризики (затримки). Оптимізація строків – це процес, під час якого потрібно зробити загальне скорочення часу реалізації проєкту, але без

зменшення послідовності задач або збільшення ресурсів без впливу на кінцевий результат. Мета оптимізації полягає в тому, щоб:

- скоротити загальну тривалість проєкту;
- догнати втрачений час у випадку затримок проєкту;
- вкластися у нові дедлайни (якщо є необхідність в цьому під час спілкування з замовником проєкту);
- оптимізувати простої в роботі команди та використання ресурсів проєкту.

Існують два основні методи стискання строків:

- Crashing – це процес додавання ресурсів до критичних робіт проєкту за для скорочення їх тривалості (наприклад, до розробки проєкту потрібно додати ще одного працівника команди);
- Fast Tracking – це процес паралельного виконання робіт, які зазвичай виконують послідовно.

3 ВИКОРИСТАННЯ МЕТОДУ МЕРЕЖЕВОГО МОДЕЛЮВАННЯ ДЛЯ УПРАВЛІННЯ ІТ-ПРОЄКТОМ

3.1 Загальні положення

Мережеве моделювання для управління ІТ-проєктом веборієнтованої програмної системи бажано використовувати на всіх етапах життєвого проєктного циклу, тому що це дозволяє зробити формальний опис структури виконання робіт, виявити критичні ділянки проєкту, визначити послідовність виконання задач, оптимізувати хід виконання робіт проєкту та прорахувати строки.

Мережева модель доцільно буде використовувати для таких етапів:

− проєктування архітектури системи – коли потрібно визначити взаємозв’язки між елементами архітектури веборієнтованої програмної системи (наприклад, бекендом, фронтендом, базою даних);

− під час програмування веборієнтованої програмної системи – для відображення та показу залежностей між фічами проєкту (наприклад, підключення до серверної частини, реалізація функцій авторизації, інтеграція з базою даних, аналітики тощо);

− під час тестування (для планування тестових сценаріїв);

− технічна підтримка та оновлення (коли потрібно зробити зміни для проєкту без впливу на ключову логіку або інші частини веборієнтованої системи).

Після розгляду етапів, роздивимось тепер для чого потрібно використовувати мережеве моделювання в управлінні ІТ-проєктом:

− для розрахунку часових параметрів;

− для побудови структурної моделі робіт проєкту;

− для визначення критичного шляху;

− для планування ресурсів;

− для адаптації до змін у проєкті;

- для оптимізації строків виконання проєкту;
- для управління ризиками.

Загальний строк виконання проєкту – це той процес та показник, в якому береться найкоротший можливий час, який необхідний для завершення робіт проєкту та ураховується їхня послідовність, тривалість та залежності. Тобто, це той мінімальний час, який необхідний для завершення проєкту, якщо:

- немає затримок в роботі проєкту;
- обговорені всі ресурси для проєкту;
- роботи в проєкті починаються якнайшвидше відповідно до їхніх залежностей.

Загальний строк виконання проєкту визначається за допомогою мережевої моделі через знаходження та урахуванням критичного шляху. Якщо кажучи формально, то загальний строк буде дорівнювати тривалості критичного шляху. З цього можна навести такий приклад, де в нас буде 5 робіт з тривалістю: 2 + 2 + 4 + 1 + 1 = 10 днів, з цього прикладу розуміємо що загальний строк виконання проєкту буде дорівнювати 10 днів розробки. Цей показник більш всього потрібен для контролю робіт проєкту, а саме:

- для планування дати завершення проєкту;
- для порівняння плану проєкту з фактичним виконанням робіт;
- для контролю часу, щоб дивитися чи встигає команда за графіком;
- для оцінки впливу змін, щоб заздалегідь реагувати на затримки під час реалізації проєкту.

Під час мережевого моделювання, часто трапляються випадки, коли команда, яка працює над проєктом, не встигає реалізувати проєкт в загальний строк виконання, то тоді потрібно застосовувати такі дії:

- проводити аналіз причин затримки;
- застосовувати методи по скороченню строків проєкту (оптимізація);
- проаналізувати актуалізацію мережевої моделі;
- зробити перерозподіл навантаження;
- застосувати можливе сценарне планування та управління ризиками;

− проводити більше комунікації в роботі команди та контролю.

Вихід за межі загального строку в першу чергу – це критичний сигнал, але не є вироком. Проєкт, котрий базується та розробляється за допомогою мережевій моделі, завжди можна структурно проаналізувати, тим самим визначити точки впливу й оперативно скоригувати план, щодо вирішення проблем без усіляких втрат якості чи функціональності.

На рисунку 3.1 наведено схему алгоритма для контролю мережевої моделі під час виконання та управління ІТ-проєктом.

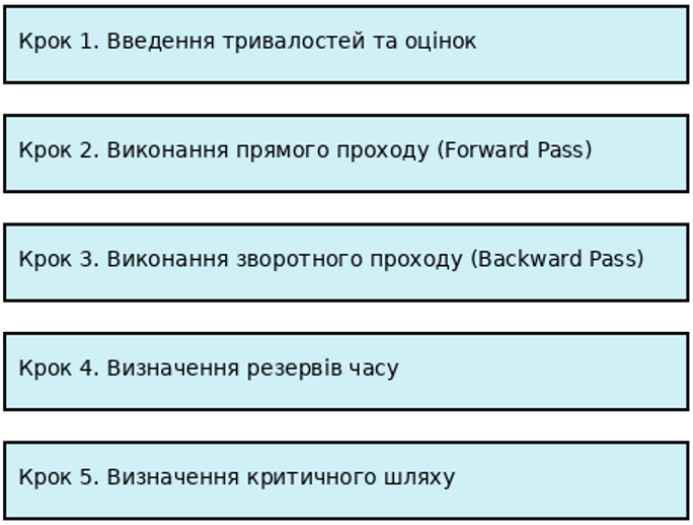

Рисунок 3.1 – Алгоритм для контролю мережевої моделі ІТ-проєкту

Розглянемо цей алгоритм більш детально.

Крок 1: введення тривалостей та оцінок, де визначається тривалість (у годинах, днях) та оцінка навантаження.

Крок 2: розрахування виконання прямого проходу (Forward Pass), де розраховуються – ES (найраніший можливий початок) та EF (найраніше завершення).

Крок 3: розрахунок виконання зворотного проходу (Backward Pass), де розраховуються – LF (найпізніше завершення) та LS (найпізніший допустимий початок).

Крок 4: процес визначення резервів часу, де TF (Total Float) – скільки

часу можливо витратити без впливу на графік проєкту, FF (Free Float) – скільки часу є до самого початку наступної задачі.

Крок 5: процес визначення критичного шляху проєкту, де виявляються всі роботи з TF = 0, де критичний шлях має визначати мінімальну тривалість проєкту.

Під час управління IT-проєктом за методологією Scrum доцільно будувати мережеві моделі для робіт таких рівнів декомпозиції [21]:

− для рівня епіків (у формі мережевого графіка);

− для рівня фіч (у формі мережевого графіка, якщо кількість фіч не перевищує 15 і формі таблиці в іншому випадку);

− для рівня User Story та Task, які знаходяться у Backlog проєкту (у формі таблиці);

− для рівня User Story та Task, які знаходяться у Backlog спринту команди (у формі діаграми Ганта).

3.2 Використання методу на рівні епіків

Використання мережевого моделювання на рівні епіків, де епік – це найвищий рівень функціональної декомпозиції в IT-проєкті, та виконує стратегічну функцію управління. Епік у веборієнтованих програмних системах зазвичай являє собою значну логічну складову проєкту, яка складається з кількох функціональних блоків або фіч, реалізація яких відбувається протягом декількох спринтів чи ітерацій [15].

