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 моделей і методів оцінки трудовитрат
при реалізації ІТ-проєктів
(тема)

Виконав:
студент 2 курсу, групи УПГІТм-22-2
Іваненко Ольга Валентинівна
(прізвище, ім'я, по батькові)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Управління проєктами
в галузі інформаційних технологій
(повна назва освітньої програми)

Керівник зав. каф. ІУС Костянтин ПЕТРОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

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

2024 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 проєктами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

« 01 » квітня 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Іваненко Ользі Валентинівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей і методів оцінки трудовитрат при реалізації ІТ-проєктів

затверджена наказом університету від 01 квітня 2024 р. № 258Cm

2. Термін подання студентом роботи до екзаменаційної комісії 05 06 2024 р.

3. Вихідні дані до роботи науково-технічна література, публікації та інтернет-ресурси, що стосуються теми кваліфікаційної роботи; матеріали звіту з науково-дослідної практики

4. Перелік питань, що потрібно опрацювати в роботі огляд існуючих методів оцінки трудовитрат та виявлення їхніх переваг та недоліків; модифікація методу для оцінки трудовитрат в ІТ-проєкті; проведення експериментальної перевірки розробленого методу; аналіз отриманих результатів.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.04.2024	виконано
2	Пошук наукових публікацій, літературних джерел та інформаційних ресурсів у наукових виданнях і Інтернеті	02.04.2024 – 05.04.2024	виконано
3	Аналіз сучасного стану об’єкта дослідження	06.04.2024 – 10.04.2024	виконано
4	Формування проблеми щодо вдосконалення з урахуванням її актуальності	11.04.2024	виконано
5	Огляд існуючих підходів, моделей, методів та технологій управління змінами при реалізації ІТ-проєктів	12.04.2024 – 15.04.2024	виконано
6	Постановка задачі дослідження	16.04.2024	виконано
7	Розробка модифікованого методу оцінки трудовитрат ІТ-проєктів	17.04.2024 – 22.04.2024	виконано
8	Практична апробація розробленого методу	23.04.2024 – 11.05.2024	виконано
9	Аналіз отриманих практичних результатів використання розробленої моделі	12.05.2024 – 15.05.2024	виконано
10	Оформлення пояснювальної записки	15.05.2024 – 20.05.2024	виконано
11	Підготовка презентації	21.05.2024 – 22.05.2024	виконано
12	Надання роботи для перевірки на плагіат	31.05.2024	виконано
13	Надання роботи на підпис науковому керівникові	02.05.2024	виконано
14	Надання роботи на рецензування	03.05.2024	виконано
15	Захист кваліфікаційної роботи в екзаменаційній комісії	07.06.2024	виконано

Дата видачі завдання 01 квітня 2024 р.

Студент _____

(підпис)

Керівник роботи _____ зав. каф. ІУС Костянтин ПЕТРОВ

(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 92 с., 4 розділи, 9 рисунків, 27 таблиці, 19 джерел, 1 додаток.

МЕТОДИ ОЦІНКИ ТРУДОВИТРАТ, МОДИФІКАЦІЯ МЕТОДУ, ПРОЄКТ, ТРУДОВИТРАТИ, ВАРІАНТ ВИКОРИСТАННЯ, PERT, USE CASE POINTS.

Об’єкт дослідження – процес оцінювання трудовитрат при реалізації ІТ-проєктів .

Предмет дослідження – моделі та методи оцінювання трудовитрат при реалізації ІТ-проєктів.

Мета роботи – аналіз методів оцінки трудовитрат при реалізації ІТ-проєктів та розробка власного методу оцінки трудовитрат, з урахуванням переваг і недоліків існуючих підходів.

Дослідження, що проведені в роботі, ґрунтуються на аналізі теоретичного матеріалу, синтезі схожих методів для оцінювання трудовитрат проєкту та на використанні статистичних методів для оцінки точності запропонованого підходу.

В процесі виконання роботи була проведена модифікація методу Use Case Points з використанням методу PERT, запропоновано алгоритм реалізації модифікованого методу, а також наведенно опис експериментальної перевірки даного методу та проведена апробація.

Запропонований метод є корисним інструментом для керівників ІТ-компаній та менеджерів проєктів. Він спрощує процес оцінки трудовитрат, забезпечуючи більш точні та обґрунтовані прогнози на ранніх стадіях розробки ПЗ. Це дозволяє ефективніше планувати розподіл ресурсів, часу та бюджету, що в свою чергу може підвищити успішність проекту.

ABSTRACT

Thesis contains: 92 pages, 4 chapters, 9 figures, 27 tables, 19 sources, 1 appendices.

ESTIMATING LABOR COSTS, LABOR COSTS, METHOD MODIFICATION, METHODS FOR ESTIMATING LABOR COSTS, PERT, PROJECT, USE CASE, USE CASE POINTS.

Object of research is the process of estimating labor costs in the implementation of IT projects.

Subject of research – models and methods for estimating labor costs in the implementation of IT projects.

Purpose – to analyze the methods of estimating labor costs in the implementation of IT projects and to develop our own method of estimating labor costs, considering the advantages and disadvantages of existing approaches.

The research conducted in this paper is grounded in the analysis of theoretical material, the synthesis of similar methods for estimating project labor costs, and the use of statistical methods to assess the accuracy of the proposed approach.

As a result of the work, the Use Case Points method was modified using the PERT method, and an algorithm for developing the modified method was proposed, as well as a description of the experimental verification of this method and its testing.

The proposed method is a convenient tool for project managers. It simplifies the process of estimating labor costs, providing more accurate and reasonable forecasts in the early stages of software development. This allows for more efficient planning of resource, time, and budget allocation, which in turn can increase project success.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ШНМ – штучна нейронна мережа

ECF – environmental complexity factor – фактор складності зовнішнього середовища

COCOMO – constructive cost model – конструктивна модель вартості

COSYSMO – constructive systems engineering cost model – конструктивна модель вартості інженерії систем

IBM – international business machines

FP – functional point – функціональна точка

GA – genetic algorithm – генетичний алгоритм

LOC – lines of code – рядки коду

MBI – manpower buildup index – індекс нарощування кадрів

PI – productivity index – індекс продуктивності

PERT – program evaluation and review technique – техніка оцінки та аналізу програм

SEER – sem – software evaluation and estimation of resources – software estimating model – оцінка програмного забезпечення та оцінка ресурсів – модель оцінки програмного забезпечення

SLIM – software life cycle management – модель життєвого циклу програмного забезпечення

TCF – technical complexity factor – фактор технічної складності

UAW – unadjusted actor weight – нескоригована вага акторів

UCP – use case points – метод бальної оцінки варіантів використання

UML – unified modeling language – уніфікована мова моделювання

UUCP – unadjusted use case points – некориговані точки варіантів використання

UUCW – unadjusted use case weights – нескоригована вага варіантів використання

ВСТУП

У сучасному світі, де інформаційні технології відіграють ключову роль в багатьох сферах життя, ефективне планування та управління ІТ-проектами стає все більш важливим. Одним з основних аспектів планування ІТ-проектів є оцінка трудомісткості розробки програмного забезпечення. Точна оцінка трудомісткості на ранньому етапі життєвого циклу проєкту може значно вплинути на успіх або провал проекту, оскільки вона впливає на розподіл ресурсів, планування часу та бюджетування.

На сьогоднішній день існує багато методів оцінки трудомісткості розробки програмного забезпечення, такі як алгоритмічні методи, експертні, методи, що орієнтовані на навчання та ін. Однак, багато з цих методів мають обмеження або не надають достатньої точності в оцінці трудомісткості, особливо на ранніх стадіях розробки ПЗ. Важливо розуміти, що від даних методів не слід очікувати абсолютної точності. Замість спроби точно визначити реальний обсяг проекту, більш продуктивним рішенням є здійснення модифікацій існуючих методів оцінки з метою підвищення їх точності.

Одним з методів, який викликає особливий інтерес, є метод Use Case Points (UCP), заснований на використанні варіантів використання (use cases). Цей підхід є алгоритмічним, що робить його достатньо надійним та передбачуваним. UCP дозволяє врахувати функціональні вимоги до системи, що робить його особливо корисним на ранніх стадіях проекту. Крім того, UCP не прив'язаний до конкретних технологій або мов програмування, що свідчить про його універсальність і гнучкість у використанні. Тому, його було обрано в якості методу, який буде детально вивчений та модифікований для підвищення його точності.

Актуальність даної кваліфікаційної роботи обумовлена необхідністю покращення існуючих методів оцінки трудомісткості розробки програмного забезпечення. Зокрема, в центрі уваги – метод Use Case Points, який,

незважаючи на свою надійність та універсальність, має деякі обмеження, які можуть впливати на точність оцінки. Тому, виникає потреба в модифікації цього методу з метою покращення його точності та надійності.

Саме тому мету дослідження можна сформулювати як вдосконалення обраного методу оцінки трудовитрат.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Огляд та загальний аналіз проблеми оцінки трудовитрат при виконанні IT-проєктів

Управління проєктами, особливо у сфері інформаційних технологій (IT), є дуже важливим напрямом досліджень в Україні. IT-проєкти займають одне з лідируючих місць за кількістю серед усіх проєктів у країні.

Дослідницька консалтингова компанія StandishGroup, у своєму аналізі, класифікує всі проєкти на три категорії: успішні проєкти, провальні проєкти та спірні – проєкти, які були завершені з відхиленням від запланованих строків, з перевищенням бюджету або досягли лише часткової реалізації поставлених цілей.

За результатами дослідження StandishGroup, тільки 36% програмних проєктів можна вважати успішними, тоді як решта 64% – це спірні (48%) і провальні (16%) проєкти. Це статистика, що не змінювалась протягом багатьох років, не дивлячись на посилення розвитку галузі.

Згідно з дослідженням від PricewaterhouseCoopers, упродовж останніх 10 років у топ-3 причин невдач проєктів завжди була вказана така причина, як неправильні оцінки проєкту. Крім того, найбільш вагомими причинами, що призводять до проблем, ризиків та, в кінці кінців, невдач у IT-проєктах в Україні можна виділити наступні:

– високий ступінь невизначеності;

– постійні зміни протягом реалізації проєкту;

– складності узгодження очікувань і сприйняття різних зацікавлених осіб;

– недостатній рівень знань і досвіду проєктного менеджера (проєктної команди);

– відсутність належного системного підходу до управління проєктом або пряме нехтування принципами й методологіями управління проєктами, що призводить до помилок при зборі та аналізі даних, плануванні, управлінні ризиками, організації комунікацій у проєкті, веденні проєктної документації та прийнятті управлінських рішень.

Успішність проєкту значною мірою залежить від якісного планування та управління, що включає точну оцінку необхідних ресурсів, включаючи час, трудовитрати та інші ресурси. Неправильні оцінки на ранніх етапах проєктування можуть призвести до серйозних фінансових втрат і навіть банкрутства IT-компаній.

Важливим аспектом оцінки проєкту є визначення трудовитрат, яке відображає обсяг та складність завдань. Аналіз показує, що оцінка трудовитрат при розробці програмного забезпечення є складним процесом, який залежить від різних чинників. Перш за все, характеристики проєкту, такі як обсяг і складність, впливають на точність оцінки. Важливою є зміна вимог, технологій та платформи. Досвід і навички команди також відіграють суттєву роль. Оцінка може бути покращена за допомогою різних методів оцінювання та уважного вивчення процесу. Проте існують проблеми, такі як недооцінка змін у процесі розробки. Це свідчить про потребу в більш глибокому та об'єктивному підході до оцінки трудовитрат, що робить дану проблему актуальною. [1] – [2].

1.2 Аналіз існуючих моделей та методів оцінки трудовитрат в IT-проєктах

На сьогоднішній день існує багато моделей та методів оцінки трудовитрат при створенні програмного забезпечення, які можна умовно розділити на алгоритмічні, неалгоритмічні та методи, що спрямовані на навчання.

До алгоритмічних методів (або параметричних) моделей та методів включають модель COCOMO, метод функціональних точок, SLIM та інші. До неалгоритмічних (також відомих, як непараметричні) відносяться: метод експертного оцінювання, метод оцінки за аналогією, Price-to-win, методи Bottom-Up, Top-Down, Planning Pocker та інші). Методи, що орієнтовані на навчання, або методи машинного навчання, базуються на використанні штучних нейроннх мереж (ШНМ), нечіткої логіки (Fuzzy Logic), Байєсівських мереж (Bayesian Network) та генетичних алгоритмах (GA).

Розглянемо ці методи більш детально.

1.2.1 Алгоритмічні моделі та методи

Ці методи використовують математичні моделі для оцінки трудовитрат та вартості програмного забезпечення. Ці математичні моделі ґрунтуються на дослідженнях історичних даних і використовують деякі вхідні дані, наприклад, кількість рядків коду, кількість функцій для виконання, а також деякі фактори витрат, такі як мова програмування, методологія дизайну, рівень кваліфікації, оцінка ризику тощо.

Роздивимось модель Патнема (SLIM). Модель життєвого циклу програмного забезпечення (Software LIfe Cycle Management), що розроблена Ларрі Путнамом, є однією з перших алгоритмічних моделей оцінки трудовитрат та вартості програмного забезпечення. Вона заснована на функції Нордена/Рейлі і відома як модель макрооцінки, призначена переважно для великих проєктів. SLIM може записувати та аналізувати дані з раніше завершених проєктів, які потім використовуються для калібрування. Після цього можна відповісти на низку питань, щоб отримати значення нарощування трудовитрат у існуючій базі даних [3]. SLIM оцінює розмір програмного забезпечення за кількістю рядків коду (LOC).

Головні показники управління в SLIM – це Індекс Продуктивності (PI) та Індекс Зростання Кадрів (MBI). PI відображає загальний потенціал розвитку організації та продуктивність процесу розробки ПО. Високе значення PI свідчить про високу продуктивність та низьку складність процесу. MBI відображає швидкість збільшення персоналу та вплив на нього різних факторів.

SLIM використовує ідею розподілу Релея для моделювання зміни використання ресурсів у проєкті з часом. Дослідження показали, що цей підхід добре апроксимує криву зусиль для різних процесів розробки апаратного забезпечення. Крива Нордена/Рейлі відображає кількість персоналу в залежності від часу, а для впливу на її форму використовуються коефіцієнти MBI та PI. Крива Рейлі представлена на рис. 1.1 [4].

