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 та методів оцінки трудовитрат на забезпечення якості IT проєктів_____
(тема)

Виконав:
студент 2 курсу, групи УПГІТм-22-2
Слєпцов Вадим Анатолійович
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Управління проектами в галузі інформаційних технологій
(повна назва освітньої програми)

Керівник професор кафедри ІУС
Ірина ПАНФЬОРОВА
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри _____ 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 проектами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«01» квітня 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Слєпцову Вадиму Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей та методів оцінки трудовитрат на забезпечення якості ІТ проєктів
затверджена наказом університету від 01 квітня 2024 р. № 258Cm
2. Термін подання студентом роботи до екзаменаційної комісії 06.06.2024 р.
3. Вихідні дані до роботи: науково-технічні публікації; джерела інтернету; науково-технічна література, що стосуються теми кваліфікаційної роботи
4. Перелік питань, що потрібно опрацювати в роботі: аналіз існуючих моделей та методів оцінки трудовитрат на забезпечення якості ІТ-проєктів, модифікація методу оцінки трудовитрат, розробка методики для модифікованого методу оцінки трудовитрат на забезпечення якості ІТ-проєктів, апробація оригінального та модифікованого методу оцінки трудовитрат на забезпечення якості ІТ-проєктів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання	Примітка
1	Дослідження і аналіз процесу забезпечення якості ІТ-проєктів	01.04.2024	виконано
2	Аналіз методів оцінки трудовитрат на забезпечення якості ІТ-проєктів	10.04.2024	виконано
3	Аналіз моделей оцінки трудовитрат на забезпечення якості ІТ-проєктів	15.04.2024	виконано
4	Модифікація методу точок використання для оцінки трудовитрат на забезпечення якості ІТ-проєктів	20.04.2024	виконано
5	Розробка методики для модифікованого методу точок використання для оцінки трудовитрат на забезпечення	05.05.2024	виконано
6	Практична апробація результатів кваліфікаційної роботи	10.05.2024	виконано
7	Оформлення пояснювальної записки та графічного матеріалу	22.05.2024	виконано
8	Підготовка презентації	26.05.2024	виконано
9	Захист кваліфікаційної роботи	10.06.2024	виконано

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. каф. ІУС Ірина ПАНФЬОРОВА
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 88 сторінок, 2 рисунки, 26 таблиць, 1 додаток, 24 джерела.

ВЕБ-ДОДАТОК, ЗАБЕЗПЕЧЕННЯ ЯКОСТІ, ІТ-ПРОЄКТ, МЕТОД, ОЦІНКА ТРУДОВИТРАТ, ТЕСТУВАННЯ, ТРУДОВИТРАТИ.

Кваліфікаційна робота спрямована на дослідження існуючих моделей та методів оцінки трудовитрат на забезпечення якості ІТ-проєктів.

Об’єктом дослідження кваліфікаційної роботи є процес оцінки трудовитрат на забезпечення якості ІТ-проєктів.

Метою роботи є аналіз сучасного стану процесу оцінки трудовитрат на забезпечення якості ІТ-проєктів, дослідження існуючих моделей та методів оцінки трудовитрат, визначення переваг та недоліків досліджуваних моделей та методів, модифікація методу оцінки трудовитрат.

Практична значимість роботи полягає у збільшені точності оцінки трудовитрат на забезпечення якості ІТ-проєктів шляхом модифікації методу точок використання.

В результаті роботи було модифіковано метод точок використання для оцінки трудовитрат необхідних для виконання задач по тестуванню, які враховують значимість факторів технічної складності і складності середовища, складність ІТ-проєкту та особливості конкретної команди забезпечення якості.

Модифікований метод точок використання можна використовувати для оцінки трудовитрат, необхідних на виконання задач по тестуванню. Отримані результати можна використовувати для подальшого покращення методу для отримання більш точних результатів оцінки трудовитрат.

ABSTRACT

The explanatory note to the qualification work: 88 pages, 2 figures, 26 tables, 1 appendix, 24 sources.

IT PROJECT, LABOR COSTS, LABOR ESTIMATION, METHOD, QUALITY ASSURANCE, TESTING, WEB APPLICATION.

The qualification work is aimed at studying existing models and methods for estimating labor costs for quality assurance of IT projects.