Мережеве моделювання на рівні епіків потрібне для:

− структуризації логіки реалізації ключових модулів (можливість демонстрації, в якій послідовності необхідно реалізовувати масштабні та великі частики проєкту (наприклад, кабінет користувача, авторизація, реєстрація));

− процесу оцінки стратегічного критичного шляху (які великі функціональні блоки являються критичними для завершення проєкту);

− процесу планування залежностей між епіками (епіки неможливо почати без завершення інших епіків;

− процесу оптимізації розподілу ресурсів між командами (кожен епік може закріплюватися за окремою командою, де мережа допомагає визначити, коли та які епіки можливо виконувати одночасно);

− процесу визначення контрольних точок (закінчення епіку може слугувати проміжною метою для оцінки прогресу в розробці проєкту).

Мережева модель для епіків має форму представлення у вигляді мережевого графіка.

Результат котрий ми отримуємо за допомогою мережевої моделі для рівня епіків – це порядок реалізації стратегічних частин системи, очікуваний строк завершення кожного епіка, створення основи для подальшої деталізації на рівні функцій і завдань та розрахування критичного шляху.

Використання епіків дає змогу організувати високорівневе планування проєкту, відображування ключових залежностей та підготовлення фундаменту для детального управління проєктом на більш нижчих рівнях.

3.3 Використання методу на рівні фіч

На рівні фіч, тобто середньому рівні декомпозиції робіт, метод мережевого моделювання застосовується для детального планування всього проєкту та його координації (контролю) виконання логічних завершених частин системи, які входять до складу епіків. За допомогою мережевої моделі цього рівня, стає можливим більш точне управління залежностями, раціональне використання ресурсів та оцінювання строків.

Призначення мережевого моделювання на рівні фіч складається з:

− встановлення залежностей між фічами, визначення фіч які мають бути реалізовані в першу чергу та залежать одна від одної (наприклад, функція входу до сайту через пошту неможливо реалізувати до того, як буде завершено реєстрацію);

− формування внутрішньої структури епіків – передбачає, що поділ кожного епіку на фічі, реалізуються як окремі функціональні елементи (наприклад, існує епік реєстрації, де окрема фіча – це перевірка валідності символів);

− оцінки термінів реалізації фічей здійснюється з урахуванням їхньої технічної складності та доступних ресурсів (наприклад, спринтах або оцінка у людино-годинах);

− можливості виявлення критичних фічей або вузьких місць проєкту (визначення критичного шляху серед фічей, затримка яких може вплинути на завершення епіку або проєкту).

Мережева модель для рівня фіч подається у вигляді табличної форми, якщо кількість фіч у проєкті перевищує, наприклад 15. При великої кількості фіч мережевий графік втрачає свою наочність.

Застосування мережевої моделі для рівня фіч можливе для таких етапів:

− планування загальної тривалості IT-проєкту;

− планування інтеграції (фічі, які можуть бути об’єднані в один реліз);

− контроль прогресу проєкту (за допомогою Jira на основі залежностей);

Фічі дозволяють керувати функціональними наповнювачами системи проєкту, щоб запобігати збоїв у реалізації та гнучко реалізовувати та розподіляти ресурси між командами проєкту.

3.4 Використання епіків на рівні User Story та Task

Використання епіків на рівні User Story та Task – це є

найдеталізованіший рівень робіт, де мережевий метод моделювання виконую функцію оперативного планування проєкту й контролю щоденної розробки командою. Цей рівень забезпечує точну координацію дій в середині спринтів та швидке реагування на зміну пріоритетів в проєкті і зменшення ризиків у разі невиконання задач у встановлені терміни [17].

Мережеве моделювання на цьому рівні потрібно для:

− формалізації залежностей між задачами – встановлення чіткого порядку виконання кроків у рамках однієї User Story або фічі (наприклад, створення компонента, далі написання тестів і далі рефакторинг);

− підвищення ефективності спринтів (у коротких ітераціях, щоб завдання не заважали одне одному, виявлення конфліктів);

− забезпечення відстежуваності (коли User Story пов’язана з конкретноє фічею, то це дає змогу відстежувати прогрес задач проєкту);

− управління деталями імплементації (процес розподілу User Story на підзадачі, що дозволяє командам паралельно працювати над різними процесами проєкту).

Переваги використання епіків на рівні User Story та Task складаються з того, що це зумовлює підвищенню контрольованості плану, зменшення ризиків простоїв, точніший розрахунок завантаження команди.

Мережеву модель для рівня User Story та Task, які знаходяться у Backlog проєкту, можна використовувати для автоматичного запобігання можливого включення до чергового спринту ланцюжків взаємопов’язаних завдань, критичний шлях яких може перевищувати тривалість спринту. Опишемо алгоритм цього процесу.

Крок 1: потрібно розрахувати довжину спринту у сторі поінтах. Для цього потрібно розрахувати навантаження на поточний спринт для кожної посади за формулою (5).

$$SP_{\text{спринту}} = Velocity \times \text{коефіцієнт корекції}, \quad (5)$$

де $Velocity$ – середня продуктивність команди за попередні спринти [15].

Після цього кроку довжина спринта може бути визначена за допомогою нормативного мінімального навантаження серед різних посад (Nmin–Nmax).

Крок 2: вибір завдання для включення до Backlog спринту, яке в мережевій моделі для Backlog проєкту не є залежною роботою (з урахуванням поміток на завданнях про включення до спринту). Тобто, якщо всі попередні завдання в шляху, до якого належить поточне завдання, помічені, як такі, що включені до спринта, то завдання вважається незалежним.

Крок 3: виконується перевірка на наявність в моделі Backlog проєкту попередніх завдань, помічених як такі, що включені до спринта. Якщо це не так, виконується крок 2.

Крок 4: розраховується тривалість частини шляху в (сторі поінтах), до якого належить обране завдання, з його початку та включаючи це завдання. Якщо тривалість частини шляху може перевищувати мінімальну довжину спринту (Nmin), то тоді обране завдання не може бути включене до поточного спринту та потрібно перейти на крок 2.

Крок 5: на обраному завданні робиться примітка про включення до спринту.

Крок 6: якщо процес наповнення спринту завданнями закінчений, тоді всі завдання, які мають помітки про включення до спринту переносяться до Backlog спринта і разом з інформацією про взаємозв’язки з Backlog проєкта. В іншому разі перехід до кроку 2.

На рис. 3.2 наведено схема алгоритму контролю включення завдань до спринта з врахуванням їх взаємозв’язків у вигляду UML діаграми дій.