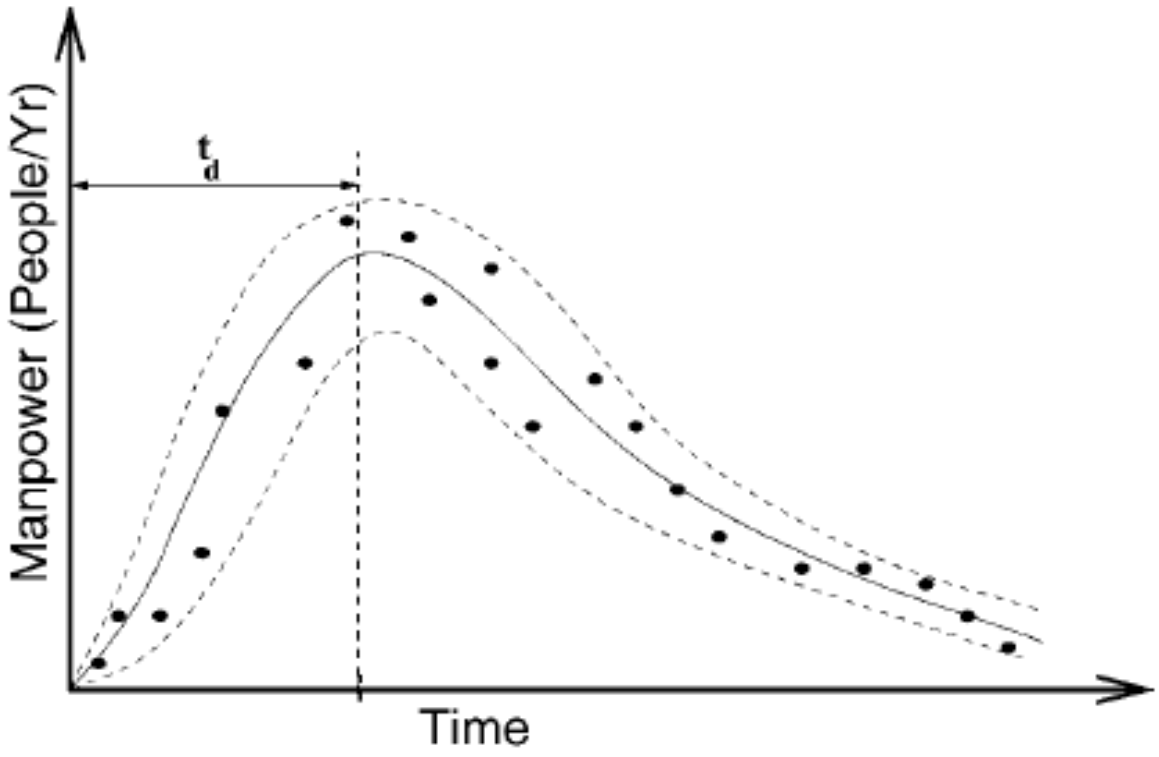

Рисунок 1.1 – Крива Рейлі

Переваги та недоліки методу представлені у табл. 1.1.

Таблиця 1.1 – Переваги та недоліки методу SLIM

Переваги	Недоліки
Використання лінійного програмування для врахування обмежень розробки як за витратами, так і за трудовими ресурсами	Оцінки моделі SLIM дуже чутливі до фактору технологій, що може призвести до неточності оцінки трудовитрат
Модель SLIM потребує менше параметрів для генерації оцінки трудовитрат, ніж COCOMO'81 і COCOMO'II	Модель SLIM не підходить для малих проєктів, оскільки вона передбачає використання лінійного програмування та інші складні методи оцінки, які можуть бути зайвими для менших обсягів робіт

Роздивимось модель COCOMO. COCOMO (Constructive Cost Model) є математичною моделлю для оцінки вартості, трудовитрат та термінів виконання проєктів розробки ПЗ. Існують дві версії: COCOMO 81 та COCOMO II.

COCOMO 81 пропонує три рівні оцінки: базовий, проміжний та деталізований, що залежать від доступної інформації про проєкт. При роботі з моделлю COCOMO важливо враховувати концепцію параметрів вартості, які впливають на оцінку трудовитрат розробки програмного забезпечення. Це величина, яка оцінює різні часові, якісні та ресурсні аспекти розробки ПЗ. Кожен з параметрів може бути відкалібрований.

COCOMO II – це покращена версія моделі оцінки вартості розробки програмного забезпечення COCOMO 81, створена для вирішення масштабування, функціональності та передбачення нових вимог та технологій, що виникли після випуску COCOMO 81.

COCOMO II має три субмоделі: композиції, раннього проєктування та пост-архітектури. Модель використовує розрахунки, подібні до COCOMO 81, але враховує деякі нові фактори та параметри, наприклад, як основна метрика

використовуються рядки коду, на відміну від точок функцій, які використовувалися як основна метрика в COCOMO 81. Крім того, COCOMO II використовує 17 параметрів витрат, порівняно з 15 у COCOMO [5].

Переваги та недоліки методу продемонстровано у табл. 1.2.

Таблиця 1.2 – Переваги та недоліки методу COCOMO II

Переваги	Недоліки
COCOMO надає простий та зрозумілий метод для оцінки трудовитрат	Для отримання надійних оцінок необхідно мати точні дані щодо розміру та складності проєкту, що може бути важко забезпечити
Модель може бути застосована для оцінки різних типів проєктів та різних етапів їх життєвого циклу	Може не враховувати прихованих витрат, що призводить до подорожчання оцінки. Необхідно калібрувати параметри

Роздивимось метод функціональних точок. Метод функціональних точок (FP), розроблений Алланом Альбрехтом в IBM, є методом оцінки розміру і складності програмного забезпечення, аналізуючи його функціональні можливості, надаються користувачам.

Цей метод оцінює програмне забезпечення, шляхом розподілу його функціональності на п'ять типів: входи, виходи, запити, внутрішні та зовнішні інтерфейси. Кожну з цих функцій класифікують за складністю та присвоюють їм певну кількість функціональних балів на основі їх складності.

Функціональний точковий аналіз приділяє особливу увагу вимірюванню розміру функціональності в процесі розробки програмного забезпечення та дозволяє прийняти універсальний підхід до вимірювання розміру, оскільки він не обмежується конкретними технологіями чи мовами програмування.

Метод аналізу функціональних точок включає в себе три основних етапи: визначення необхідних форм операцій, оцінку та аналіз компонентів

системи програмування, та оцінку загальних характеристик системи. Ці загальні характеристики можуть включати такі аспекти, як:

– обробка даних;

– продуктивність;

– конфігурацію обладнання;

– швидкість транзакцій;

– введення даних;

– ефективність роботи конечного користувача;

– простоту використання і можливість полегшення внесення змін [5] – [6].

Переваги та недоліки методу представлені у табл. 1.3.

Таблиця 1.3– Переваги та недоліки методу функціональних точок

Переваги	Недоліки
Незалежність від використовуваних мов та інструментів	Атрибути якості, час розробки та людські ресурси не враховуються
Легко оцінити вартість розробки на етапі збору вимог/початковому етапі	Не здатний працювати з гібридними системами
Незалежність від вимог до до програмного забезпечення	Довготривалий

Роздивимось метод COSYSMO. Метод COSYSMO (Constructive Systems Engineering Cost Model) є параметричною моделлю оцінки вартості програмного забезпечення, який оцінює трудовитрати та час, необхідні для виконання завдань системної інженерії. Модель містить 14 мультиплікаторів трудовитрат та 4 коефіцієнти розміру, загалом 18 параметрів.

Для визначення розміру використовуються різні метрики, включаючи функціональні точки та варіанти використання. COSYSMO пройшла три основні ітерації. Кожна наступна ітерація вдосконалювала і розширювала попередню, включаючи перегляд факторів вартості та внесення змін до

параметрів моделі. Оціночне співвідношення COSYSMO використовує регресійне рівняння для прогнозування трудовитрат в людино-місяцях. Остання модифікація включає два нових фактори витрат: розподіл ризиків та можливостей, і стиснення розкладу, що розширює загальну кількість факторів витрат до 16 [7].

Переваги та недоліки методу представлені у табл. 1.4.

Таблиця 1.4 – Переваги та недоліки методу COSYSMO

Переваги	Недоліки
COSYSMO може бути адаптована до різних типів проєктів шляхом коригування параметрів	Точність моделі залежить від наявності достатньо надійних проєктних даних.
Метод враховує різні аспекти розробки систем, включаючи складність, яка часто є ключовою складовою трудовитрат	Модель вимагає розуміння та оцінки численних факторів, що може бути складним завданням
COSYSMO пропонує прозорий процес, дозволяючи зрозуміти, яким чином визначаються оцінки	Вона перетинається з моделлю COCOMO II. що призводить до непотрібного подвійного підрахунку трудовитрат

Роздивимось модель SEER-SEM. SEER-SEM (Software Evaluation and Estimation of Resources – Software Estimating Model) є моделлю оцінювання програмного проєкту, яка використовує підхід на основі параметрів і заснована на моделі Дженсена (вперше опублікована в 1983 році). SEER-SEM використовує параметри розміру, персоналу, складності, середовища та обмежень для оцінки вартості, ризиків та графіку проєкту. Він охоплює все – від фази планування проєкту до його реалізації, і працює з декількома конфігураціями середовища і додатків.

SEER-SEM складається з групи моделей, які працюють разом, щоб

надати оцінку трудовитрат, тривалості, дефектів та персоналу. Він використовує бази знань, параметричні алгоритми, можливості моделювання та історичні прецеденти [5], [7].

Переваги та недоліки методу представлені у табл. 1.5.

Таблиця 1.5 – Переваги та недоліки методу SEER-SEM

Переваги	Недоліки
Є дуже потужною моделлю, яка містить різноманітні інструменти для виконання різних оцінок, не тільки трудовитрат	Велика кількість вхідних параметрів пов'язаних з різними факторами проєкту, що збільшує складність SEER-SEM
SEER-SEM надає користувачам гнучкість у виборі підходящих баз знань для настройки платформ, програм, методів розробки та стандартів розробки згідно з вимогами проєктів	Специфічні деталі проєкту можуть ускладнювати модель, особливо при ідентифікації нелінійних зв'язків між параметрами входу і виходу.

Роздивимось метод Use Case Points. Метод Use Case Points (UCPs) була створена Густавом Карнером у 1993 році і заснована на порівняльних стандартах, подібних до процедури оцінки Functional Points (FP). Вона була призначена для специфічних впорядкованих систем і вимог до систем, які були записані за допомогою варіантів використання (use cases), які є частиною систем Unified Modeling Language (UML).

На основі компонентів варіантів використання системи визначається UCP, який використовується для кількісного визначення розміру програмного забезпечення, яке потім використовується для оцінки трудовитрат проєкту.

Метод Use Case Points (UCP) використовує три основних параметри для оцінки обсягу роботи:

– нескориговані бали варіантів використання, які виражають кількість і

складність варіантів використання системи;

– фактор технічної складності, який враховує вплив технічних аспектів (обмеження апаратного обладнання, вимоги до безпеки, та ін);

– фактор складності середовища.

Можна також включити коефіцієнт продуктивності, який допомагає врахувати ефективність роботи команди для настроювання оцінки під конкретний проєкт [8].

Переваги та недоліки методу продемонстровано у табл. 1.6.

Таблиця 1.6 – Переваги та недоліки методу Use Case Points

Переваги	Недоліки
Може бути виміряний на ранній стадії проєкту життєвого циклу проєкту	
Коли оцінка виконується кваліфікованими людьми кваліфікованими людьми, оцінки будуть близькі до фактичних даних	Застосовується лише для тих програмних проєктів, специфікація яких специфікація може бути виражена через варіанти використання
Простий у використанні	

1.2.2 Неалгоритмічні методи

Неалгоритмічні методи використовують деяку інформацію про попередні проєкти, які схожі на поточний, і зазвичай процес оцінки вартості в цих методах здійснюється відповідно до аналізу попередніх наборів даних.

У більшості випадків, неалгоритмічні методи оцінки базуються на консенсусі, тобто на спроможності одного або групи людей (експертів)

працювати разом для оцінки трудовитрат, необхідних для реалізації проєкту.

Метод експертного оцінювання є одним з традиційних методів, які використовуються для оцінки трудовитрат на ранніх етапах його розробки. Він передбачає консультації з експертом або групою експертів, які мають значний досвід роботи з аналогічними проєктами. Ці спеціалісти можуть надати цінну інформацію про потреби проєкту, включаючи конкретні задачі та ресурси, які необхідні для його успішного виконання. Такий підхід дозволяє враховувати практичний досвід і експертні знання для більш точної оцінки трудовитрат.

Експертна оцінка може бути надзвичайно корисною, особливо в умовах, коли існують обмеження, що перешкоджають ефективному та дієвому збору даних. При цьому експертна оцінка є необхідною для прийняття рішень з урахуванням обмежень та жорстких факторів, що впливають на реалізацію проєкту. Вона забезпечує можливість адаптувати оцінку трудовитрат до конкретних умов і потреб проєкту, що є важливим для його успішного виконання. [5].

Роздивимось техніку Delphi. Delphi – відомий експертний метод, розроблений наприкінці 1940-х років як метод передбачення майбутніх подій. Він передбачає реалізацію наступних етапів оцінювання.

Процес оцінки вартості проєкту починається з того, що керівник проєкту надає експертам специфікацію та форму оцінки, які вони заповнюють анонімно. Потім проводиться групова зустріч для обговорення питань оцінки. Керівник проєкту готує резюме оцінки вартості та розсилає його експертам для повторного заповнення форм. Ці кроки повторюються до досягнення консенсусу щодо оцінки вартості проєкту [7].

Переваги та недоліки методу продемонстровано у табл. 1.7.

Таблиця 1.7 – Переваги та недоліки методу Delphi

Переваги	Недоліки
Експерти відомі своєю експертизою та досвідом у конкретній області, що дозволяє надавати кваліфіковані та обґрунтовані оцінки	Оцінки можуть бути суб'єктивними та залежати від особистих думок та емоцій експертів, що може призвести до неоднозначних результатів
Процес експертної оцінки може бути швидким та зручним на ранніх етапах проєкту.	Оцінки можуть бути обмеженими областю компетенції експертів, що може призвести до пропуску важливих аспектів проєкту
Не потрібно витрачати час на складний аналіз або збір історичних даних, оскільки експерти можуть швидко надати свої оцінки	Різні експерти можуть давати різні оцінки на одну і ту ж проблему або питання, що може призвести до неоднозначності та недостатньої точності

Роздивимось метод Planning Poker. Це метод на основі експертних оцінок, який вперше був визначений Джеймсом Греннінгом у 2002 році. Ця методика створює оцінку трудовитрат шляхом об'єднання думок декількох експертів. Учасниками планувального покеру є всі розробники команди, тобто програмісти, аналітики, тестувальники, власник продукту і ін., враховуючи, що власник продукту лише бере участь у процесі, але не робить оцінку.

На початку сесії планування, команда представляє користувацькі історії. Кожен учасник отримує колоду карт з оцінками та обирає картку, що відповідає його оцінці. Після цього всі одночасно відкривають свої картки, і якщо оцінки відрізняються, вони обговорюються. Цей процес повторюється до досягнення консенсусу. У випадку відсутності консенсусу історія може бути відкладена для подальшого оцінювання. [9].

Переваги та недоліки методу представлені у табл. 1.8.