The object of research of the qualification work is the process of estimating labor costs for quality assurance of IT projects.

The purpose of the work is to analyze the current state of the process of estimating labor costs for quality assurance of IT projects, to study existing models and methods for estimating labor costs, to determine the advantages and disadvantages of the models and methods under study, to modify the method of estimating labor costs.

The practical significance of the work is to increase the accuracy of estimating labor costs for quality assurance of IT projects by modifying the use case points method.

As a result of the work, the use case points method was modified to estimate the labor required to perform testing tasks, which consider the importance of technical and environmental complexity factors, project complexity, and the characteristics of a particular quality assurance team.

The modified use case points method can be used to estimate the labor required to perform testing tasks. The results can be used to further modify the method to obtain more accurate labor estimates.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – Application Programming Interface

AUCP – Adjusted Use Case Points

AW – Actor Weight

CF – Conversion Factor

ECF – Environmental Complexity Factor

ICF – Initial Conversion Factor

PERT – Project Evaluation and Review Technique

RUP – Rational Unified Process

SDLC – Software Development Life Cycle

TCF – Technical Complexity Factor

TEF – Technical and Environmental Factors

UCP – Use Case Points

UCW – Use Case Weight

UML – Unified Modeling Language

UUCP – Unadjusted Use Case Points

ВСТУП

В сучасному світі, де швидкість, ефективність та якість є ключовими факторами конкурентоспроможності, керування ІТ-проєктами стає однією з найважливіших складових стратегічного успіху підприємств. В контексті цього, ефективне керування трудовитратами є критичною складовою, що визначає якість та результативність ІТ-проєктів.

В свою чергу, забезпечення якості ІТ-проєкта є одним із ключових аспектів успішної реалізації проєктних завдань. Задоволеність замовників визначає успіх нового продукту, і лише продукти високої цінності задовольняють потреби клієнтів, які очікують належної роботи продукту протягом усього життєвого циклу. В сучасному світі кожного дня створюються нові ІТ-продукти, такі як мобільні застосунки, веб-сайти, соцмережі, середовища для програмування, антивіруси тощо, та тільки половина з них стає по-справжньому успішними. Успішність продукту напряму залежить від декількох критеріїв, які були досягнуті в ході виконання ІТ-проєкту, а саме відповідність поставленим вимогам, вчасна дата вводу продукту в експлуатацію, не перевищення обговореного бюджету та якість кінцевого продукту.

Оцінка трудовитрат на забезпечення якості ІТ-проєкту дозволяє зрозуміти, скільки ресурсів, включаючи час та кошти, буде необхідно витратити на досягнення вимог щодо якості. Ефективне забезпечення якості ІТ-проєкту передбачає не тільки визначення вимог та стандартів якості, але й раціональну оцінку трудовитрат, які необхідні для досягнення цих вимог. Вірна оцінка трудовитрат дозволяє побудувати реалістичний графік робіт, розподілити ресурси ефективно та забезпечити вчасне виконання ІТ-проєкту з відповідною якістю.

Однак, в динамічному середовищі сучасного бізнесу вимоги до оцінки трудовитрат зростають і стають більш складними. Виникає необхідність в

розробці та застосуванні нових моделей та методів оцінки трудовитрат, які враховують специфіку різноманітних ІТ-проєктів, їх складність, терміни виконання та особливості команд робітників.

Потреба в систематичному дослідженні моделей та методів оцінки трудовитрат на забезпечення якості ІТ-проєктів стає більш актуальною в умовах постійних змін у технологіях, методах роботи та вимогах споживачів. Розробка ефективних інструментів оцінки трудовитрат дозволить підприємствам оптимізувати витрати, планувати ресурси та забезпечити успішну реалізацію ІТ-проєктів, що відповідає стратегічним цілям організації.