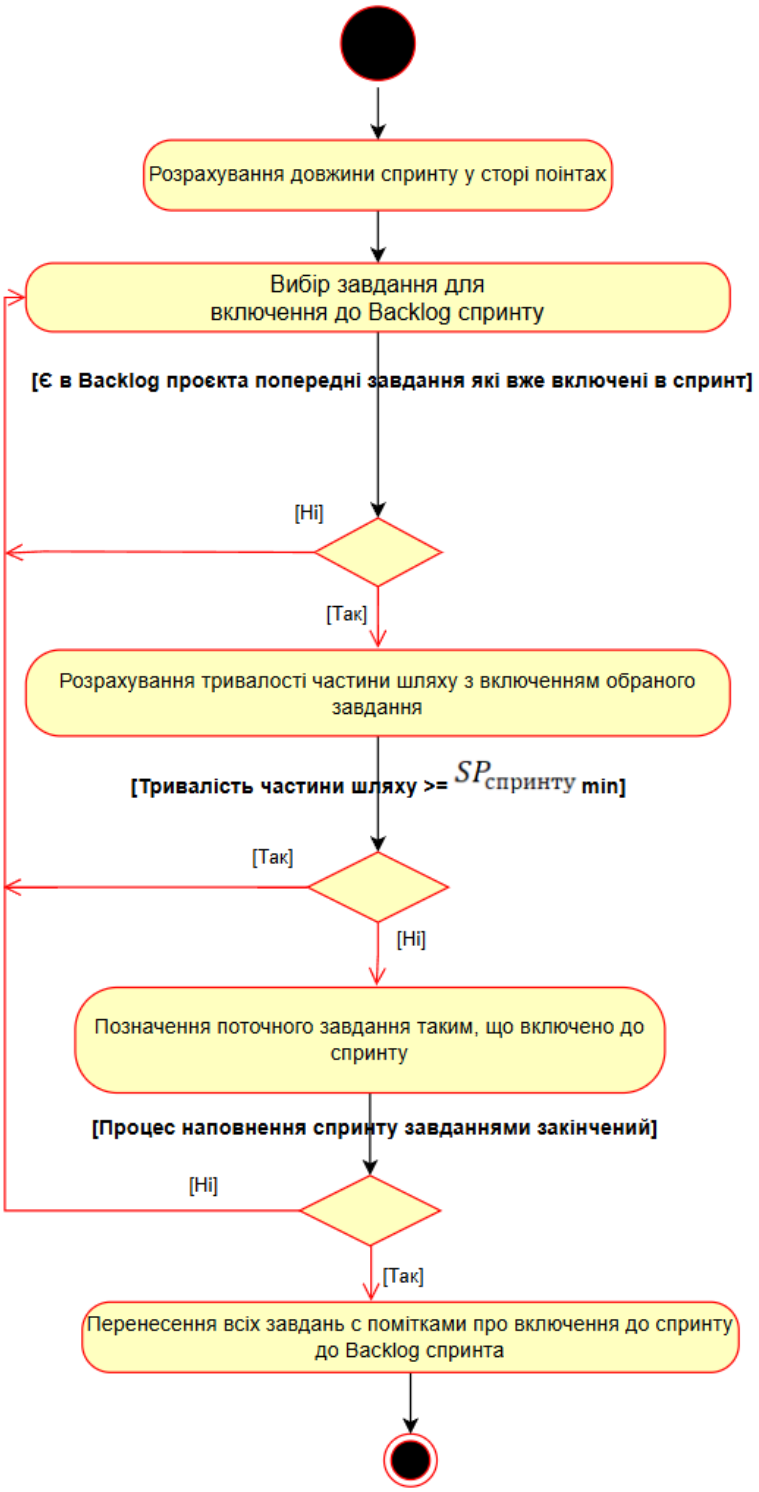

Рисунок 3.2 – Схема алгоритму контролю включення завдань до спринта з врахуванням взаємозв'язків між завданнями

Після закінчення формування Backlog спринта можна будувати мережеву модель для Backlog спринта команди у формі діаграми Ганта для

більш зручного відображення взаємозалежностей завдань поточного спринта. Така форма подання мережевої моделі для Backlog спринта команди більш прийнятна, тому що кількість завдань, які команда відбирає для виконання протягом спринта не дуже велика, тому діаграма буде наочна. До того ж на діаграмі Ганта можна побачити тривалість спринту та дати початку, закінчення виконання роботи, які завдання не вміщуються в поточний спринт, які роботи виконуються паралельно, хто виконавець роботи.

4 ЕКСПЕРЕМЕНТАЛЬНА ПЕРЕВІРКА МЕТОДУ МЕРЕЖЕВОГО МОДЕЛЮВАННЯ ПІД ЧАС УПРАВЛІННЯ ІТ-ПРОЄКТОМ ВЕБЗАСТОСУНКУ

4.1 Опис особливостей ІТ-проєкту вебзастосунку «Благодійні суботники»

Для проведення експериментальної перевірки методу мережевого моделювання під час управління ІТ-проєктом вебзастосунку були використані матеріали кваліфікаційної роботи, присвяченої розробці вебзастосунку «Благодійні суботники» [22]. Цей вебзастосунок призначений для автоматизації процесів управління благодійними суботниками. Об’єктом автоматизації виступає організація, котра організовує збори коштів і проведення заходів для благоустрою навколишнього середовища.

Особливості цього ІТ-проєкту включають такі ключові аспекти:

− розробку та використання вебзастосунку, призначеного для ефективної організації та управління всіма етапами реалізації благодійних заходів (охоплення етапів від початкового планування і збору фінансованих ресурсів для звітності за досягнутими результатами);

− наявність двох основних категорій користувачів: перша – менеджери, які адмініструють заходи, а друга – активні та ініціативні громадяни, котрі мають можливість та бажання робити пожертвування, пропанувати власні ініціативи та брати безпосередню участь у проведені заходів;

− інтеграцію з платіжними системами для забезпечення можливості обробки онлайн-платежів, а також із сервісами електронної скриньки для комунікації з користувачами;

− формування та реалізацію функціоналу, щодо розробки планів майбутніх суботників, ведення обліку про поточні заходи, а також створення архіву проведених подій із фіксацією про їх результат;

− автоматизацію процесів для економії часу, прозорості управління та

залучення ширшого кола учасників за допомогою зручного інтерфейсу вебзастосунка.

Таким чином можна зазначити, що особливістю цього ІТ-проєкту є поєднання соціальної місії в сукупності з сучасними вебтехнологіями, які дозволяють зручно та ефективно організувати заходи та забезпечити повну прозорість у благодійній діяльності.

4.2 Розробка тезаурусу ІТ-проєкта вебзастосунку

Для ефективного виконання будь-якого проєкту спочатку потрібно розробити тезаурус цього конкретного проєкта. В першу чергу тезаурус проєкту дозволяє використовувати одні і ті самі терміни на різних рівнях проєктування та розробки вебзастосунку. По-друге, він допомагає в поясненні термінів замовнику або новим членам команди та зумовлює більш досконалу комунікацію між менеджерами, розробниками на етапі проєктування та розробки проєкту.

Тезаурус проєкту має певний ряд плюсів:

- спрощення проєктування API та баз даних;
- приведення до єдиної назв компонентів та їх функцій;
- оптимізація процесів розробки та тестування;
- підтримка та масштабованість проєкту;
- полегшення роботи з документацією.

Тому першим кроком перед проведенням експериментальної перевірки методу мережевого моделювання під час управління ІТ-проєктом вебзастосунку був розроблений тезаурус ІТ-проєкту вебзастосунку «Благодійні суботники», який наведено в табл. 4.1.

Таблиця 4.1 – Тезаурус IT-проєкту вебзастосунку «Благодійні суботники»

Термін	Визначення, пояснення
Epic	Великий функціональний блок, який охоплює кілька фіч
Feature (Фіча)	Функціональна складова, яка реалізує частину епіку
User Story	Конкретне завдання або потреба користувача, яку необхідно реалізувати
Task (Завдання)	Технічна дія або робота, яку необхідно виконати для реалізації User Story
Backlog	Список усіх запланованих (необхідних) для реалізації завдань
Sprint	Чітко визначений часовий проміжок для виконання частини завдань із Backlog
Critical Path (Критичний шлях)	Найдовша послідовність залежних завдань, яка визначає мінімальну тривалість усього проєкту
Velocity	Середня продуктивність команди за один спринт, яка рахується у Story Point
Story Point	Умовна одиниця оцінки складності (обсягу роботи), яку необхідно виконати
AON (Activity on Node)	Метод мережевого моделювання
Backlog Project	Повний список завдань і User Story, котрі необхідно реалізувати в усьому проєкті
Backlog Sprint	Ряд завдань, які вибрані для виконання в конкретному спринті
Stakeholder (Зацікавлена особа)	Особа яка зацікавлена у результатах проєкту