Таблиця 1.8 – Переваги та недоліки методу Planning Poker

Переваги	Недоліки
Приємний спосіб оцінки в ігровій формі	Вимагає багато часу
Обговорення призводить до кращих оцінок	Залежить від експертності команди
Учасники не впливають один на одного під час оцінки завдань через конфіденційні індивідуальні оцінки, що допомагає уникнути впливу сторонніх чинників	Менш точні результати, якщо команда не має попереднього досвіду з подібними завданнями

Розглянемо метод «Top to Bottom». Дизайн методу «зверху вниз» був запропонований дослідниками IBM Харланом Міллсом та Ніклаусом Віртом у 1970-х роках і з того часу активно використовується в області оцінки вартості IT-проєктів.

Цей підхід передбачає, що спочатку визначається загальна вартість проєкту на основі глобальних характеристик програмного продукту. Це може бути зроблено за допомогою алгоритмічних або неалгоритмічних методів. Після цього проєкт розбивається на окремі компоненти та підкомпоненти, що дозволяє детальніше оцінити кожну частину проєкту. Цей процес також відомий як декомпозиція.

Як тільки загальна вартість проєкту оцінена, оцінка пропорційно розподіляється між різними компонентами. Для точної оцінки вартості проєкту метод оцінки «зверху вниз» потребує історії та знань про цінову політику проєкту.

Оцінка «зверху вниз» використовується для прийняття стратегічних рішень на високому рівні, особливо коли період планування досить тривалий. Цей метод є особливо корисним, коли доступна лише обмежена інформація про проєкт, і потрібно отримати приблизну оцінку. Однак, варто пам'ятати, що основним недоліком цього методу є його відносна неточність. [5] – [6].

Переваги та недоліки методу представлені у табл. 1.9.

Таблиця 1.9 – Переваги та недоліки методу Top to Bottom

Переваги	Недоліки
Вимагає мінімальну кількість деталей проєкту	Не виявляє складних низькорівневих технічних проблем, які можуть вплинути на збільшення трудовитрат
Легкий у використанні	Надає невелику кількість деталей щодо трудовитрат проєкту
	Є неточним

Розглянемо метод Bottom to Top. Методика оцінки знизу вгору (Bottom-Up) передбачає початок процесу оцінки трудовитрат з найменших компонентів програмної системи і наступне об'єднання оцінок для отримання загальних трудовитрат проєкту. Спочатку кожен компонент оцінюється окремо людьми, які будуть відповідальними за його розробку. Потім ці індивідуальні оцінки сумуються та узагальнюються до вищого рівня для визначення загальної кількості трудовитрат для всього програмного продукту.

Для застосування цього методу необхідне наявність початкового проєктного дизайну, який чітко визначає розбиття компонентів системи, і робота зазвичай представлена у вигляді структури розбиття робіт. Такий підхід дозволяє розглядати трудовитрати з більш деталізованої точки зору, що зазвичай робить оцінки більш точними порівняно з іншими методами [9].

Основна мета методу знизу вгору – зібрати оцінки трудовитрат всіх дрібних компонентів програмного забезпечення для отримання загального кошторису проєкту. Цей підхід направлений на конструювання оцінки системи на основі накопичених знань про малі компоненти програмного забезпечення та їх взаємодію [5].

Переваги та недоліки методу продемонстровано у табл. 1.10.

Таблиця 1.10 – Переваги та недоліки методу Bottom to Top

Переваги	Недоліки
Детальна оцінка кожного виду діяльності для точного визначення загальних витрат проєкту	Оцінка може зайняти багато часу, а отже, вона може бути дорогою
Точність та прозорість (потенційні помилки можуть бути досліджені, а їх вплив може бути протестований через деталізовані дані)	Якщо сумується кожен вид діяльності, може виникнути нестача координації (у зв'язку з перекриттям ресурсів або залежністю від послідовності виконання різних видів діяльності щодо трудовитрат)
Метод дозволяє систематично відслідковувати прогрес проєкту	

Роздивимось метод оцінки співвідношення ціни та виграшу (Price-to-win). Cost-to-win (інша можлива назва) – це метод, за яким витрати на програмування оцінюються як найкраща вартість для перемоги у проєкті. Бюджет замовника більше спрямований на функціональність програмного забезпечення, а проєкт коштує стільки, скільки замовник на нього витратить.

Такий підхід не рекомендується, оскільки замість того, щоб зосередитися на функціональності програмного забезпечення, він більше орієнтований на бюджет та можливості замовника. Точність значно варіюється залежно від бюджету клієнта, тому він оцінюється як малоточний. Це не найкраща практика, оскільки вона може призвести до затримок у розробці та поставці, а також змусити команду розробників працювати понаднормовий час [6].

Переваги та недоліки методу можна знайти у табл. 1.11.

Таблиця 1.11 – Переваги та недоліки методу Price-to-win

Переваги	Недоліки
При угоді про співпрацю ви отримаєте контракт	Може призвести до перевищення робочого часу (працювання розробників понаднормовий час, виснаження та зниження продуктивності)
	Через обмеження бюджету розробляється система низької якості
	Малоточний

Розглянемо метод оцінки за аналогіями. Ця оцінка включає порівняння проєкту з раніше завершеними схожими проєктами. Фактичні дані з завершених проєктів екстраполюються для оцінки трудовитрат запропонованого проєкту. Метод аналогії може бути використаний як на рівні системи, так і на рівні компонентів. Цей метод передбачає такі етапи оцінки.

Спочатку визначаються необхідні характеристики запропонованого проєкту. Потім з історичної бази даних вибираються найбільш схожі завершені проєкти. На основі цих аналогічних проєктів встановлюється оцінка для нового проєкту [8].

Переваги та недоліки методу продемонстровані у табл. 1.12.

Таблиця 1.12 – Переваги та недоліки методу оцінки за аналогіями

Переваги	Недоліки
Може бути точним	Може бути ненадійним через обмежену кількість схожих проєктів
Простий, якщо організація повторює схожу роботу	Існує складність у коректній оцінці відмінностей між проєктами (різні технічні аспекти, різні умови та обмеження)
Оцінки негайно доступні	

1.2.3 Методи, що орієнтовані на навчання

Метод аналізу даних, заснований на припущенні, що машини можуть навчатися із даних та передбачати майбутні результати, називається машинним навчанням. Використання моделей машинного навчання для трудовитрат стає популярним через їхню здатність передбачати точні результати на основі попередніх проєктів. Різні моделі машинного навчання мають свої сильні та слабкі сторони, але загалом їх застосування зазвичай призводить до більш точних прогнозів, ніж це можливо з класичними моделями оцінки.

Розглянемо методи, що базуються на використанні штучних нейронних мереж. Штучні Нейронні мережі (ШНМ) є одним із основних підходів у галузі моделей машинного навчання, натхненних функціонуванням нейронної системи мозку. Вони складаються з вхідного, прихованого та вихідного шарів, де прихований шар складається з блоків, званих нейронами, які надають ваги вхідним даним.

Штучні нейронні мережі можуть використовуватися для різноманітних задач, включаючи класифікацію, кластеризацію та прогнозування даних, що робить їх популярним інструментом для оцінки трудовитрат. Їх особливістю є здатність виявляти нелінійні зв'язки та формувати власні ознаки на основі вхідних даних. Однак, варто враховувати, що ШНМ можуть бути схильні до перевчання, а також вони можуть вимагати значних обчислювальних ресурсів.

Незважаючи на свої обмеження, різні типи ШНМ, такі як нейронна мережа прямої дії, рекурентні нейронні мережі та радіально-базисні нейронні мережі, успішно застосовуються в оцінці вартості програмного забезпечення. Вони мають унікальні характеристики, які можуть бути корисними в різних контекстах оцінки [5].

Переваги та недоліки методу продемонстровано у табл. 1.13.

Таблиця 1.13 – Переваги та недоліки штучних нейронних мереж

Переваги	Недоліки
Дуже добре відбивають нелінійні зв'язки	Вимагають великих обчислювальних потужностей та їх складно програмувати
Можуть автоматично адаптувати свої параметри навчання, ґрунтуючись на наявних даних, що дозволяє їм покращувати свою продуктивність та точність із часом	Аналітик не може вплинути на мережу після завершення етапу навчання. Є «чорним ящиком», тому важко зрозуміти, що відбувається всередині нейронної мережі
Можуть добре вирішувати складні проблеми	Схильні до перенавчання

Розглянемо методи, що базуються на використанні генетичних алгоритмів. Генетичні алгоритми є одним з еволюційних методів оцінки трудовитрат. Еволюційний процес обчислень характеризується тим, що рішення досягається шляхом перебору поколінь циклів рішень-кандидатів, які обмежуються принципом «виживання найбільш пристосованих».

Коли генетичний алгоритм використовується для вирішення реальних проблем, генерується популяція, що складається з випадкового набору індивідів. Популяцію вивчають в процесі еволюції, присвоюючи кожній особині рейтинг, який показує ступінь її пристосованості до середовища. Найкращі особини зберігаються, а менш пристосовані виключаються. Особини, залишені в процесі відбору, можуть зазнавати змін у своїх основних характеристиках через механізм розмноження. Цей механізм застосовується до поточної популяції з метою дослідження простору пошуку і знаходження кращих рішень проблеми за допомогою операторів кросинговеру і мутації, що генерують нові особини для наступного покоління. Цей процес, який називається розмноженням, повторюється до тих пір, поки не буде знайдено задовільного рішення.

На основі досвіду, отриманого з аналізу відомого набору даних Desharnais, генетичне програмування можна рекомендувати до використання для оцінювання трудовитрат. Аналіз показав, що генетичне програмування може пропонувати важливі поліпшення в точності оцінювання і має потенціал стати ефективним доповненням до інструментів для оцінки трудовитрат, що необхідні для розробки програмного забезпечення [8].

Переваги та недоліки методу продемонстровано у табл. 1.14.

Таблиця 1.14 – Переваги та недоліки генетичних алгоритмів

Переваги	Недоліки
Легкі для розуміння концепції	Складний в реалізації та вимагає великих обчислювальних потужностей
Не залежить від специфічних знань про проблеми	Низька зручність використання, якщо алгоритм недостатньо довго тренувався
Модель демонструє швидке навчання порівняно з деякими іншими методами	Алгоритм передбачає кодування даних у двійкову форму, що може обмежувати гнучкість

Роздивимось методи, що використовують Fuzzy Logic (нечітку логіку). Нечітка логіка є важливим інструментом, який можна використовувати для вирішення дуже складних проблем, коли математична модель занадто складна або нездійсненна для побудови.

У процесі розробки програмного забезпечення завжди існують параметри, які володіють певним рівнем нечіткості. Дослідження показали, що використання нечіткої логіки може бути корисним при оцінці трудовитрат. Застосування нечіткої логіки здатне подолати деякі проблеми, які притаманні існуючим методам оцінки трудомісткості. Нечітка логіка не тільки допомагає прогнозувати трудомісткість, але й необхідна для того, щоб покращити якість

існуючих методів оцінки.

Нечітка логіка дозволяє лінгвістично представити вхідні та вихідні дані моделі, що є толерантними до неточності. Вона особливо підходить для оцінки трудовитрат, оскільки багато атрибутів програмного забезпечення вимірюються за номінальною або порядковою шкалою, що є окремим випадком лінгвістичних значень [5], [8].

Переваги та недоліки методу продемонстровано у табл. 1.15.

Таблиця 1.15 – Переваги та недоліки використання нечіткої логіки

Переваги	Недоліки
Розглядає стани з реальними значеннями, а не бінарними	Відсутність гарантії точності
Дозволяє враховувати і робити висновки на основі неповних або неточних даних	Не дуже добре працює зі складними наборами даних
Нечітка логіка є інтуїтивно зрозумілою і не вимагає складної математичної моделі	

Розглянемо методи, що засновані на використанні Bayesian networks (Байєсівських мереж). Метод оцінювання трудовитрат, заснований на ймовірнісних спрямованих ациклічних графових моделях, використовує графічні структури для представлення залежностей між змінними через спрямовані ациклічні графи.

Цей метод, також відомий як байєсівська експертиза, використовує байєсівський підхід для ймовірнісних обчислень і моделювання умовних залежностей. Крім того, модель використовує три основні завдання виведення: виведення неспостережуваних змінних, вивчення параметрів і вивчення структури. Кожне ребро в моделі відповідає умовній залежності, водночас кожен вузол відповідає унікальній випадковій змінній.

Таким чином, для досягнення функціональності моделі в процесі оцінювання вартості програмного забезпечення в ній зазвичай використовується певний шаблон [7].

Переваги та недоліки методу представлені у табл. 1.16.

Таблиця 1.16 – Переваги та недоліки використання Байєсівських мереж

Переваги	Недоліки
Є доволі точними	Є складними в проєктуванні
Дозволяють моделювати складні умовні залежності між різними змінними, що сприяє більш точним та реалістичним оцінкам	Витрачається багато часу на навчання
Кодують є всі змінні. У разі, якщо задіяні випадкові зв'язки, вони допомагають краще зрозуміти проблемну сферу, а також спрогнозувати наслідки в разі втручання	
Завдяки ймовірнісній і випадковій семантиці, ідеально підходять для представлення попередніх даних	

1.3 Постановка задачі

У ході аналізу існуючих методів оцінки трудовитрат при реалізації ІТ-проєктів були з’ясовані основні недоліки існуючих методів:

– алгоритмічні моделі мають велику залежність від початкових даних, які важко отримати на початковому етапі, або вимагають великої кількості кроків для обчислення трудовитрат. Крім того, мають обмеженість у використанні, наприклад, деякі не підходять для малих проєктів, або не

враховують нові технології або досвід команди;

– неалгоритмічні методи, або експертні, можуть бути достатньо точними, але вони обмежені рівнем професійності експертів та суб’єктивністю оцінки, можуть показувати меншу точність без попереднього досвіду;

– методи, що орієнтовані на навчання, є складними у проєктуванні. Вони можуть бути обмеженими з точки зору недостатньої якості навчальних даних та можливості надмірної адаптації до навчального набору даних, що може призвести до перенавчання та недооцінки нових даних.

Таким чином, основуючись на всьому вищезазначеному, можна визначити мету, об'єкт та предмет дослідження.

Метою роботи є аналіз методів оцінки трудовитрат при реалізації ІТ-проєктів та розробка власного методу оцінки трудовитрат, з урахуванням переваг і недоліків існуючих підходів.

Об’єктом дослідження є процес оцінювання трудовитрат при реалізації ІТ-проєктів.

Предметом дослідження є моделі та методи оцінювання трудовитрат при реалізації ІТ-проєктів.

Для досягнення мети необхідно вирішити наступні основні завдання:

– провести огляд існуючих методів оцінки трудовитрат та виявити їхні переваги та недоліки;

– розробити метод оцінки трудовитрат на виконання ІТ-проєкту, який буде достатньо ефективним та зрозумілим для використання на основі додавання або зміни метрик для поліпшення точності оцінки трудовитрат;

– провести тестування та оцінити ефективність розробленого методу;

– проаналізувати отримані результати.

2 РОЗРОБКА МОДИФІКОВАНОГО МЕТОДУ ОЦІНКИ ТРУДОВИТРАТ ПРИ РЕАЛІЗАЦІЇ ІТ-ПРОЄКТУ

Розглянемо етапи розробки власного методу оцінки трудовитрат при реалізації ІТ-проєкту на основі використання модифікації методу Use Case Points та методу PERT.