У цьому контексті дане дослідження має на меті розглянути існуючі моделі та методи оцінки трудовитрат, проаналізувати їх ефективність та надати пропозиції щодо подальшої модифікації. Дослідження спрямоване на вирішення актуальних проблем керування ІТ-проєктами та надання практичних рекомендацій для підвищення ефективності процесу оцінки трудовитрат на забезпечення якості ІТ-проєктів. Його результати можуть бути корисними як для академічного світу, так і для практиків у сфері управління ІТ-проєктами, сприяючи подальшому розвитку та вдосконаленню цієї важливої галузі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Дослідження і аналіз процесу забезпечення якості ІТ-проєктів

Одним із ключових елементів конкурентоспроможності є якість. Якість – це ступінь відповідності системи, компоненту або процесу заданим вимогам, потребам або очікуванням користувача. З метою визначення добротності системи, компоненту або процесу використовують атрибути якості – характеристики, що відображають дану властивість [1].

Керування якістю є важливою складовою керування ІТ-проєктом разом з іншими процесами. Зростання та постійне вдосконалення продуктивності ІТ-проєкту значною мірою залежить від того, як забезпечити належне керування якістю. Якість керування проєктом стосується не лише часу та бюджету, але й вимог до специфікації та якості. Завданням процесу керування якістю є забезпечення якості, яку повинна мати робота. Керування якістю вирішує завдання здійснення процесів керування якістю впродовж усього ІТ-проєкту. У рамках процесу керування якістю вимоги до якості, встановлені в рамках процесу планування якості, перетворюються на інструменти тестування та оцінювання, які надалі застосовують під час процесу контролю якості для перевірки дотримання в ІТ-проєкті зазначених вимог до якості. Контроль якості вирішує завдання зіставлення результатів роботи з вимогами до якості з метою забезпечити задоволення результату встановленим вимогам.

Керування якістю ІТ-проєкту складається з чотирьох підкатегорій [1]:

- планування якості ІТ-проєкту;
- забезпечення якості ІТ-проєкту;
- контроль якості ІТ-проєкту;
- вдосконалення процесів забезпечення якості.

На рисунку 3.1 проілюстрована модель підходу до керування якістю в ІТ-проєктах [2].

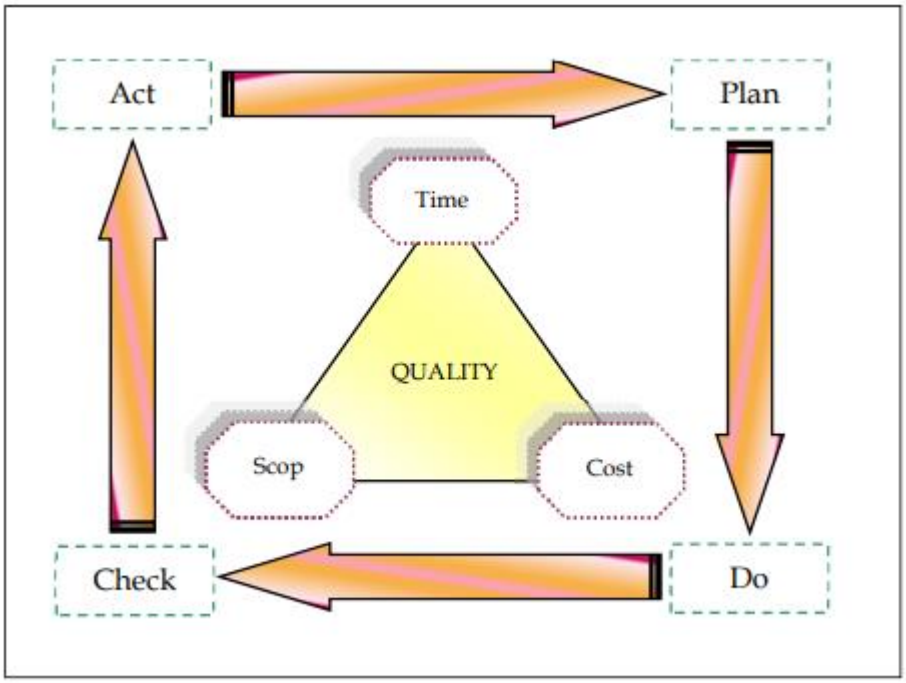

Рисунок 3.1 – Модель керування якістю IT-проєкту

Забезпечення якості IT-проєкту – це комплекс заходів, які визначають і оцінюють адекватність процесів розробки програмного забезпечення для надання доказів, які підтверджують впевненість у тому, що процеси розробки програмного забезпечення є належними і створюють програмні продукти відповідної якості для їхнього використання за призначенням [3]. Ключовим атрибутом забезпечення якості є об'єктивність функції забезпечення якості по відношенню до IT-проєкту. Функція забезпечення якості також може бути організаційно незалежною від IT-проєкту, тобто вільною від технічного, управлінського та фінансового тиску з боку IT-проєкту. Забезпечення якості має два аспекти: забезпечення якості продукту (програмного забезпечення) та забезпечення якості процесу.