Продовження таблиці 4.1

Термін	Визначення, пояснення
UI (Інтерфейс користувача)	Графічне та функціональне оформлення вебзастосунку, з яким взаємодіє користувач
Network Diagram (Мережевий графік)	Графік, котрий показує залежності між роботами проєкту та дозволяє визначити критичний шлях
Float (Резерв часу)	Час, на який можна відкласти виконання роботи без його зміщення кінцевого терміну проєкту
Благодійні суботники	Заходи з благоустрою, прибирання, які організовані з добровільною участю людей та збором пожертвувань
Благодійність	Добровільне надання ресурсів без матеріальної винагороди
Суботник	Заходи громадської праці для прибирання або покращення стану навколишнього середовища, які проведені зазвичай у вихідний день
Захід (подія)	Організована подія (суботник), яка має конкретну дату, мету, місце
Менеджер заходу	Людина, яка має право створювати заходи
Учасник	Будь-яка особа, яка зареєструвалась для суботника
Пожертва	Добровільний внесок у грошовому еквіваленті для підтримки заходів
Платіжний модуль	Функція, за допомогою якої робляться пожертвування через платіжні системи
Календар заходу	Інтерфейс вебзастосунку, котрий відображає всі заплановані та проведені суботники

Продовження таблиці 4.1

Термін	Визначення, пояснення
Картка заходу	Окремий розділ або сторінка з детальною інформацією (місце, мета, час) про конкретний суботник
Реєстрація на захід	Процес запису учасників у конкретному суботнику
Збір коштів	Процес для збору фінансів для проведення суботників
Прозорість пожертвувань	Оприлюднення даних про надходження та витрати фінансів
Звіт про захід	Розділ вебзастосунку, в якому відображено результати про проведений суботник
Відгук учасника	Оцінка чи коментар учасника суботника після завершення заходу
Сповіщення	Автоматичні повідомлення про зміну статусу або оновлення заходу
Ініціатива	Пропозиція чи ідея щодо нового заходу, яку можуть подати користувачі
Архів заходів	Дані звітів про вже проведені заходи
Місце проведення	Локація заходу
Опис заходу	Інформаційний блок, в якому відображено мету конкретного суботника
Ціль збору	Потреба, для якої збираються кошти
Ресурси заходу	Перелік необхідного обладнання для проведення суботника
Пріоритет заходу	Визначення критичності заходу
Рівень активності учасника	Міра залученості учасника
Аналітика участі	Статистики про кількість учасників, обсяги зібраних пожертв

Кінець таблиці 4.1

Термін	Визначення, пояснення
Динаміка пожертвувань	Аналітичні дані, які відображають обсяги та зміни пожертвувань для проміжку часу
Фото та відео звіти	Візуальні матеріали з проведених заходів
Адміністрування	Функції управління, а саме створення заходів, перегляд статистики, процес модерації заходу у вебзастосунку

Такий документ, тобто тезаурус – дозволяє та допомагає в забезпеченні єдиного розуміння між усіма учасниками проєкту, від замовника до розробника.

4.3 Розробка мережевих моделей робіт вебзастосунку

Для початку розробки мережевих моделей для ІТ-проєкта треба визначитися з епіками та розподілити їх на фічі (табл. 4.2)

Таблиця 4.2 – Перелік епіків і фіч, що входять до епіків

Ідентифікатор епіка	Епіки	Фічі
P25E1	Планування та організація заходів	1. Календар заходів 2. Карта заходу 3. Реєстрація користувачів та учасників заходу 4. Пожертвування 5. Платіжний сервіс

Кінець таблиці 4.2

Ідентифікатор епіка	Епіки	Фічі
P25E2	Комунікація та залучення користувачів	1. Сервіс розсилки 2. Пропозиції 3. Відгуки 4. Архів благодійних заходів 5. Підготовка БД системи

Тепер, коли ми визначились з переліком фіч, то знайдемо та відобразимо взаємозв’язки між цими фічами (рис. 4.1).

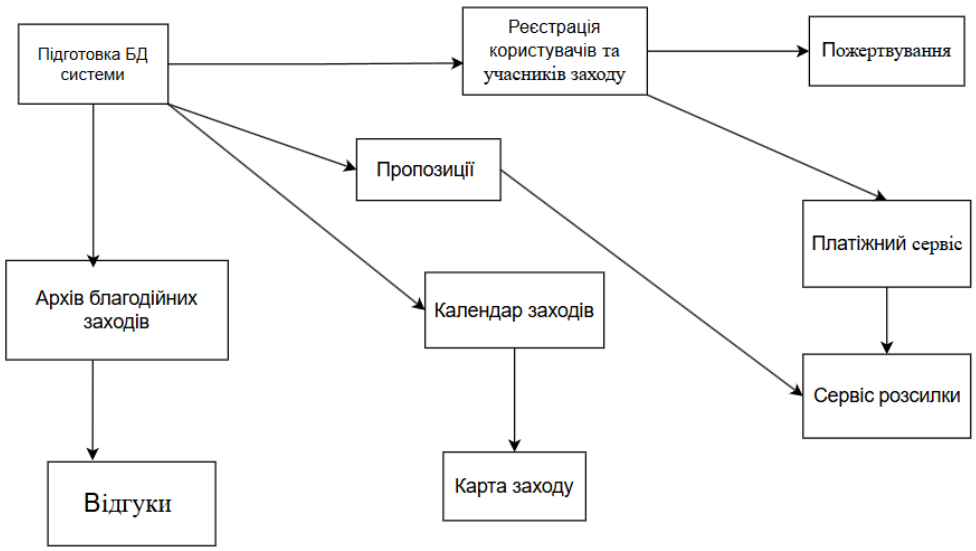

Рисунок 4.1 – Схема взаємозв’язків між фічами

Після того, як встановили взаємозв’язки між фічами, то розрахуємо такі показники: ранній час початку виконання роботи, пізній час початку виконання роботи, повний резерв часу виконання роботи, тривалість виконання робіт.

Ранній час початку виконання роботи (EST) – це найраніший можливий момент часу, з якого може початися виконання роботи. Розраховується з урахуванням завершення всіх попередніх робіт, від яких залежить дана робота. Для першої роботи у проєкті EST=0. Для наступних робіт EST визначається як

максимальний EF (ранній фініш) серед попередніх робіт (6):

$$EST_i = \max_{j \epsilon Pred(i)} (EFT_j), \quad (6)$$

де EFT_j – ранній час початку роботи i;

$Pred(i)$ – множина попередників роботи i;

EFT_j – ранній час завершення роботи j.

Пізній час початку виконання роботи (LST) – це найпізніший можливий момент часу для початку роботи, щоб завершити проєкт у встановлений строк. Для останньої роботи LST розраховується як LFT (пізній час закінчення) мінус тривалість роботи. Для попередніх робіт LST визначається як мінімальний LST наступної роботи (7):

$$LST_i = LFT_i - t_i, \quad (7)$$

Тривалість виконання роботи (t_j) – це час, необхідний для виконання роботи. У даному випадку тривалість визначається у люд./год., що відображає трудовитрати.

Ранній час закінчення (EFT) обраховується за такою формулою:

$$EFT = EST + t, \quad (8)$$

Повний резерв часу виконання роботи (TF) – це час, на який можна відкласти початок роботи без впливу на загальний строк завершення проєкту [18].

Результати розрахунків відобразимо в таблиці 4.3.