2.1 Оригінальний метод Use Case Points

Метод Use Case Points (UCP) – це методологія оцінки трудовитрат на розробку програмного забезпечення, що базується на аналізі функціональних вимог системи через використання прецедентів використання (use cases). Цей метод дає змогу оцінити обсяг роботи, необхідний для розроблення програмного продукту, і використовувати його в управлінні проєктами та плануванні ресурсів. Система Use Case Points була створена Густавом Карнером у 1993 році і заснована на порівняльних стандартах, подібних до процедури оцінки Functional Points (FP).

В основі даного методу лежить поняття «usecase» (варіант використання або прецедент). Варіант використання – це концепція, яку використовують у розробленні програмного забезпечення, дизайні продукту та інших галузях для опису того, як систему можна використовувати для досягнення конкретних цілей або завдань. У ньому описується взаємодія між користувачами або учасниками (акторами) та системою для досягнення конкретного результату [10].

Потрібно зазначити, що use case застосовується для специфікації загальних особливостей поведінки системи або іншої сутності без розгляду її внутрішньої структури (наприклад, оформлення замовлення на купівлю товару, отримання інформації про кредитоспроможність клієнта,

відображення графічної форми на екрані монітора).

Основна мета варіанту використання – допомагати донести стратегію до зацікавлених сторін і усунути розрив між бізнес-обґрунтуванням і технічними вимогами. Крім того, його призначення полягає у наступному:

– керувати обсягом проєкту;

– організувати функціональні вимоги;

– окреслити способи взаємодії користувача з системою;

– візуалізувати архітектуру системи.

Основними складовими прецедентів використання є актори, система, мета, основні та альтернативні потоки. Розглянемо кожну складову окремо.

Актор (або актант) – зовнішня по відношенню до модельованої системи сутність, яка взаємодіє із системою і використовує її функціональні можливості для вирішення певних завдань. При цьому актори слугують для позначення узгодженої множини ролей, які можуть грати користувачі в процесі взаємодії з проєктованою системою. Актантом може бути як людина, так і неживі об'єкти: комп'ютерна система, інша підсистема або ще якийсь вид об'єктів.

Оскільки в загальному випадку актор завжди перебуває поза системою, його внутрішня структура ніяк не визначається. Для актора має значення тільки його зовнішнє уявлення, тобто те, як він сприймається з боку системи. Актори взаємодіють із системою за допомогою передавання та приймання повідомлень від варіантів використання. Повідомлення являє собою запит актором сервісу від системи та отримання цього сервісу. Ця взаємодія може бути виражена за допомогою асоціацій між окремими акторами та варіантами використання або класами.

Система є комплексом або об'єднанням продуктів або послуг з визначеною функціональністю, які спрямовані на вирішення конкретної задачі або задоволення потреби користувача. Це може бути програмний продукт, апаратне забезпечення, мережеві сервіси або будь-яка інша технічна або не технічна система, яка має певну функціональність і використовується для

досягнення певних цілей.

Мета визначає те, що користувачі прагнуть досягти за допомогою функцій системи. Це конкретний результат або вигода, яку користувачі сподіваються отримати від використання системи. Мета може бути різноманітною: від вирішення конкретної проблеми або завдання до отримання інформації, полегшення робочих процесів або забезпечення зручності та комфорту для користувачів.

Базові потоки – це основні сценарії використання системи, при яких всі дії виконуються відповідно до очікуваного плану, і система досягає своєї мети. Вони представляють «ідеальний» шлях використання системи, коли немає ніяких відхилень або помилок. Базові потоки важливі для розуміння основних функцій системи і її основного призначення.

Альтернативні потоки (або розширення) – це сценарії використання системи, які відбуваються, коли є відхилення від базового потоку. Вони можуть включати різні ситуації, такі як обробка помилок, відхилення від стандартного поведінки або відповіді на особливі вимоги користувача. Альтернативні потоки важливі для розуміння того, як система повинна реагувати на різні ситуації, і допомагають забезпечити більш гнучке проектування системи.

Окрім вище перелічених складових, в варіантах використання можуть бути використані:

– основний актор: актор, який ініціює функцію системи для досягнення мети;

– передумови: фактори, що лежать в основі, необхідні для реалізації варіанту використання;

– післяумови: стан системи або результати, які очікуються після виконання варіанту використання. Вони визначають, що конкретно повинно бути досягнуто або змінено в системі після завершення варіанту використання [11].

При використанні методу UCP, є два основних підходи до представлення

варіантів використання: візуальне представлення за допомогою діаграм юзкейсів та текстовий опис юзкейсів. Обидва підходи мають свої переваги, і вибір між ними залежить від конкретних потреб та обставин.

Переваги використання діаграми варіантів використання:

– візуальне представлення. Діаграми юзкейсів надають візуальне зображення взаємодії між акторами та системою. Це не лише допомагає краще зрозуміти, які функції надає система, але й дозволяє відслідкувати, як користувачі взаємодіють із нею;

– простота розуміння. Діаграми юзкейсів є інтуїтивно зрозумілими та легко читабельними. Це робить їх доступними не лише для розробників, але й для менеджерів проекту, клієнтів та інших зацікавлених сторін;

– виявлення проблем. Завдяки візуальному поданню, діаграми юзкейсів дають змогу швидко виявити потенційні проблеми або поліпшення в проектованій системі. Це може бути особливо корисно на ранніх стадіях проекту, коли ще можливі значні зміни.

Переваги використання опису варіантів використання у вигляді тексту:

– більше деталей. У текстовому форматі можна дати докладніший опис кожного юзкейса, включно з вхідними та вихідними даними, кроками взаємодії та умовами. Це дозволяє більш глибоко розуміти процеси, що відбуваються в системі;

– гнучкість у створенні. Текстовий опис дозволяє більш гнучко підходити до створення сценаріїв використання. Він не обмежується просторовими рамками або строгими структурними вимогами, що дозволяє детально описувати різноманітні варіанти використання та поведінки системи;

– зручність для документування. Текстовий опис зручний для включення в документацію проєкту або вимоги, а також для обміну інформацією між членами команди проєкту. Це дозволяє забезпечити єдине розуміння процесів в системі серед всіх учасників проекту.

Приклад діаграми варіантів використання зображений на рисунку 2.1.

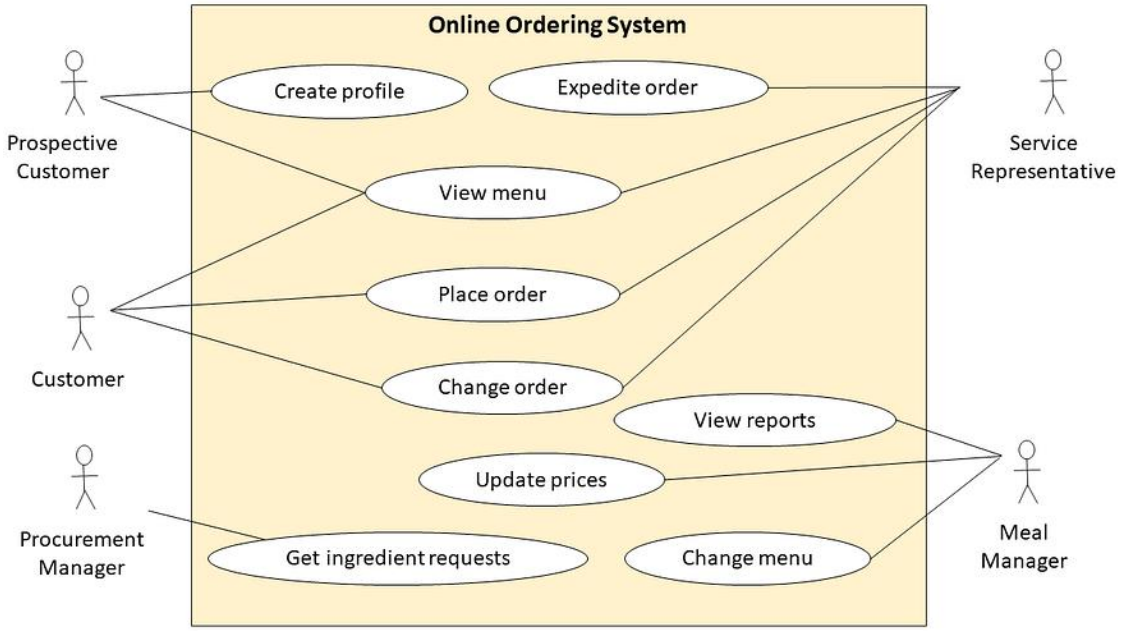

Рисунок 2.1 – Приклад діаграми варіантів використання

На основі компонентів варіантів використання системи визначається бал UCP, який використовується для кількісного визначення розміру програмного забезпечення, яке потім використовується для оцінки трудовитрат проєкту.

Загальна формула методу виглядає наступним чином:

$$UCP = (UUCP) \cdot TCF \cdot ECF, \tag{2.1}$$

де *UCP* – Use Case Points;

UUCP – некориговані точки варіантів використання;

TCF – фактор технічної складності;

ECF – фактор складності середовища.

UUCP розраховується за наступною формулою:

$$UUCP = (UAW + UUCW), \tag{2.2}$$

де *UAW* – нескоригована вага акторів;

UUCW – нескоригована вага варіантів використання.

Таким чином, можна сформувати алгоритм використання методу UCP:

– обчислення нескоригованої ваги акторів;

– обчислення нескоригованої ваги варіантів використання;

– обчислення нескоригованих точок варіантів використання;

– оцінка технічних факторів;

– оцінка зовнішніх чинників;

– остаточний підрахунок UCP.

Розглянемо більше детально кожний етап.

1. Обчислення нескоригованої ваги акторів (UAW). На цьому кроці визначають усіх акторів системи. Після визначення для кожного актора встановлюється вага, відповідно до його оцінкою. Актори бувають трьох типів: прості, середні і складні. У табл. 2.1 продемонстровано більш детальну інформацію щодо кожного типу акторів.

Таблиця 2.1 – Вагові коефіцієнти для акторів

Тип	Опис	Вага
Простий	Інша система із визначеним API	1
Середній	Або інша система яка взаємодіє через протокол, такий як TCP/IP, або людина, яка взаємодіє через текстовий інтерфейс	2
Складний	Особа, яка взаємодіє через графічний інтерфейс	3

Оцінка акторів відбувається за наступною формулою:

$$UAW = \Sigma_{i=1}^{3} a_i \times w_i, \tag{2.3}$$

де a_i – кількість акторів в і-му типі акторів;

w_i – вага для кожного типу.

2. Обчислення нескоригованої ваги варіантів використання (UUCW).

Кожен варіант використання класифікується як простий, середній або складний. Основою для цього рішення є кількість транзакцій у варіанті використання, включаючи альтернативні шляхи. Для цього транзакція визначається як атомарний набір дій, який або виконується повністю, або не виконується взагалі. У табл. 2.2 представлені вагові коефіцієнти для юзкейсів на основі транзакцій.

Таблиця 2.2 – Вагові коефіцієнти на основі транзакцій

Тип	Опис	Вага
Простий	3 або менше транзакцій	5
Середній	від 4 до 7 транзакцій	10
Складний	більше 7 транзакцій	15

Нескориговану оцінку варіантів використання, або UUCW можна розрахувати за формулою:

$$UUCW = \Sigma_{i=1}^{3} uc_j \times w_j, \quad (2.4)$$

де uc_j – кількість варіантів використання в j-му типі варіантів використання;

w_j – вага для кожного типу.

3. Обчислення нескоригованих точок варіантів використання (UUCP). Некоригована вага варіанта використання (UUCW) і нескоригована вага актора (UAW) разом дають нескоригований розмір системи, який називають нескоригованими точками варіанта використання. Наступні кроки – скоригувати нескориговані точки варіантів використання (UUCP) з урахуванням технічної складності та складності навколишнього середовища. UUCP можна розраховувати за формулою (2.2).

4. Обчислення фактору технічної складності (TCF). Оцінка технічних чинників використовується для визначення складності архітектури застосунку. Кожен фактор оцінюють за шкалою від 0 до 5. Оцінка 0 означає, що фактор не має значення для цього проекту, 5 означає, що він є важливим.

Перелік технічних факторів можна знайти в табл. 2.3.

Таблиця 2.3 – Показники технічної складності проєкту

Фактор	Назва	Ваговий коефіцієнт
T1	Розподілена система	2
T2	Висока продуктивність	2
T3	Ефективність роботи кінцевого користувача	1
T4	Складна обробка даних	1
T5	Повторне використання коду	1
T6	Легкість інсталяції	0,5
T7	Легкість використання	0,5
T8	Портативність	2
T9	Легкість корегування змін	1
T10	Паралельність	1
T11	Наявність спеціальних функцій безпеки	1
T12	Доступ з боку зовнішніх користувачів	1
T13	Вимоги до попереднього навчання користувачів	1

Показник TCF обчислюють за формулою:

$$TCF = 0{,}6 + (0{,}01 \times \Sigma_{i=1}^{13} t_i \times w_i), \quad (2.5)$$

де t_i – значення технічного фактору і;

w_i – вага фактору і.

5. Оцінка фактору зовнішніх чинників (ECF). Цей вид оцінки використовується для визначення коефіцієнта впливу організаційних ризиків на розробку. Обчислення проводяться за аналогією з технічними факторами.

Перелік технічних факторів можна знайти в табл. 2.4.

Таблиця 2.4 – Показники складності зовнішніх чинників

Фактор	Назва	Ваговий коефіцієнт
E1	Знайомство з UML	1,5
E2	Досвід роботи із конкретним середовищем	0,5
E3	Досвід використання ООП	1
E4	Кваліфікація системного аналітика	0,5
E5	Мотивація	1
E6	Стабільність вимог	2
E7	Неповний робочий день	-1
E8	Складність мови програмування	-1

Показник ECF обчислюють за формулою:

$$ECF = 1{,}4 - (0{,}03 \times \Sigma_{i=1}^{8} e_i \times w_i), \quad (2.6)$$

де e_i – значення фактору зовнішнього чинника і;

w_i – вага фактору і.

6. Остаточний підрахунок. На останньому етапі обчислень оцінюють загальну кількість варіантів за формулою (2.1) [12].

Однак, оцінка трудовитрат на цьому не закінчується. Правильний підхід до планування проекту передбачає оцінку його розміру та визначення тривалості на основі цієї оцінки. Точки використання застосовуються для оцінки розміру проекту, але їх недостатньо для безпосередньої відповіді на питання про трудовитрати проекту. Виходячи з оцінки розміру, необхідно визначити тривалість, враховуючи швидкість роботи команди над варіантами використання.