Якість програмного забезпечення – це комплекс характеристик програмного продукту, який визначає здатність виконувати покладені на нього функції [1]. Розрізняються поняття внутрішньої якості, пов'язаної з характеристиками програмного забезпечення самого по собі без урахування його поведінки та зовнішньої якості, що характеризує програмне забезпечення

з точки зору його поведінки і якості програмного забезпечення при використанні в різних контекстах – тієї якості, яку відчуває користувач при конкретних сценаріях роботи програмного забезпечення.

Діяльність із забезпечення якості процесів гарантує, що процеси, які використовуються для розробки, впровадження, експлуатації та обслуговування програмного забезпечення, відповідають умовам контрактів, дотримуються всіх встановлених законів, правил і норм, а також є адекватними, ефективними та результативними за своїм призначенням.

Діяльність із забезпечення якості продукту (програмного забезпечення) гарантує надання доказів того, що програмні продукти та пов’язана з ними документація визначені в контрактах і відповідають їм, а також гарантує, що невідповідності виявлені та усунуті.

У рамках забезпечення якості існує також поняття тестування. Терміни забезпечення якості та тестування пов’язані між собою, але не тотожні. Забезпечення якості відповідає за весь процес розробки та інтегрується у всі його етапи: від створення вимог до майбутнього рішення до тестування, релізу продукту та його пострелізного обслуговування. Тестування – це перевірка програмного забезпечення на відповідність вимогам [1]. Таким чином, можна визначити, що забезпечення якості – більш ширше поняття, котре включає в себе роботи з тестування.

Тестування – це не ізольований процес. Це частина моделі життєвого циклу програмного забезпечення. Саме тому вибір засобів і методик тестування буде прямо залежати від обраної моделі розробки.

Життєвий цикл програмного забезпечення (Software Development Life Cycle, SDLC) – це умовна схема, що включає окремі етапи, які представляють стадії процесу створення програмного забезпечення [1].

При цьому на кожному етапі циклу виконуються різні дії. Життєвий цикл програмного забезпечення поділяється на впорядковані стадії, серед яких безпосередньо пов’язаними із процесом розробки програмного забезпечення є наступні:

– аналіз вимог;

– проєктування (попереднє і детальне);

– реалізація;

– тестування.

Моделі життєвого циклу програмного забезпечення описують взаємозв’язки стадій. Найбільш відомі типи моделей життєвого циклу: послідовні та ітераційні. Ці моделі на практиці можуть змішуватись та утворювати змішані моделі життєвого циклу програмного забезпечення. Мета використання моделі життєвого циклу – створити ефективний, економічно вигідний і якісний програмний продукт.

Серед найбільш відомих та часто використовуваних можна виділити наступні моделі життєвого циклу програмного забезпечення:

– класична каскадна модель;

– V-подібна модель;

– ітераційні моделі з приростом життєвого циклу програмного забезпечення;

– спіральна модель.

Незважаючи на існуючі обмеження, класична каскадна модель використовувалася спеціалістами з програмної інженерії протягом тривалого періоду часу. Розуміння її сильних сторін та недоліків покращує оціночний аналіз інших, часто більш ефективних моделей життєвого циклу, базованих на даній моделі. В моделі передбачено, що кожна наступна фаза розпочинається тільки тоді, коли повністю завершене виконання попередньої фази. Кожна фаза каскадної моделі має свої певні критерії входу та виходу: вхідні та вихідні дані. Спроби оптимізації каскадної моделі життєвого циклу програмного забезпечення привели до виникнення інших циклів розробки програмного забезпечення. Прототипування програм дозволяє забезпечити повне розуміння вимог, в той час як інкрементні та спіральні моделі дозволяють

повторно повертатись до фаз, відповідних класичній каскадній моделі, перш ніж отриманий продукт буде визнано завершеним.