Таблиця 4.3 – Основні часові показники для фіч

Фіча	Ранній час початку (EST), люд./год.	Ранній час закінчення (EFT), люд./год.	Пізній час початку (LST), люд./год.	Пізній час закінчення (LF), люд./год.	Тривалість (tj), люд./год.	Повний резерв часу (TF), люд./год.
Підготовка БД	0	151	0	151	151	0
Архів благодійних заходів	151	235	369	453	84	218
Відгуки	235	285	453	503	50	218
Реєстрація користувачів та учасників	151	319	151	319	168	0
Пропозиції	151	285	403	453	50	252
Календар заходів	151	319	234	402	168	83
Карта заходу	319	420	402	503	101	83
Пожертвування	319	403	369	503	84	50
Платіжний сервіс	319	453	319	453	134	0
Сервіс розсилки	453	503	453	503	50	0

Тепер за допомогою цих розрахунків зробимо мережеву модель робіт (рис. 4.2) на рівні фіч, де відобразимо ідентифікатор, ранній час початку (EST), пізній час початку (LST), тривалість роботи (tj), повний резерв часу (TF) у люд./год.

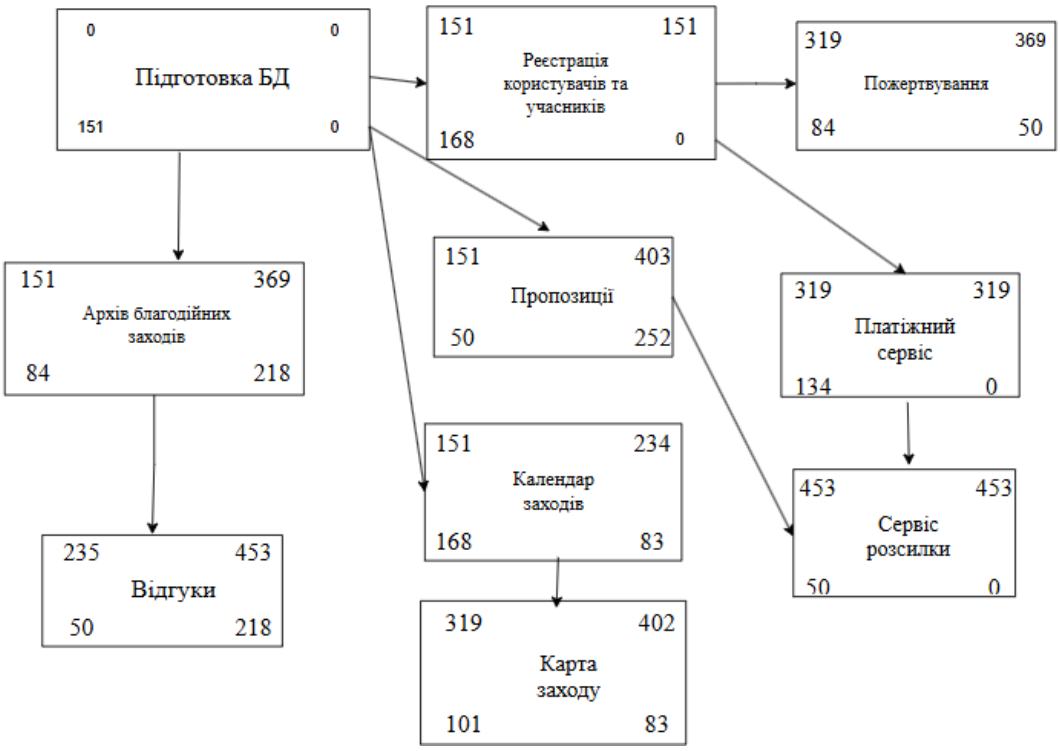

Рисунок 4.2 – Мережева модель робіт проєкту на рівні фіч

Ця мережева модель наочно показує фічі проєкту «Благодійні суботники», які представлені у вигляді прямокутників (вузлів).

Часові параметри кожної фічі, відображені всередині кожного прямокутника:

- лівий верхній сектор — ранній час початку;
- правий верхній сектор — пізній час початку;
- лівий нижній сектор — тривалість роботи;
- правий нижній сектор — повний резерв часу.

Стрілки показують послідовність виконання робіт (логічні та часові залежності), гілки відображають, як роботи об’єднуються у ланцюжки, які формують різні шляхи у проєкті. На цій моделі можна легко ідентифікувати критичний шлях (роботи з TF=0), від якого залежить загальний строк виконання проєкту.

Критичний шлях складається з робіт, у яких повний резерв часу (TF) = 0. Затримка будь-якої з цих робіт призводить до затримки всього проєкту.

Критичний шлях = Підготовка БД (151 люд./год.) → Реєстрація користувачів та учасників (168 люд./год.) → Платіжний сервіс (134 люд./год.)

→ Сервіс розсилки (50 люд./год.) = 151+168+134+50=503 люд./год.

Таким чином, ця мережева модель дозволяє візуалізувати структуру робіт проєкту, аналізувати можливі затримки у виконанні робіт і резерви часу для оптимізації, визначати критичний шлях.

Далі визначимо трудомісткість та пріорітети для фіч (табл. 4.4).

Таблиця 4.4 – Визначення показників фіч проєкту

Порядковий номер	Ідентифікатор фічі	Назва фічі	Трудомісткість (люд./год.)	Пріоритет
1	P25E1_1	Календар заходів	168	1
2	P25E1_2	Карта заходу	101	1
3	P25E1_3	Реєстрація користувачів та учасників заходу	168	1
4	P25E1_4	Пожертвування	84	2
5	P25E1_5	Платіжний сервіс	134	3
6	P25E2_1	Сервіс розсилки	67	2
7	P25E2_2	Пропозиції	50	1
8	P25E2_3	Відгуки	50	3
9	P25E2_4	Архів благодійних заходів	84	2
10	P25E2_5	Підготовка БД системи	151	1

Наступним кроком відбираємо 3 фічі («Календар заходів», «Карта заходу», «Підготовка БД системи») з врахуванням їх пріоритетів, взаємозв'язків та трудомісткостей та щоб забезпечити роботою команду на весь спринт та ще мати невеликий запас). Далі виконуємо декомпозицію цих

фіч для беклогу проєкту та розподілимо на User Story та Task (табл. 4.5).

Таблиця 4.5 – Декомпозиція фіч на User Story та Task

Ідентифікатор фічі	Назва фічі	User Story та Task
P25E1_1	Календар заходів	1) створення форми додавання заходу до плану; 2) створення API з включення заходу до плану; 3) створення фільтру заходів за періодом та типом; 4) створення API для фільтрування даних про заходи з БД; 5) створення таблиці запланованих заходів; 6) створення форми з додавання нового заходу; 7) створення API з додавання нового заходу; 8) створення форми додавання нового заходу на базі пропозиції.
P25E1_2	Карта заходу	1) відображення загальних даних про запланований захід; 2) створення API вибірки основних даних про захід; 3) відображення переліку учасників заходу; 4) створення API відбору учасників заходу; 5) відображення переліку матеріалів для заходу; 6) створення API для відбору матеріалів заходу.
P25E2_4	Підготовка БД системи	1) створення БД; 2) заповнення БД налагоджувальними даними; 3) створення індексів; 4)налаштування прав доступу; 5) налаштування резервного копіювання.

Після цього, зробимо таблицю с характеристиками User Story та Task (табл. 4.6), які знаходяться в беклозі проєкту, і зазначимо ідентифікатор та трудоємкість у Story Points.