Карнер спочатку запропонував співвідношення 20 годин на один варіант використання, що означає, що для проекту з, наприклад, 545 варіантами

використання необхідно 10 900 годин роботи. Пізніші дослідження показали, що цей показник може варіюватися від 15 до 30 годин на один варіант використання. Інші дослідники запропонували метод визначення тривалості на основі факторів зовнішніх чинників. Вони пропонують підрахувати кількість факторів навколишнього середовища в E1-E6, які перевищують 3, а в E7 і E8 – менше трьох. Якщо кількість таких факторів дорівнює двом або менше, слід використовувати 20 годин на варіант використання. Якщо кількість факторів дорівнює 3 або 4, слід використовувати 28 годин на варіант використання. Якщо кількість факторів перевищує 4, проект слід призупинити для покращення умов середовища.

Однак, замість використання загальних рекомендацій, більш доцільним є розрахунок власних історичних середніх значень для організації на основі даних попередніх проектів. Створення сховища проектних даних для збереження таких показників дозволить точніше оцінювати майбутні проекти.

Для приблизного визначення тривалості проекту можна використовувати діапазон значень. Наприклад, діапазон від 20 до 28 годин на варіант використання за методикою Шнайдера і Вінтерса. Виходячи з досвіду написання варіантів використання та оцінок UCP, цей діапазон можна розширити або звузити. Використовуючи діапазон годин та кількість варіантів використання, можна визначити очікувану оцінку трудовитрат проекту [13].

Таким чином, остаточна оцінка трудовитрат проекту оцінюється за наступною формулою:

$$Effort = UCP \times ER, \tag{2.7}$$

де *Effort* – значення трудовитрат;

ER – коефіцієнт трудомісткості, який вказує на кількість людино-годин, необхідних для виконання одного UCP.

2.2 Метод PERT

Метод PERT (Project Evaluation and Review Technique) або метод оцінки и аналізу проєктів використовується різними організаціями вже понад 50 років. Спочатку розроблена в 1958 році для використання Офісом спеціальних проектів ВМС США, методика PERT є системою управління проєктами, призначеною для допомоги в плануванні великих і складних проєктів. Метод був використаний при розробці ракети «Поларіс», а також при проведенні зимових Олімпійських ігор 1968 року в Греноблі. Хоча подібні концепції управління розроблялися і вдосконалювалися з того часу, PERT був першою методикою такого роду.

PERT був розроблений головним чином для спрощення планування на папері та складання графіків великих і складних проєктів. Метод особливо націлений на аналіз часу, який потрібен для виконання кожного окремого завдання, а також на визначення мінімального необхідного часу для виконання всього проєкту.

Для ефективного планування проєктів важливо правильно оцінити часові затрати на кожну задачу. Однак, зазвичай виконавці мають тенденцію переоцінювати або недооцінювати час, не завжди оцінюючи його об'єктивно. Щоб отримати більш об'єктивні оцінки та врахувати можливі ризики, метод PERT використовує оцінку за трьома точками. Очікуваний час операції обчислюється за формулою:

$$t_c = \frac{O + 4M + P}{6}, \tag{2.8}$$

де t_c – очікуваний час операції;

O – оптимістичний час операції;

M – найбільш імовірний час операції;

P – песимістичний час операції.

Розглянемо наступні оцінки, які є складовими формули:

– оптимістична оцінка (optimistic time, O), що припускає мінімально

можливу тривалість завдання (усе складається краще, ніж очікується);

– песимістична оцінка (pessimistic time, P), що має на увазі максимально можливу тривалість завдання (усе складається гірше, ніж очікується);

– найбільш імовірна оцінка (most likely time, M), що передбачає тривалість завдання, за якої все відбувається як зазвичай.

Метод PERT передбачає, що тривалість кожної операції змінюється в межах деякого статистичного розподілу. Для розрахунку тривалості операцій використовуються три зазначені оцінки (O, M і P). Таким чином, вона може випадковим чином змінюватися від оптимістичного (найкращого) значення до песимістичного (найгіршого), і для кожної операції можна розрахувати середній, або зважений показник.

Для вираження тривалості операції розробники PERT обрали апроксимацію у вигляді функції бета-розподілу, головним чином несиметричної форми. Підставою для такого вибору стало те, що робота (виконання операції) може відставати від графіка, і якщо в певний момент це трапилося, то відставання триватиме й далі, можливо, збільшуючись.

Таким чином, середнє значення розподілу визначають як середньозважене мінімальне, найвірогідніше й максимальне значення, яких може набувати змінна, із чотирикратною вагою, застосованою до найвірогіднішого значення, що дає змогу уникнути виникнення перекосу за одним із наявних напрямів. Це припущення про середнє значення було вперше запропоновано Кларком у 1962 році для оцінки впливу невизначеності тривалості завдань на результат графіка проєкту, оцінюваного за допомогою методу оцінки та аналізу проєкту, звідси і його назва.

Найімовірніша оцінка має найбільшу ймовірність (близько 0,5 або 50%), а песимістична й оптимістична – близько 0,1 або 10%. Оскільки ми маємо справу з часом, коротша тривалість являє собою оптимізм, а триваліша – песимізм.

Ще одна особливість бета-розподілу – це перекіс (згладжування праворуч), який говорить про те, що (у більшості випадків) оцінка схиляється

в бік оптимізму, а не в бік песимізму [14].

На рис. 2.2 зображений графік PERT-розподілу (або бета):

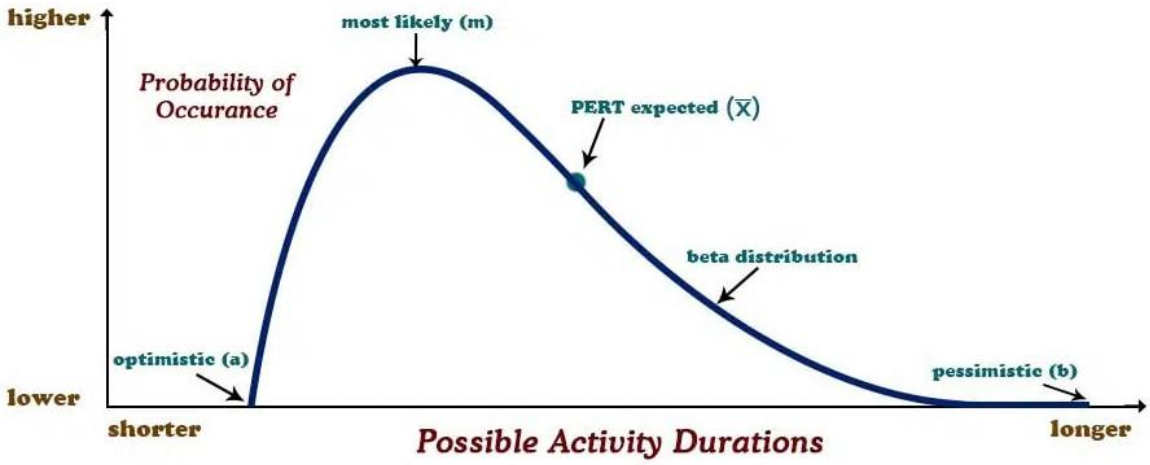

Рисунок 2.2 – PERT-розподіл

Одною з складових частин PERT є побудова мережевого графу (мережевої діаграми PERT) – діаграми взаємозв'язків робіт і подій, необхідних для завершення проєкту. Приклад такої діаграми представлено на рис. 2.3:

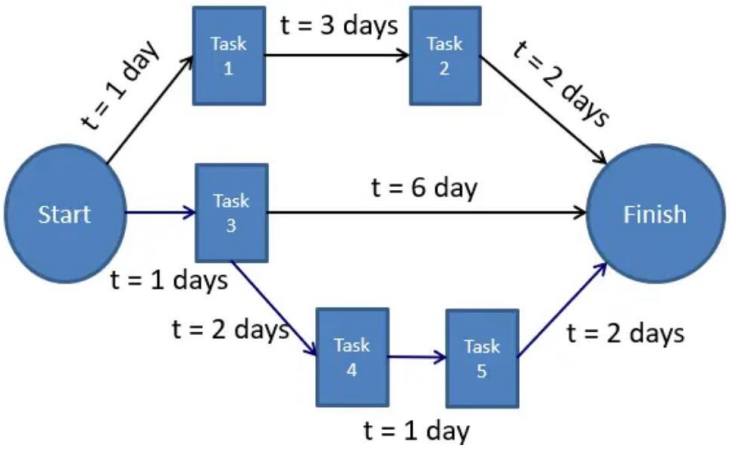

Рисунок 2.3 – Мережева діаграма PERT

В цілому, метод PERT складається з наступних кроків:

– визначити конкретні дії: перший крок – визначити, що необхідно

зробити для завершення роботи. Кожне завдання має бути чітким, вимірним і мати певну кількість часу;

– визначити правильну послідовність дій: потрібно з'ясувати, як завдання залежать одне від одного. Деякі речі необхідно робити по порядку, а інші можуть працювати одночасно. Цей крок допоможе визначити, як різні завдання пов'язані одне з одним і як вони впливають на графік проєкту;

– оцінити тривалість операції: потрібно з’ясувати, скільки часу знадобиться на виконання кожного завдання. Це можна зробити, поговоривши з експертами в цій галузі, використовуючи дані з минулого або використовуючи наукові методи. Щоб скласти весь графік проєкту, потрібно знати точні оцінки того, скільки часу він займе;

– скласти схему мережі. Створити мережеву діаграму, що показує, як завдання працюють разом і як вони залежать одне від одного. На цій діаграмі зазвичай використовуються вузли для позначення дій і стрілки для відображення потоку і порядку. Це допоможе зрозуміти ключовий шлях і з'ясувати, які завдання можна перенести вгору або назад, не змінюючи дату завершення проєкту;

– знайти критичний шлях. Критичний шлях – найдовший ланцюжок взаємозалежних завдань, який визначає, скільки часу займе вся робота. Будь-яка затримка в діях на критичному шляху безпосередньо вплине на завершення проєкту. Це допомагає розставити пріоритети і вирішити, як використовувати ресурси;

– розклад і контроль. Скласти розклад, використовуючи очікуваний час і результати аналізу критичного шляху. Графік показує, як виконуватиметься робота, і дає змогу відстежувати хід виконання та керувати ним. Регулярне відстеження, моніторинг і зміни необхідні для забезпечення того, щоб проєкт продовжував іти за планом і щоб будь-які проблеми вирішували швидко [15].

2.3 Модифікація методу Use Case Points із використанням методу PERT

Оцінка трудовитрат на ранніх стадіях розробки програмного забезпечення відіграє критично важливу роль в успішному виконанні проекту. Це дозволяє команді розробників прийняти відповідні рішення щодо планування. Втім, через недостатність наявної інформації, процес оцінки часто характеризується високим рівнем невизначеності. Для зменшення цієї проблеми, менеджери використовують функціональні вимоги як вихідні дані для проведення попередньої оцінки зусиль. Розмір програмного проекту дає загальне уявлення про те, наскільки великий програмний проект, і, таким чином, вказує на те, скільки ресурсів знадобиться проекту для досягнення успішних результатів.

Метод Use Case Points був обраний для дослідження через його специфічні характеристики та можливості, які він пропонує. Цей метод відрізняється своєю алгоритмічною природою, що забезпечує стабільність та прогнозованість при оцінці трудовитрат. Він також враховує функціональні вимоги до системи, що робить його незамінним інструментом на початкових стадіях проекту.

Особливо важливо відзначити, що UCP ідеально підходить для проєктів, що базуються на об'єктно-орієнтованому програмуванні (ООП), завдяки його здатності до адаптації до різних мов програмування. Це свідчить про його універсальність і гнучкість у використанні.

Проте, не дивлячись на ці переваги, UCP був проаналізований дослідниками та має певні обмеження, зокрема, не завжди високу точність в оцінці трудовитрат. Деякі науковці висловили зацікавленість у підходах, заснованих на використанні варіантів використання. З цієї причини було запропоновано багато методів, які є модифікаціями UCP. Більшість з них зосереджені лише на зміні значення нескоригованої точки використання (UUCP), яка представляє вагу, присвоєну акторам і варіантам використання.

Однак, крім UUCP на точність оцінки впливають ще два інші коригувальні фактори – TCF та ECF. При визначенні значень цих коригувальних коефіцієнтів виникають певні труднощі, оскільки часто

відсутній базовий рівень для порівняння. Керівники проектів можуть зіткнутися з необхідністю інтерпретувати кожен фактор та згадувати інші проекти для порівняння з поточним. Так, деякі з коригувальних коефіцієнтів можуть бути неоднозначними. Наприклад, фактор «Паралельність» може включати в себе паралельну обробку, паралельне програмування або взаємодію системи з іншими додатками. Відсутність чітких вказівок в моделі UCP щодо того, що саме цей фактор повинен вимірювати, може призвести до неточностей в його оцінці [12].

Таким чином, можна зробити висновок, що TCF та ECF можуть значно впливати на точність оцінки і, отже, потребують додаткового аналізу. Незначна зміна вагового значення цих коефіцієнтів може суттєво вплинути на оцінку розміру програмного забезпечення і, відповідно, на оцінку трудомісткості.

Для вирішення цих проблем запропоновано використовувати метод PERT для розрахунку вказаних факторів. PERT допомагає краще управляти невизначеністю, враховуючи різні можливі сценарії – оптимістичний, песимістичний та найбільш ймовірний. Це дозволяє отримати більш точну і збалансовану оцінку. Завдяки PERT, менеджери проекту можуть визначити діапазон можливих оцінок для кожного фактору, що дозволяє вибрати найбільш ймовірну оцінку, замість простого призначення оцінки.

Оскільки принцип обчислення TCF та ECF однаковий, то сформуємо загальну формулу, за якою буде вираховуватись очікуване значення кожного фактору:

$$f_i = \frac{O_i + 4M_i + P_i}{6}, \tag{2.9}$$

де f_i – середньозважене значення фактору і;

O – оптимістична оцінка фактору і;

M – найбільш вірогідна оцінка фактору і;

P – песимістична оцінка фактору і.

Тоді, нові формули обчислення TCF та ECF виглядають наступним чином:

$$TCF = 0{,}6 + (0{,}01 \times \Sigma_{i=1}^{13} f_i \times w_i), \quad (2.10)$$

$$ECF = 1{,}4 - (0{,}03 \times \Sigma_{i=1}^{8} f_i \times w_i), \quad (2.11)$$

де f_i – очікуване значення технічного фактору і;

де f_i – очікуване значення фактору зовнішніх чинників і.

Алгоритм, що реалізує модифікований метод Use Case Points можна описати таким чином.

– обчислити нескориговану вагу акторів (UAW) за формулою (2.3);

– обчислити нескориговану вагу варіантів використання (UUCW) за формулою (2.4);

– обчислити нескориговані точки варіантів використання (UUCP) за формулою (2.2);

– обчислити оцінку технічних факторів із використанням методу PERT за формулою (2.10);

– обчислити оцінку зовнішніх чинників із використанням методу PERT за формулою (2.11);

– обчислити остаточну оцінку трудовитрат проєкту за формулами (2.1) та (2.7).

В рамках даної роботи була розроблена діаграма IDEF0 для процесу оцінки трудовитрат за методом UCP з модифікацією PERT. Діаграма допомогла визначити ключові етапи процесу, вхідні та вихідні дані, а також управляючі системи.

На рис. 2.4 зображена контекстна діаграма IDEF0, яка відображає процес оцінки трудовитрат за методом UCP з модифікацією PERT на високому рівні. Вхідні дані для процесу – це вимоги замовника, які подаються на вхід процесу

оцінки. Вихідні дані – це розраховані трудовитрати для проєкту, які є результатом процесу. Механізми процесу включають менеджера, бізнес-аналітика та розробника, які виконують процес оцінки. Управляючі параметри включають правила використання UCP, правила використання PERT та коефіцієнт трудомісткості, які впливають на виконання процесу оцінки.

Рисунок 2.4 – Контекстна діаграма IDEF0

На рис. 2.5 зображена декомпозиція контекстної діаграми IDEF0, яка деталізує процес оцінки трудозатрат за методом UCP з модифікацією PERT. Діаграма розбиває основний процес на шість ключових етапів, кожен з яких представляє окрему функцію в процесі оцінки. Цей процес повторює алгоритм розрахунку трудовитрат за модифікованим методом UCP, але окрім цього використовує вхідні та вихідні дані, механізми та управляючі параметри.

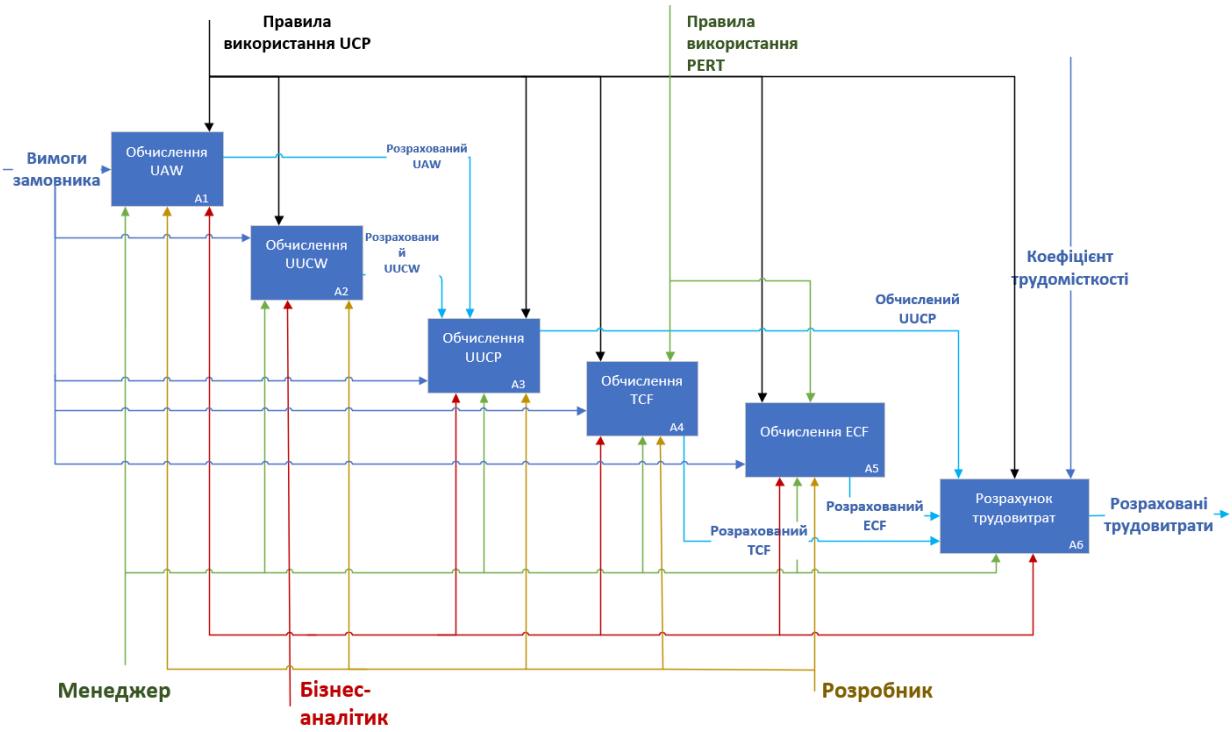

Рисунок 2.5 – Декомпозиція діаграми IDEF0

3 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА МОЖЛИВОСТЕЙ ВИКОРИСТАННЯ РОЗРОБЛЕНОГО МЕТОДУ ОЦІНЮВАННЯ ТРУДОВИТРАТ ДЛЯ ІТ-ПРОЄКТУ

Для демонстрації роботи методу, описаного в підрозділі 2.3, було запропоновано розглянути модель проєкту платформи для онлайн-навчання. Цей проект включає:

– різноманітні функціональні можливості;

– сценарії використання, що дозволяють наочно продемонструвати процес оцінки складності програмного забезпечення за допомогою поєднання методів Use Case Points та PERT.

В епоху цифровізації та стрімкого розвитку технологій онлайн-навчання стає все більш популярним і затребуваним. Проект «Платформа для онлайн-навчання» спрямований на створення зручної, функціональної та ефективної системи, яка дозволить студентам і викладачам взаємодіяти в рамках освітнього процесу, забезпечуючи доступ до якісних навчальних матеріалів, курсів та вебінарів.

Метою проєкту є розробка інтегрованої онлайн-платформи для навчання, яка підтримує велику кількість функцій, необхідних для повноцінного навчального процесу. Платформа повинна надати можливості для реєстрації та входу користувачів, управління курсами, проведення вебінарів та онлайн-лекцій, взаємодії студентів і викладачів, а також відстеження прогресу навчання.

3.1 Обчислення нескоригованої ваги акторів

Розглянемо основних акторів, які будуть застосовуватись в системі: студент, викладач, адміністратор, платіжна система та email-сервіс.

Відповідно до табл. 2.1 визначимо до яких типів відносяться актори (табл. 3.1).

Таблиця 3.1 – Визначення типів акторів для платформи онлайн-навчання

Актор	Тип	Бал
Студент	Складний	3
Викладач	Складний	3
Адміністратор	Складний	3
Платіжна система	Простий	1
Email-сервіс	Простий	1

Розрахуємо UAW за формулою (2.3):

$$UAW = (1\times2) + (2\times0) + (3\times3) = 11.$$

3.2 Обчислення нескоригованої ваги варіантів використання

Нижче, у табл. 3.2 перелічені основні юзкейси для перших трьох акторів, які є безпосередніми користувачами системи.

Таблиця 3.2 – Варіанти використання для студентів

Назва	Опис
1	2
Реєстрація	Студент вводить свої дані в форму реєстрації на платформі та підтверджує свою електронну пошту

Кінець таблиці 3.2

1	2

Логін	Вхід у систему за допомогою введення логіна та пароля.
Логаут	Вихід із системи
Реєстрація на курс	Студент записується на обраний курс
Виконання завдань курсу	Студент виконує завдання, запропоновані в курсі
Завантаження виконаних завдань	Студент завантажує виконані завдання на платформу
Виконання фінального завдання	Студент після навчання виконує фінальне завдання, після чого його курс закінчується
Отримання сертифіката	Студент отримує сертифікат після успішного завершення курсу
Відстеження прогресу	Студент переглядає свій навчальний прогрес, включаючи виконані завдання, отримані оцінки та відвідуваність
Відправка повідомлень	Студент надсилає повідомлення або електронні листи викладачу для отримання консультацій та роз'яснень
Спілкування на форумі	Студент бере участь у дискусіях на форумі, створює нові теми та залишає коментарі
Оцінювання курсу	Студент оцінює курс та залишає відгук про якість викладання та матеріали
Оплата курсу	Студент вводить платіжні дані та підтверджує оплату за курс
Редагування профілю	Студент змінює інформацію у своєму профілі, включаючи особисті дані та налаштування
Зв'язок з адміністраторами	Студент може зв’язатися із адміністраторами для вирішення проблем
Пошук курсів	Студент може шукати курси, які його цікавлять
Перегляд списку курсів	Студент може переглядати список всіх курсів, які доступні для нього
Перегляд сторінки конкретного курсу	Студент може переглядати інформацію щодо конкретного курсу

Таблиця 3.3 – Варіанти використання для викладачів

Назва	Опис
1	2
Реєстрація	Викладач вводить свої дані в форму реєстрації та підтверджує свою електронну пошту
Логін	Викладач виконує вхід у систему з використанням логіна та пароля
Логаут	Викладач виконує вихід із системи
Управління курсами	Викладач управляє курсами, включаючи створення, оновлення та видалення курсу.
Перегляд списку студентів	Викладач переглядає список студентів, зареєстрованих на його курси.
Завантаження навчальних матеріалів	Викладач завантажує лекції, відео, презентації та інші матеріали на платформу.
Перевірка завдань	Викладач перевіряє виконані студентами завдання і виставляє оцінки.
Організація вебінарів/онлайн-лекцій	Викладач планує та проводить онлайн-заняття.
Спілкування на форумі	Викладач бере участь у дискусіях на форумі та залишає коментарі.
Відправка повідомлень	Викладач відправляє та отримує повідомлення від студентів або інші важливі оголошення.
Редагування профілю	Студент змінює інформацію у своєму профілі, включаючи особисті дані та налаштування.
Зв'язок з адміністраторами	Викладач може зв’язатися із адміністраторами для вирішення проблем.
Пошук курсів	Викладач може шукати курси, які його цікавлять.
Перегляд списку курсів	Викладач може переглядати список всіх курсів, які доступні для нього.
Перегляд сторінки конкретного курсу	Викладач може переглядати інформацію щодо конкретного курсу.

Таблиця 3.4 – Варіанти використання для адміністраторів

Назва	Опис
1	2
Логін	Адміністратор входить в систему за допомогою облікових даних.
Логаут	Адміністратор виходить із системи
Управління користувачами	Адміністратор ефективно керує користувачами системи, включаючи редагування користувачів, їх видалення
Проведення аналітики та звітності	Адміністратор переглядає звіти та статистику щодо активності користувачів та успішності курсів, а також надає звіти про діяльність платформи керівництву
Управління курсами	Адміністратор переглядає всі курси на платформі, а також може редагувати та видаляти їх
Обслуговування платформи	Адміністратор налаштовує системні параметри платформи, виконує своєчасні оновлення системи
Забезпечення підтримки користувачів	Адміністратор відповідає на запити та вирішує технічні проблеми користувачів
Забезпечення безпеки	Адміністратор відстежує безпеку платформи та вживає заходів для запобігання порушенням

На рис. 3.1 представлена діаграма юзкейсів для даного проекту.

У зв'язку з високою складністю і багатогранністю системи, яка містить у собі безліч варіантів використання, на діаграмі варіантів використання було непросто відобразити всі можливі зв'язки. Це обумовлено тим, що включення всіх зв'язків між варіантами використання могло б призвести до перевантаженості та незрозумілості діаграми. Якщо б була спроба включити всі можливі зв'язки, діаграма швидко стала б надто заплутаною і важкою для розуміння.

Тому було ухвалено рішення сфокусуватися на основних і найбільш значущих зв'язках, забезпечуючи тим самим чіткість і зрозумілість подання. Оскільки це і є метою складання даної діаграми – зрозумілість та

інформативність, але при цьому не навантаженість деталями, що дозволяє краще зрозуміти систему та її взаємодію з користувачами.

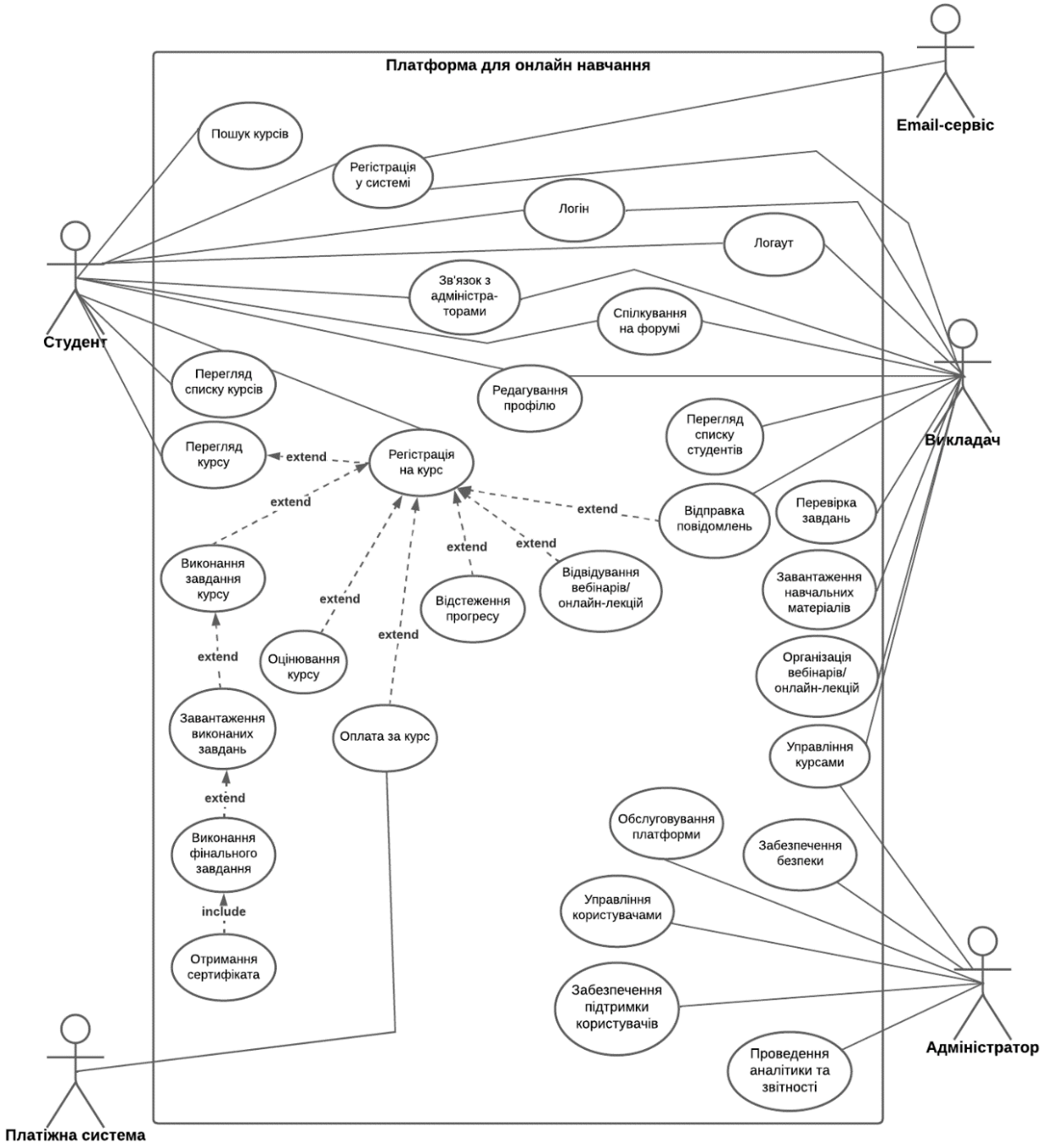

Рисунок 3.1 – Діаграма варіантів використання платформи онлайн-навчання

Кожен варіант використання в цьому проекті описується відповідно до його специфічних вимог і контексту. В якості прикладу можна

продемонструвати опис юзкейсу «Оплата курсу».