Таблиця 4.6 – Характеристики User Story та Task в беклозі проєкту

№	User Story та Task	Ідентифікатор	Трудоємкість, SP
1	Створення форми додавання заходу до плану	ТХ-1	4
2	Створення API з включення заходу до плану	ТХ-2	6
3	Створення фільтру заходів за періодом та типом	ТХ-3	2
4	Створення API для фільтрування даних про заходи з БД	ТХ-4	6
5	Створення таблиці запланованих заходів	ТХ-5	2
6	Створення форми з додавання нового заходу	ТХ-6	3
7	Створення API з додавання нового заходу	ТХ-7	6
8	Створення форми додавання нового заходу на базі пропозиції	ТХ-8	3
9	Відображення загальних даних про запланований захід	ТХ-9	2
10	Створення API вибірки основних даних про захід	ТХ-10	6
11	Відображення переліку учасників заходу	ТХ-11	4
12	Створення API відбору учасників заходу	ТХ-12	6

Кінець таблиці 4.6

№	User Story та Task	Ідентифікатор	Трудоємкість, SP
13	Відображення переліку матеріалів для заходу	ТХ-13	3
14	Створення API для відбору матеріалів заходу	ТХ-14	6
15	Створення БД	ТХ-15	4
16	Заповнення БД налагоджувальними даними	ТХ-16	4
17	Створення індексів	ТХ-17	2
18	Налаштування прав доступу	ТХ-18	1
19	Налаштування резервного копіювання	ТХ-19	3

Для побудування мережевої моделі на рівні декомпозиції User Story та Task для беклогу проєкту спочатку створимо схему взаємозв'язків цих робіт (див. рис. 4.3) для полегшення подальший дій з обчисленнями.

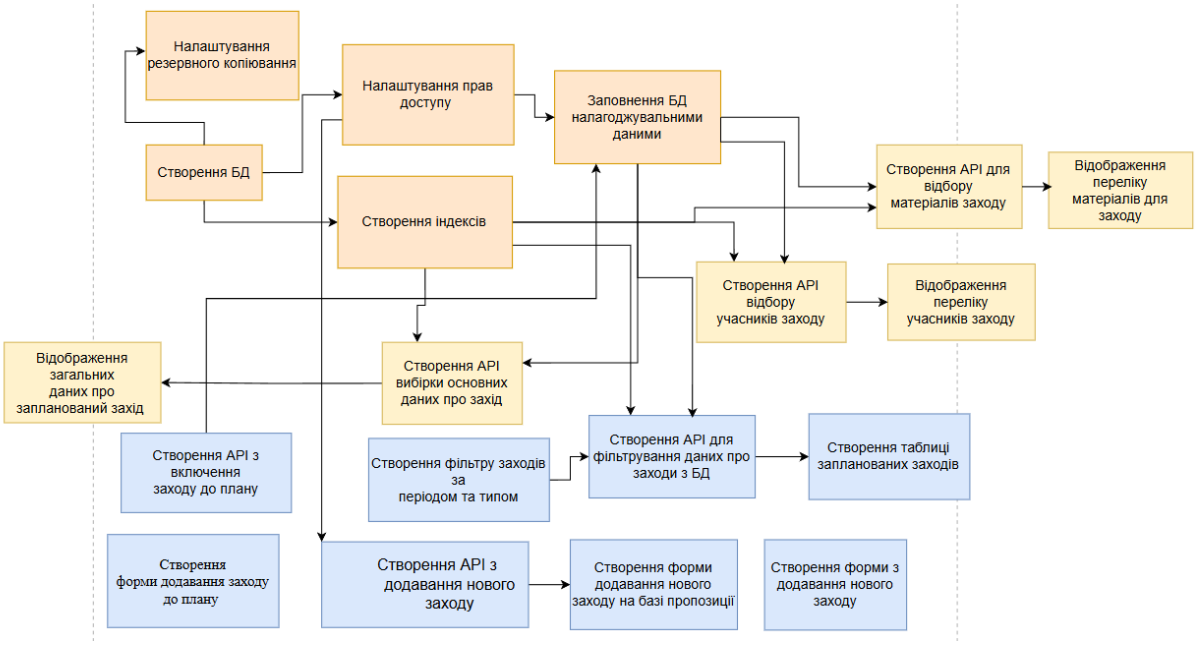

Рисунок 4.3 – Схема взаємозв’язків між User Story та Task, що знаходяться в беклозі проєкту

Наступним кроком створимо мережеву модель у вигляді табличної форми для User Story та Task беклогу проєкту (див. табл. 4.7), де відобразимо результати розразунків.

Для кожної роботи були визначені наступні параметри:

- тривалість виконання роботи (T) — кількість Story Points, що характеризує обсяг роботи;
- ранній час початку виконання роботи (EST) — мінімальний можливий час початку роботи за умови своєчасного завершення попередніх робіт;
- раннє закінчення роботи (EFT) — час закінчення роботи, розрахований як сума раннього часу початку та її тривалості;
- пізній час початку виконання роботи (LST) — максимальний допустимий час початку роботи, який не вплине на загальний термін завершення проекту;
- пізній час закінчення виконання роботи (LFT) — час закінчення роботи, що дозволяє уникнути затримок у проекті;
- повний резерв часу (TF) — кількість часу, на яку робота може бути відкладена без затримки загального терміну завершення проекту;
- вільний резерв часу (FF) — проміжок часу, на який роботу можна відкласти без впливу на час початку наступних робіт;
- незалежний резерв часу (IF) — час, протягом якого можна відкласти роботу, не впливаючи на інші роботи та логіку мережі.

Усі розрахунки проводились у Story Points для більшої зручності та у відповідності сучасним підходам Agile-методології. Загальна тривалість виконання всіх робіт становить 73 Story Points.

Таблиця 4.7– Мережева модель у вигляді табличної форми для рівня декомпозиції User Story та Task беклогу проєкту

№	Назва User Story та Task	Ідентифікатор	Трудоємкість, SP	Індекси попередніх робіт	Індекси наступних робіт	Тривалість виконання роботи (T)	Ранній час початку виконання роботи (EST)	Раннє закінчення роботи (EFT)	Пізній час початку виконання роботи (LST)	Пізній час закінчення виконання роботи (LFT)	Повний резерв часу (TF)	Вільний резерв часу (FF)	Незалежний резерв часу роботи (IF)
1	Створення форми додавання заходу до плану	TX-1	4	-	-	4	0	4	16	20	16	16	16
2	Створення API з включення заходу до плану	TX-2	6	-	TX-16	6	0	6	0	6	0	0	0
3	Створення фільтру заходів за періодом та типом	TX-3	2	-	TX-3	2	0	2	10	12	10	8	8
4	Створення API для фільтрування даних про заходи з БД	TX-4	6	TX-16, TX-17, TX-3	TX-5	6	10	16	12	18	2	0	0

Продовження таблиці 4.7

№	Назва User Story та Task	Ідентифікатор	Трудоємкість, SP	Індекси попередніх робіт	Індекси наступних робіт	Тривалість виконання роботи (T)	Ранній час початку виконання роботи (EST)	Раннє закінчення роботи (EFT)	Пізній час початку виконання роботи (LST)	Пізній час закінчення виконання роботи (LFT)	Повний резерв часу (TF)	Вільний резерв часу (FF)	Незалежний резерв часу роботи (IF)
5	Створення таблиці запланованих заходів	TX-5	2	-	TX-4	2	0	2	18	20	18	8	8
6	Створення форми з додавання нового заходу	TX-6	3	-	-	3	0	3	17	20	17	17	17
7	Створення API з додавання нового заходу	TX-7	6	TX-18	TX-8	6	5	11	11	17	6	0	0
8	Створення форми додавання нового заходу на базі пропозиції	TX-8	3	TX-7	-	3	11	14	16	20	5	5	5

Продовження таблиці 4.7

№	Назва User Story та Task	Ідентифікатор	Трудоємкість, SP	Індекси попередніх робіт	Індекси наступних робіт	Тривалість виконання роботи (T)	Ранній час початку виконання роботи (EST)	Раннє закінчення роботи (EFT)	Пізній час початку виконання роботи (LST)	Пізній час закінчення виконання роботи (LFT)	Повний резерв часу (TF)	Вільний резерв часу (FF)	Незалежний резерв часу роботи (IF)
9	Відображення загальних даних про запланований захід	TX-9	2	TX-10	-	2	16	18	17	19	1	1	0
10	Створення API вибірки основних даних про захід	TX-10	6	TX-16	TX-9	6	10	16	12	18	2	0	0
11	Відображення переліку учасників заходу	TX-11	4	TX-12	-	4	16	20	16	20	0	0	0
12	Створення API відбору учасників заходу	TX-12	6	TX-16, TX-17	TX-11	6	10	16	10	16	0	0	0