Назва варіанту використання: Оплата курсу на онлайн-платформі навчання.

Основний актор: Студент.

Додатковий актор: Платіжна система.

Передумова: Студент вже зареєстрований на платформі та вибрав курс для оплати.

Післяумова: Курс оплачений; студент отримує підтвердження оплати на електронну пошту

Основний сценарій успіху:

– студент підтверджує дані своєї платіжної картки (1);

– система перевіряє введені дані (2);

– платіжна система обробляє транзакцію (3);

– система підтверджує оплату курсу (4);

– email-сервіс надсилає студенту лист із підтвердженням оплати (5);

– студент переходить до перегляду курсу (6).

Розширення:

– 2a. Дані платіжної картки введені невірно;

– 2a1. Система повідомляє студента ввести дані картки знову;

– 2b. Термін дії картки закінчився;

– 2b1. Система повідомляє студента ввести дані картки знову;

– 3a. Платіжна система відхиляє транзакцію;

– 3a1. Система повідомляє студента про проблему з оплатою.

Для визначення кількості транзакцій у кожному варіанті використання необхідно проаналізувати його структуру, включно з основним сценарієм успіху та всіма можливими розширеннями, що є альтернативними шляхами виконання варіанту використання. Транзакція, у контексті варіантів використання, еквівалентна кроку у варіанті використання. Перший крок у кожному розширенні, який є результатом попередньої транзакції, не враховується. Однак, всі наступні кроки в розширенні, які представляють нові

транзакції, враховуються. Таким чином, не всі транзакції з розширення просто додаються, а лише ті, які є новими транзакціями.

Розрахуємо кількість транзакцій для даного варіанту використання та визначимо до якого типу він відноситься. Отримуємо: 6 транзакцій в основному сценарії, 1 транзакція в 2a, 1 транзакція в 2b, 1 транзакція в 3a. Усього виходить 9 транзакцій. Це дорівнює 9, тому варіант використання вважається складним.

Аналогічним чином розрахуємо інші юзкейси. Ті варіанти використання, які одночасно є у студента, викладача та адміністратора враховується 1 раз. Таким чином:

– кількість простих юзкейсів складає 17;

– кількість середніх юзкейсів складає 8;

– кількість складних юзкейсів складає 5.

За формулою (2.4) розрахуємо UUCW:

$$UUCW = (17 \times 5) + (8 \times 10) + (5 \times 15) = 240$$

3.3 Обчислення нескоригованих точок варіантів використання

Нескориговані точки варіантів використання обчислюються за формулою (2.2). Таким чином, маємо наступний результат:

$$UUCP = 11 + 240 = 250.$$

3.4 Обчислення фактору технічної складності

Перейдемо до визначення оцінки технічних факторів.

– T1 – розподілена система (4). Платформа онлайн-навчання потребує розподіленої системи для обробки великого обсягу даних (курси, користувачі, контент), забезпечення масштабованості та надійності. Збереження даних, обробка запитів та виконання завдань повинні бути розподілені між кількома серверами, щоб забезпечити доступність та швидкість роботи системи;

– T2 – висока продуктивність (4). Система повинна бути швидкою і ефективною, особливо під час пікового навантаження;

– T3 –ефективність роботи кінцевого користувача (3). Ефективність роботи користувача залежить від зручності інтерфейсу, навігації та простоти використання. Важливо, щоб користувач міг легко знайти потрібну інформацію, виконати завдання та переглянути результати. Проте, це не найважливіший фактор, оскільки основний акцент платформи – на наданні контенту та функціональності для навчання;

– T4 – складна обробка даних (4). Платформа онлайн-навчання обробляє різні типи даних, такі як: інформація про користувачів, контент курсів, результати навчання, прогрес користувачів, статистика використання. Це вимагає складної обробки та структурування даних;

– T5 – повторне використання коду (4). Повторне використання коду є важливим фактором для зменшення часу розробки, покращення якості та підтримки коду. Наприклад, модулі для обробки користувачів, оплати або відстеження прогресу можуть бути повторно використані в різних частинах платформи;

– T6 – легкість інсталяції (3). Інсталяція платформи може бути складною, особливо для розподіленої системи. Однак, важливо, щоб вона була якомога простішою та зручнішою для адміністратора системи;

– Т7 – легкість використання (4). Важливо, щоб система була інтуїтивно зрозумілою для користувачів, незалежно від їхнього рівня комп'ютерної грамотності. Це включає чіткий інтерфейс, просту навігацію та контекстну допомогу;

– Т8 – портативність (3). Платформа онлайн-навчання повинна працювати на різних пристроях (комп'ютери, планшети, смартфони) та операційних системах. Це може вимагати використання мобільних версій, адаптивного дизайну та крос-платформних технологій, але не є критичним фактором;

– Т9 – легкість корегування змін (3). Фактор важливий для швидкого оновлення контенту, додавання нових функцій та виправлення помилок;

– Т10 – паралельність (4). Використання паралельної обробки даних дозволяє ефективно розподілити завдання між різними процесорами, що покращує продуктивність системи. Це особливо важливо для обробки великих обсягів даних, наприклад, під час перевірки завдань або аналізу статистики;

– Т11 – наявність спеціальних функцій безпеки (4). Платформа онлайн-навчання повинна забезпечувати високий рівень безпеки для захисту даних користувачів, контенту курсів та фінансової інформації. Це включає автентифікацію користувачів, шифрування даних, контроль доступу та інші заходи безпеки;

– Т12 – доступ з боку зовнішніх користувачів (3). Платформа може потребувати надання доступу до зовнішніх користувачів (наприклад, батькам студентів, роботодавцям). Важливо забезпечити безпечний та контрольований доступ до системи для таких користувачів;

– Т13 – вимоги до попереднього навчання користувачів (2). Система повинна бути інтуїтивно зрозумілою для користувачів, щоб не вимагати складного попереднього навчання. Однак, для деяких функцій (наприклад, створення курсів, управління контентом) може знадобитися навчання або технічна підтримка.

Будемо вважати, що оцінки для кожного фактору є найбільш вірогідними оцінками (М). У табл. 3.5 визначимо відповідно оптимістичні (О) та песимістичні (Р) оцінки, та за формулою (2.9) розрахуємо очікувану оцінку для кожного фактору (t_c):

Таблиця 3.5 – Розрахунок факторів технічної складності за методом PERT

Фактор	Назва	Ваговий коефіцієнт	О	М	Р	t_c
Т1	Розподілена система	2	3	4	5	4
Т2	Висока продуктивність	2	3	4	5	4
Т3	Ефективність роботи кінцевого користувача	1	2	3	5	3,17
Т4	Складна обробка даних	1	2	4	5	3,83
Т5	Повторне використання коду	1	3	4	5	4
Т6	Легкість інсталяції	0,5	1	3	4	2,83
Т7	Легкість використання	0,5	2	4	5	3,83
Т8	Портативність	2	2	3	4	3
Т9	Легкість корегування змін	1	2	3	4	3
Т10	Паралельність	1	3	4	5	4
Т11	Наявність спеціальних функцій безпеки	1	3	4	5	4
Т12	Доступ з боку зовнішніх користувачів	1	2	3	4	3
Т13	Вимоги до попереднього навчання користувачів	1	1	2	3	2

Для кращої наочності виділимо з формули (2.10) частину сумування факторів:

$$(4\times2) + (4\times2) + (3{,}17\times1) + (3{,}83\times1) + (4\times1) + (2{,}83\times0{,}5) + (3{,}83\times0{,}5) + (3\times2) + (3\times1) + (4\times1) + (4\times1) + (3\times1) + (2\times1) = 52{,}33.$$

За формулою (2.10) розрахуємо TCF:

$$TCF = 0{,}6 + (0{,}01 \times 52{,}33) = 1{,}12.$$

3.5 Оцінка фактору зовнішніх чинників

Аналогічно попередньому кроку проставимо оцінки для факторів зовнішніх чинників:

– E1 – знайомство з UML (4). Важливий фактор, оскільки UML є стандартом для моделювання та документації систем;

– E2 – досвід роботи із конкретним середовищем (3). Досвід роботи із конкретним середовищем (наприклад, мовою програмування, фреймворком, платформою) є важливим, але не критичним. Завжди можна навчитися новому, і доступність ресурсів для навчання може компенсувати брак досвіду;

– E3 – досвід використання ООП (4). Важливий, оскільки більшість сучасних мов програмування базуються на ООП;

– E4 – кваліфікація системного аналітика (4). Системний аналітик відіграє ключову роль у визначенні вимог та архітектури системи. Його досвід та навички мають велике значення для успіху проекту;

– E5 – мотивація (4). Важливий фактор, який впливає на продуктивність та якість роботи;

– E6 – стабільність вимог (3). Вимоги до платформи онлайн-навчання можуть змінюватися, але очікується, що вони будуть відносно стабільними.

Незначні зміни вимог можна очікувати, але значні зміни можуть негативно вплинути на проект;

– E7 – неповний робочий день (1). Неповний робочий день може впливати на швидкість розробки, але не є критичним. Якщо розробники працюють ефективно, то неповний робочий день не буде серйозною проблемою;

– E8 – складність мови програмування (2). Складність мови програмування може вплинути на час розробки, але не є надзвичайно проблематичною. Завжди можна використовувати різні інструменти та бібліотеки, які спрощують розробку.

Аналогічно табл. 3.5 створимо табл. 3.6.

Таблиця 3.6 – Оцінка факторів зовнішніх чинників за методом PERT

Фактор	Назва	Ваговий коефіцієнт	O	M	P	t_c
E1	Знайомство з UML	1,5	3	4	5	4
E2	Досвід роботи із конкретним середовищем	0,5	3	3	5	3,33
E3	Досвід використання ООП	1	2	4	5	3,83
E4	Кваліфікація системного аналітика	0,5	2	4	5	3,83
E5	Мотивація	1	3	4	5	4
E6	Стабільність вимог	2	1	3	4	2,83
E7	Неповний робочий день	-1	2	1	4	1,67
E8	Складність мови програмування	-1	2	2	4	2,33

Для кращої наочності виділимо з формули (2.11) частину сумування факторів:

$$(6\times1,5) + (1,58\times0,5) + (3,83\times1) + (1,92\times0,5) + (4\times1) + (5,67\times2) - (1,67\times1) - -(2,33\times1) = 19.$$

За формулою (2.11) розрахуємо ECF:

$$ECF = 1,4 + (-0,03 \times 19) = 0,83.$$

3.6 Підрахунок UCP та трудовитрат

Тепер, розрахуємо UCP за формулою (2.1):

$$UCP = (11+240) \times 1,12 \times 0,83 = 232,4.$$

Для обчислення трудовитрат, які потребуються на проект, оберемо коефіцієнт зусиль, який вказує на кількість людино-годин, необхідних для виконання одного UCP. Зазвичай, використовують значення ER 20 одиниць. Таким чином, за формулою (2.7) маємо:

$$Effort = 232,4 \times 20 = 4648 \text{ (людино-годин)}.$$

Одиницю виміру людино-години можна перевести у людино-дні, -тіжні або -місяці, що дає для замовника системи більше розуміння термінів розробки.

4 ЕКСПЕРИМЕНТАЛЬНЕ МОДЕЛЮВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Порівняння точності оригінального UCP та модифікованого методів оцінки трудовитрат

Наступною фазою в процесі розробки є етап перевірки ефективності використання метода. Ця стадія є критичною для оцінки придатності нового метода та визначення його ефективності в реальних умовах. Процес аналізу даних забезпечує важливу інформацію для подальшого вдосконалення та розвитку методу. Використання реальних даних дозволяє отримати об'єктивну оцінку та забезпечує більш достовірні результати. Такий підхід сприяє зростанню наукової обґрунтованості та вірогідності отриманих висновків.

З цією метою був використаний датасет, який був виявлений у відкритому доступі та який забезпечується Університетом Томаша Баті в місті Злін, Чехія. Відповідно до джерела, ці дані були зібрані з трьох компаній-розробників програмного забезпечення та є реальним набором інформації. Для аналізу кількості кроків або акторів було використано метод UCP.

Датасет складається з 18 атрибутів:

– project_No –ідентифікатор проекту для цілей ідентифікації;

– simple actors – кількість акторів, класифікованих за UCP – прості актори;

– average actors – кількість акторів, що класифікуються відповідно до UCP – середні актори;

– complex actors – кількість акторів, що класифікуються відповідно до UCP – складні актори;

– uaw – нескоригована вага актора, обчислена за допомогою рівняння UCP;

– simple UC – кількість варіантів використання, класифікованих як прості – використовується кількість кроків UCP;

– average UC – кількість варіантів використання, класифікованих як середні – використовується кількість кроків UCP;

– complex UC – кількість варіантів використання, класифікованих як складні – використовується кількість кроків UCP;

– uucw – нескоригована вага варіантів використання – обчислюється за допомогою рівняння UCP;

– tcf – фактор технічної складності;

– ecf – фактори екологічної складності;

– real_p20 – реальні зусилля в людино–годинах, визначаються за допомогою коефіцієнта продуктивності (PF = 20);

– real_effort_person_hours – реальні зусилля (час розробки) в людино-годинах;

– sector – проблемна область проекту;

– language – мова програмування, що використовується для проекту;

– methodology – методологія розробки, що використовується для розробки проекту;

– applicationtype – класифікація типу проекту – надається донором;

– datadonator – анонімізована абревіатура для донора даних.

Всього в датасеті зібрана інформація щодо 71 проєкту.

Для ефективного тестування техніки, яка описується в кваліфікаційній роботі, ключовими атрибутами є Project_No, Simple Actors, Average Actors, Complex Actors, UAW, Simple UC, Average UC, Complex UC, TCF, ECF, Real_P20 та Real_Effort_Person_Hours. Ці параметри становлять основу для отримання інформації щодо кожного проекту, в той час як інші атрибути можуть бути менш важливими для перевірки ефективності методу.

Оскільки в описаній техніці, зазначеній в пункті 2.3, передбачається використання методу PERT для розрахунку TCF та ECF, необхідно мати інформацію про кількість балів, що присвоєно кожному конкретному фактору. Однак, зазначений масив даних не містить таких деталей, тому було вирішено розрахувати ці значення вручну. Хоча ці розрахунки можуть не бути

абсолютно точними, вони найбільш наближені до реальних показників.

Для тестування було випадковим чином обрано 21 проєкт з 71, що дозволило продемонструвати результати. Далі були розраховані UCP за визначеною формулою (2.1).

Наступним кроком є обчислення точності нового методу та порівняння його із оригінальним. Для цього використовувались наступні метрики:

– MMRE (Mean Magnitude of Relative Error) – середня величина відносної похибки;

– PRED(x) (Percentage of Prediction within x%) – відсоток передбачення в межах x%;

– MAE (Mean Absolute Error) – середня абсолютна похибка.

Точність оцінки трудовитрат з точки зору MMRE та PRED(x) – це дві найпоширеніші метрики, що використовуються при розробці програмного забезпечення. Обидва параметри засновані на величині, яка називається величиною відносної похибки (MRE), що описується формулою:

$$MRE = \frac{|act_i - est_i|}{act_i}, \tag{4.1}$$

де act_i – реальне значення (фактичне);

est_i – прогнозоване значення.

Розрахувати MMRE можна за наступною формулою:

$$MMRE = \frac{1}{n}\Sigma_{i=1}^{n} MRE, \tag{4.2}$$

де n – кількість спостережень.

Обчислення метрики PRED(x) відбувається за формулою:

$$PRED(x) = \frac{1}{n}\Sigma_{i=1}^{n} \begin{cases} 1, & if\ MRE_i \le x \\ 0, & otherwise \end{cases}, \tag{4.3}$$

де *x* – порогове значення, яке визначає допустимий рівень відносної похибки.

Ідеальною вважається модель, яка має низьке значення MMRE та високе значення PRED(x). Ці величини допомагають визначити точність оцінки та стійкість моделі. Точна модель прогнозування трудовитрат повинна мати MMRE≤0.25 (щоб гарантувати, що середня помилка оцінки не перевищує 25%) і PRED(25)≥0.75 (щонайменше 75% прогнозованих значень повинні мати відносну похибку (MRE) менше 25%) [12].

MAE, або середня абсолютна похибка, показує ступінь невідповідності між фактичними та прогнозованими значеннями та розраховується за формулою:

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|act_i - \ est_i| \ , \tag{4.4}$$

MAE – досить популярна метрика, оскільки значення помилки легко інтерпретується, а не конвертується у відсотки або будь-які інші одиниці виміру. Чим ближче MAE до нуля, то точніша модель. Але MAE повертається в тому ж масштабі значень, що й вихідні дані [16].

Нижче продемонстровано порівняльну табл. 4.1, у яку занесені результати точності оригінального UCP та модифікованого методів.

Таблиця 4.1 – Порівняння точності методів оцінки трудовитрат

Метод	MMRE	PRED (25)	MAE
UCP	0,22	63,64%	77,21
Модифікований UCP	0,17	86,36%	60,54

4.2 Аналіз результатів

Аналіз отриманих результатів показав, що модифікований метод UCP є більш точним, ніж оригінальний метод UCP. Це підтверджується значеннями метрик MMRE, PRED (25) і MAE (табл. 4.1).

Для наочної демонстрації результатів дослідження були побудовані два графіки. Перший графік, представлений нижче (рис. 4.1), є гістограмою значень середньої абсолютної похибки (MAE) для двох методів оцінки трудовитрат: оригінального методу UCP та модифікованого методу UCP (позначається як UCP_2).

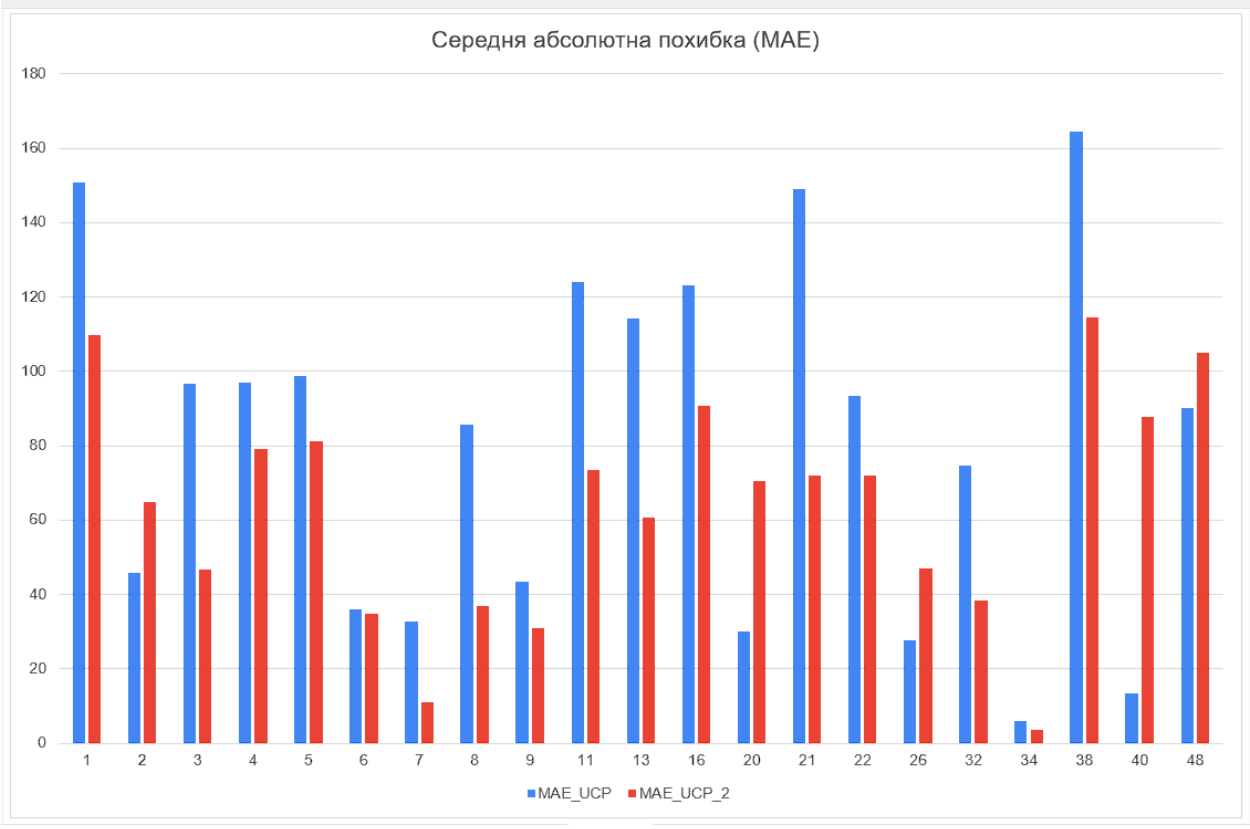

Рисунок 4.1 – Гістограмма значень MAE

Побудована гістограма ясно демонструє, що модифікований метод UCP (червоні стовпчики) забезпечує більш точні передбачення трудовитрат порівняно з оригінальним методом UCP (сині стовпчики). У більшості

випадків, модифікований метод UCP має меншу середню абсолютну похибку (MAE), що підтверджує його вищу ефективність і надійність у прогнозуванні трудовитрат.

На рис. 4.2 представлений графік «Аналіз точності передбачень методами UCP і модифікованого UCP», який є комбінацією діаграми розсіяння та лінійного графіка. Цей рисунок демонструє, що модифікований метод є більш точним, оскільки точки, що відповідають прогнозам оригінального методу, мають більший розкид відносно реальних значень. Це вказує на менш точні передбачення оригінального методу порівняно з модифікованим методом UCP.

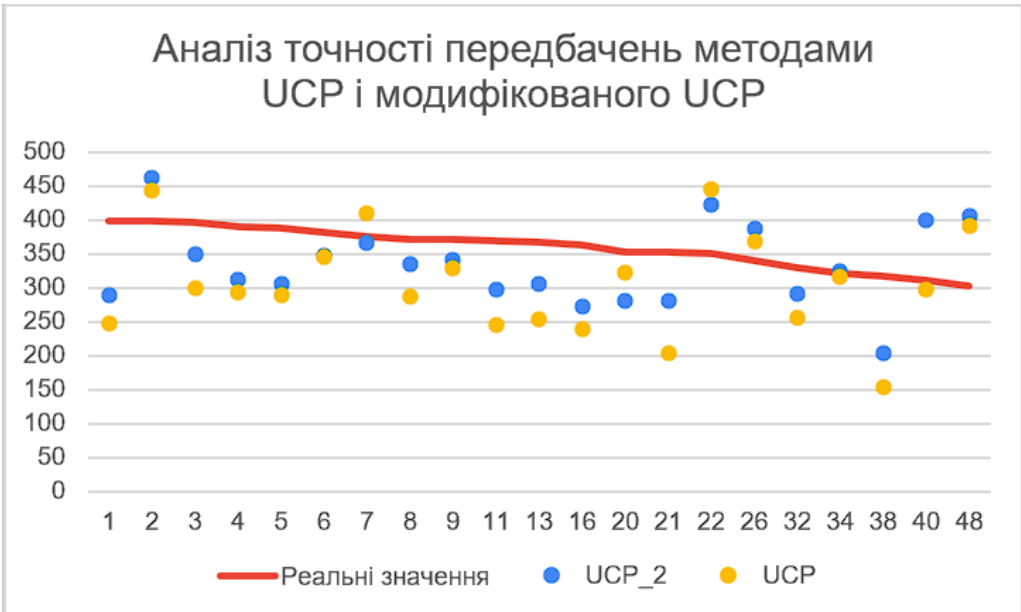

Рисунок 4.2 – Комбінована діаграма розсіяння та лінійного графіка

На основі отриманих результатів, можна зробити висновок, що модифікований метод UCP дає більш точні результати прогнозу трудовитрат, що необхідні для успішного виконання проєктів.

ВИСНОВКИ

У ході виконання цієї кваліфікаційної роботи метод Use Case Points, який використовується для оцінки трудовитрат, був удосконалений за допомогою використання ідей методу PERT.

У першому розділі «Аналіз предметної області та постановка задачі дослідження» був проведений огляд та загальний аналіз проблеми, яка розглядається в роботі, проаналізовані існуючи методи оцінки трудовитрат та здійснена постановка задачі, яка включала в себе мету та опис основних завдань для її досягнення.

У другому розділі «Розробка модифікованого методу оцінки трудовитрат при реалізації IT-проєкту» на основі попередніх досліджень були описані базові засади методу Use Case Points та PERT, а також модифікований метод та алгоритм його реалізації із побудовою діаграм DFD.

Третій розділ «Експериментальна перевірка можливостей використання розробленого методу оцінювання трудовитрат для IT-проєкту» присвячений демонстрації використання модифікованого методу на прототипі реального проєкту.

У четвертому розділі «Експериментальне моделювання та аналіз результатів дослідження» демонструються результати тестування модифікованого методу на основі датасету, який містить інформацію щодо проєктів, трудовитрати яких були розраховані за допомогою оригінального методу Use Case Points та його модифікації і здійснено порівняння результатів розрахунків обома методами.

На основі отриманих результатів, можна зробити висновок, що модифікований метод UCP може бути використаний для підвищення точності прогнозування трудовитрат в майбутніх проектах. Зокрема, збільшення точності на 22% за показником PRED(25) може допомогти компаніям більш точно керувати проектами.

Однак, слід врахувати деякі обмеження цього дослідження. По-перше,

результати базуються на обмеженому наборі даних, що складається з 71 проектів, з яких для тестування було вибрано лише 21. По-друге, розрахунки оцінки факторів TCF та ECF були виконані вручну, що може вплинути на точність результатів. Тому, хоча отримані результати є обнадійливими, для підтвердження цих висновків може бути необхідне додаткове дослідження з використанням більшого та повнішого набору даних.

Ця робота була виконана відповідно до вимог методичних вказівок [17]. При оформленні даного звіту було використано державні стандарти України [18] – [19].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Новохацька Д.В. Особливості та проблеми реалізації ІТ-проєктів в Україні. Вісник Черкаського державного технологічного університету. 2016. С.72-77. DOI: https://doi.org/10.24025/2306-4412.2.2016.82990.

2. Іваненко О.В. Дослідження моделей і методів оцінки трудовитрат при реалізації ІТ-проєктів / О. В. Іваненко ; наук. керівник д.т.н., проф. Петров К.Е. // Радіоелектроніка та молодь у XXI столітті : матеріали 27-го Міжнар. молодіж. форуму, 10–12 травня 2023 р. – Харків : ХНУРЕ, 2023. – Т. 6, ч.1. – С. 188–189.

3. Ghafory H., Sahnosh F.A. The review of software cost estimation model: SLIM. Journal of Advanced Academic Research. 2020. C. 511-515. DOI: 10.33545/27068919.2020.v2.i4h.447.

4. Pillai K., Nair V.S. Sukumaran. A Model for Software Development Effort and Cost Estimation. IEEE Transactions on Software Engineering. 1997. Vol. 23. P. 485-497. DOI: 10.1109/32.624305.

5. Chirra S.M.R., Reza H. A survey on Software Cost Estimation Techniques. Journal of Software Engineering and Applications. 2019. P.226-248. DOI: https://doi.org/10.4236/jsea.2019.126014.

6. Rashid J., Nisar M.W., Mahmood T., Rehman A., Arafat S.Y. A study of Software Development Cost Estimation Techniques and Models. Mehran University Research Journal of Engineering and Technology. 2020. Vol. 39. No. 2. P.413-431. DOI: 10.22581/muet1982.2002.18.

7. Tripathi R., Ra Dr. P. K. Comparative Study of Software Cost Estimation Techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 2016. Vol. 6. P. 323−328.

8. Sinhal A., Bhupendra V. Software Development Effort Estimation: A Review. International Journal of Advanced Research in Computer Science and Software Engineering. 2013. Vol. 3. P.1120-1135.

9. Gandomani T. J.., Koh T.W., Binhamid A. A Case Study Research on

Software Cost Estimation Using Experts'. Estimates, Wideband Delphi, and Planning Poker Technique. International Journal of Software Engineering and Its Applications. 2014. Vol. 8, No. 11. P. 173-182.

10. What Is a Use Case? [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу: https://www.wrike.com/blog/what-is-a-use-case/. – 19.05.2024.

11. What is a use case? How to write one, examples, + template. [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу: https://www.figma.com/resource-library/what-is-a-use-case/. 19.05.2024.

12. Nhung H.L.T.K., Hoc H.T., Hai V.V. An Evaluation of Technical and Environmental Complexity Factors for Improving Use Case. Software Engineering Perspectives in Intelligent Systems. 2020. P.757-768. DOI: 10.1007/978-3-030-63322-6_64

13. Estimating With Use Case Points. [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу: https://www.mountaingoatsoftware.com/articles/estimating-with-use-case-points. 21.05.2024.

14. PERT: Definition, PERT Formula, PERT Chart, Technique & Example. [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу: https://pmstudycircle.com/pert-program-evaluation-and-review-technique/. 22.05.2024.

15. Program Evaluation and Review Technique (PERT) Analysis. [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу: https://acqnotes.com/acqnote/tasks/pert-analysis. 22.05.2024.

16. A Comprehensive Introduction to Evaluating Regression Models. [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу: https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/. 29.05.2024.

17. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проектами в галузі інформаційних технологій» /

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.

18. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. Чинний від 22.06.2015. Київ: ДП «УкрНДНЦ», 2016. 31 с.

19. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складання. Чинний від 04.03.2016. Київ: ДП «УкрНДНЦ», 2016. 20 с.