Продовження таблиці 4.7

№	Назва User Story та Task	Ідентифікатор	Трудоємкість, SP	Індекси попередніх робіт	Індекси наступних робіт	Тривалість виконання роботи (T)	Ранній час початку виконання роботи (EST)	Раннє закінчення роботи (EFT)	Пізній час початку виконання роботи (LST)	Пізній час закінчення виконання роботи (LFT)	Повний резерв часу (TF)	Вільний резерв часу (FF)	Незалежний резерв часу роботи (IF)
13	Відображення переліку матеріалів для заходу	ТХ-13	3	ТХ-14	-	3	16	19	17	20	1	1	0
14	Створення API для відбору матеріалів заходу	ТХ-14	6	ТХ-16, ТХ-17	ТХ-13	6	10	16	10	16	0	0	0
15	Створення БД	ТХ-15	4	-	ТХ-19, ТХ-18, ТХ-17	4	0	4	1	5	1	0	0
16	Заповнення БД налагоджувальними даними	ТХ-16	4	ТХ-18, ТХ-2	ТХ-14, ТХ-12	4	6	10	6	10	0	0	0
17	Створення індексів	ТХ-17	2	ТХ-15	ТХ-10, ТХ-12, ТХ-14	2	4	6	8	10	4	4	4

Кінець таблиці 4.7

№	Назва User Story та Task	Ідентифікатор	Трудоємкість, SP	Індекси попередніх робіт	Індекси наступних робіт	Тривалість виконання роботи (T)	Ранній час початку виконання роботи (EST)	Раннє закінчення роботи (EFT)	Пізній час початку виконання роботи (LST)	Пізній час закінчення виконання роботи (LFT)	Повний резерв часу (TF)	Вільний резерв часу (FF)	Незалежний резерв часу роботи (IF)
18	Налаштування прав доступу	ТХ-18	1	ТХ-15	ТХ-16, ТХ-7	1	4	5	5	6	1	0	0
19	Налаштування резервного копіювання	ТХ-19	3	ТХ-15	-	3	4	7	16	19	12	12	12

Тепер зазначимо посади команди, кількість посад нормативну трудоємкість для кожної посади в Story Points (див. табл. 4.8).

Таблиця 4.8 – Характеристики команди виконавців

ПІБ	Посада	Story Points
Петров А. А.	Junior	10
Іванов І. В.	Junior	10
Зінченко Д.Є.	Middle	17
Малюк А. І.	Senior	25

Якщо підрахувати нормативні сумарніу трудоміскість для усієї команди то виходить 62 SP.

Загальна кількість Story Points беклогу спринта команди складає 69 SP. Для подальшого відображення мережевої моделі для User Story та Task, що знаходяться в беклозі спринта команди, в формі діаграми Ганта, нам потрібно перерахувати Story Points в людино-години.

Команда (4 розробники) має виконати 62 SP за один 10-денний спринт. Тоді нормативне навантаження команди в людино-годинах на один спринт розрахуємо за формулою (9).

$$\text{Один спринт} = 10 \text{ робочих днів} \times 4 \text{ людини} = 40 \text{ людино-день} = 320 \text{ людино-годин} \quad (9)$$

Тоді 62 SP ≈ 320 год, тому 1 SP = 320 / 62 ≈ 5,16 год.

Таким чином, весь беклог проєкту буде дорівнювати 65 × 5,16 ≈ 335,4 людино-годин.

Далі відбираємо роботи з беклогу проекту в беклог спринта команди таким чином, щоб їх загальна трудомістскіть трохі перевищувала сумарне нормативне навантаження команди (62 Story Points). В таблиці 4.9 зазначимо

перелік відібраних User Story та Task та їх характеристики.

Таблиця 4.9 – Значення User Story та Task до беклогу спринта

User Story та Task	Story Points	Людино-години
Створення БД	4	20,64
Налаштування прав доступу	1	5,16
Заповнення БД налагоджувальними даними	4	20,64
Створення API для відбору матеріалів заходу	6	30,96
Відображення переліку матеріалів для заходу	3	15,48
Створення індексів	2	10,32
Створення API відбору учасників заходу	6	30,96
Відображення переліку учасників заходу	4	20,64
Створення API вибірки основних даних про захід	6	30,96
Створення API з додавання нового заходу	6	30,96
Створення форми додавання нового заходу на базі пропозиції	3	15,48
Створення форми з додавання нового заходу	3	15,48
Створення форми додавання заходу до плану	4	20,64
Створення фільтру заходів за періодом та типом	2	10,32
Створення API для фільтрування даних про заходи з БД	6	30,96
Створення таблиці запланованих заходів	2	10,32
Налаштування резервного копіювання	3	15,48
Створення таблиці запланованих заходів	2	10,32
Створення фільтру заходів за періодом та типом	2	10,32

Наступним кроком призначимо роботам, які знаходяться в беклозі

спринта команди, виконавців і побудуємо мережеву модель для робот спринта команди в формі діаграми Ганта , яка наведено на рис. 4.4.

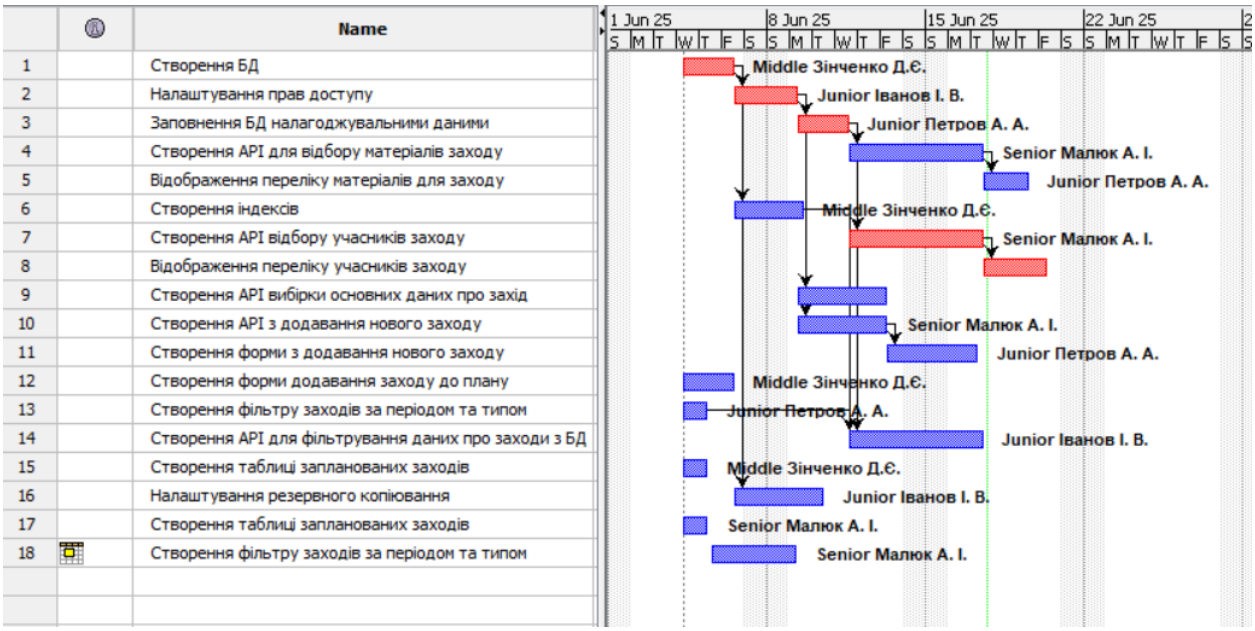

Рисунок 4.4 – Мережева модель беклогу спринта команди у формі діаграми Ганта

Побудована діаграма Ганта [23] для беклогу спринта команди показала нам наступне:

– для того щоб набрати необхідні норми, потрібно застосовувати більше завдань до беклогу спринта команди, тому що деякі роботи можуть виконуватися тількі після виконання попередніх;

– при розподілені задач між участниками команди, деякі учасники виявились неповністю завантажені роботами задач (знову через взаємозв'язки між роботами);

– одна з задач із-за взаємозв’язків та того, що її нема кому назначити залишилась без виконавця роботи, де інші задачі були розподілені між учасниками команди;

– задачі, які не мають прямих взаємозалежностей, можуть виконуватися одночасно різними учасниками команди, а задачі з залежностями – лише після

завершення попередніх, тому краще відбірати такі роботи до беклогу спрінта, що, якщо вони належать до різних фіч, щоб ці фічі не були зовсім пов'язані між собою;

− шлях, який відображений червоним, це той шлях котрий команда не встигає зробити і остання задача буде перенесена на наступний спринт.

4.4 Аналіз отриманих результатів

При аналізі результатів дослідження можна зазначити, що експеримент підтвердив доцільність використання мережевої моделі типу AON для моделювання робіт IT-проєкту.

Була доведена ефективність мережевого моделювання. Завдяки цій моделі ми чітко та легко простежили, які завдання є попередниками, а які – наступниками, тим самим побудували послідовність запуску завдань. Виявлення критичного шляху дало нам змогу сфокусувати увагу на ключових роботах.

Експериментальна перевірка показала, що мережеве моделювання можна застосовувати не тільки на рівні User Story та Task, але і на рівні фіч, де мережева модель відображує чіткі взаємозв’язки. Вона дозволяє визначити приблизні календарні строки виконання проєкту в цілому за допомогою критичного шляху та визначати пріоритети фіч.

Мережеве моделювання дає нам прозорість у баченні структури проєкту та гнучкість при плануванні.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено методи та моделі створення мережевої структури робіт ІТ-проєкту.

У першому розділі кваліфікаційної роботи проведено аналіз особливостей розробки веборієнтованих програмних систем та аналіз методологій і програмних засобів управління проектом з розробки вебзастосунків.

У другому розділі досліджено особливості методів побудови та аналізу мережевої структури робіт проєкту, зроблено обгрунтований вибір типу мережевої моделі та форм її подання для опису зв'язків між функціями одного рівня декомпозиції (роботами з виконання ІТ-проєкту).

У третьому розділі запропонований спосіб використання мережевих моделей для ефективного планування на різних рівнях декомпозиції робіт ІТ-проєкту, коли управління проєктом виконується за методологією Scrum. Також запропонований алгоритм запобігання можливості включення до чергового спринту ланцюжків взаємопов'язаних завдань, критичний шлях яких перевищує тривалість спринту, розроблений тезаурус ІТ-проєкту вебзастосунку.

У четвертому розділі проведено експериментальну перевірку методу мережевого моделювання під час управління ІТ-проєктом вебзастосунку. Результати експериментальної перевірки підтвердили доцільність побудови та використання мережевих моделей для ефективного керування ІТ-проєктом за методологією Scrum.

Новизна отриманих результатів полягає у використанні мережевого моделювання при управлінні ІТ-проєктом за методологією Scrum.

Даний напрямок досліджень має перспективи для подальшого продовження та поширення у бік автоматичного встановлення зв'язків між роботами ІТ-проєкту .

За тематикою кваліфікаційної роботи було опубліковано тези доповіді на тему «Дослідження методів та моделей створення мережевої структури робіт ІТ проєкту» [24].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Чинний від 2017-07-01. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.

2. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016 07 01. – Вид. офіц. Київ : УкрНДНЦ, 2016. 16 с.

3. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проектами в галузі інформаційних технології»/Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2024. – 24 с.

4. Етапи розробки вебзастосунку. URL: https://terentevdesignstudio.com/blog/etapi-rozrobki-veb-dodatkiv/ (дата звернення: 04.02.2025).

5. Архітектура вебзастосунку. URL: https://testmatick.com/uk/glossary/arhitektura-veb-sajtu/ (дата звернення: 07.02.2025).

6. Що таке фронтенд. URL: https://asabix.com.ua/what-is-frontend/ (дата звернення: 10.02.2025).

7. Що таке бекенд. URL: https://webtune.com.ua/statti/web-rozrobka/frontend-i-backend-rozrobka/#id2 (дата звернення: 15.02.2025).

8. Методологія Agile. URL: https://qagroup.com.ua/publications/agile-info/ (дата звернення: 20.02.2025).

9. Що таке методологія Scrum. URL: https://career.softserveinc.com/uk-ua/stories/what-is-scrum-methodology (дата звернення: 22.02.2025).

10. Nexus в масштабованих середовищах Scrum. URL: https://www.thescrummaster.co.uk/uk/docs/what-challenges-does-nexus-

specifically-aim-to-address-in-scaled-scrum-environments/ (дата звернення: 27.02.2025).

11. Методологія LeSS та як вона працює. URL: https://dou.ua/forums/topic/39906/ (дата звернення: 02.03.2025).

12. Що таке SAFe, кому він потрібен і де його застосовують. URL: https://brainrain.com.ua/uk/chto-takoe-safe/ (дата звернення: 06.03.2025).

13. Чому IT-компанії обирають Jira. URL: https://iampm.club/ua/blog/shho-take-jira-i-yak-z-neyu-praczyuvati/ (дата звернення: 10.03.2025).

14. Trello і для чого він потрібен. URL: https://trello.com/uk/tour (дата звернення: 12.03.2025).

15. Azure DevOps Services. URL: https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops (дата звернення: 15.03.2025).

16. Mike Cohn. Agile Estimating and Planning. — Upper Saddle River, NJ: Prentice Hall, 2005. — 368 с.

17. Kenneth S. Rubin. Essential Scrum: A Practical Guide to the Most Popular Agile Process. — Upper Saddle River, NJ: Addison-Wesley, 2012. — 504 с.

18. Gunther Verheyen. Scrum: A Pocket Guide. — Boston: Van Haren Publishing, 2013. — 120 с.

19. Peter Götz, Uwe M. Schirmer, Kurt Bittner. The Professional Scrum Team. — Boston: Addison-Wesley, 2023. — 240 с.

20. Mitsuo Gen, Lin Lin, Runwei Cheng. Network Models and Optimization: Multiobjective Genetic Algorithm Approach. Springer, 2008. — 706 с.

21. Erol Gelenbe, Ibrahim Mitrani. Analysis and Synthesis of Computer Systems. Imperial College Press, 2010. — 324 с.

22. Кушнарьов Д. В. ІС організації з проведення благодійних суботників: пояснювальна записка до кваліфікаційної роботи рівня бакалавра / Харків. нац. ун-т радіоелектроніки; фак. Комп'ютер. наук; каф. ІУС – Харків, 2022. – 81 с.

23. Cristina & Olivier Rebiere. Mastering the Gantt Chart: Understand and Use the "Gantt Project" Open Source Software Efficiently. Independently Published, 2017. — 56 с.

24. Бордюг Д.Є., науковий керівник: Борисенко Т.І. Дослідження методів та моделей створення мережевої структури робіт ІТ-проєкту: матеріали VII Всеукраїнської студентської наукової конференції, м.Київ, 23 травня, 2025 рік / ГО «Молодіжна наукова ліга».— Вінниця: ТОВ«УКРЛОГОС Груп», 2025. с. 455-458.