Переваги каскадної моделі полягають в розгорнутому документуванні кожного етапу розробки, чіткому плануванню термінів і витрат та абсолютна прозорість усіх процесів розробки для замовника. Головним же недоліком цієї моделі є відсутність гнучкості, бо якщо постає необхідність внесення змін до вимог на пізніших етапах, то відбувається повернення до найпершої стадії і переробка заново всієї виконаної роботи, що призводить до збільшення об'єму витрат коштів та часу.

V-подібна модель життєвого циклу програмного забезпечення була створена з метою допомогти працюючій над IT-проєктом команді в плануванні із забезпеченням подальшої можливості тестування системи. В цій моделі особливе значення приділяється діям, направленим на верифікацію та атестацію продукту. Вона демонструє, що тестування продукту обговорюється, проєктується та планується на ранніх етапах життєвого циклу розробки програмного забезпечення. План випробування прийомки замовником розробляється на етапі планування, а компонентного випробування системи – на фазах аналізу, розробки проєкту тощо. На практиці задачі тестування програмного забезпечення і системного тестування часто об'єднуються в один процес, однак для складних програмних систем тестування технічних характеристик має виконуватись окремо.

Ітераційні моделі можна розділити на два класи: моделі з приростом (incremental) та еволюційні (evolutionary). Згідно з цими моделями програмний продукт розробляється ітераціями і кожна ітерація закінчується випуском працездатної версії програмного продукту. Основна відмінність між моделями – підхід до визначення вимог.

Ітераційна розробка представляє собою процес часткової реалізації системи та повільного нарощування функціональних можливостей. Цей підхід дозволяє зменшити витрати, які витрачаються до моменту досягнення рівня вихідної продуктивності. За допомогою цієї моделі пришвидшується процес

створення функціонуючої системи. Цьому сприяє застосовуваний принцип компонування з стандартних блоків, завдяки якому забезпечується контроль над процесом розробки змінних вимог. Ітераційна модель діє за принципом каскадної моделі розробки з перекриттям, завдяки чому функціональні можливості продукту, які придатні до експлуатації, формуються заздалегідь. Для цього може знадобитись наперед сформований повний набір вимог, котрі виконуються у вигляді послідовних невеликих за розміром ІТ-проєктів, або ж виконання ІТ-проєкту може розпочатись із формування загальних цілей, котрі потім уточняються та реалізуються групами розробників. Ітераційні моделі з приростом життєвого циклу програмного забезпечення широко використовуються для розробки комерційних програмних продуктів, які розвиваються протягом довгого періоду часу або для яких зовнішні вимоги змінюються слабо.

На відміну від моделей з приростом еволюційні моделі застосовуються в тих випадках, коли усі вимоги не можуть бути визначені одразу або відомо, що вони можуть змінитись. Розробка ІТ-проєкту за цими моделями також виконується ітераціями. Але кожна ітерація охоплює усі стадії розробки – від аналізу вибраного набору вимог до випуску версії. На кожній ітерації виконується прототипування вимог і проєкту. До найбільш відомих еволюційних моделей відноситься спіральна модель.

Спіральна модель життєвого циклу вміщує в собі переваги каскадної моделі. При цьому в ній закладені аналіз ризиків, керування ними, а також процеси підтримки та менеджменту. В ній передбачено розробку програмного продукту при використані методу прототипування чи швидкої розробки додатків шляхом застосування мов програмування та засобів четвертого покоління. Модель відображає базову концепцію, котра полягає в тому, що кожний цикл являє собою набір операцій, котрому відповідає така ж кількість стадій, як і в моделі каскадного процесу. При чому приймається до уваги кожна складова частини продукту та кожний рівень складності, починаючи із

загального формулювання потреб та завершуючи кодуванням кожної окремої програми.

Проведення ефективного забезпечення якості ІТ-проєкту потребує точної оцінки трудовитрат для визначення кількості ресурсів, потрібних для досягнення визначеного рівня якості. Трудовитрати, необхідні для забезпечення якості, можуть бути різними залежно від галузі та стилів керування проєктом. Наприклад, у таких галузях, як фармацевтика, охорона здоров'я, транспорт і ядерна енергетика, можуть бути встановлені суворіші процедури контролю якості порівняно з іншими галузями, а трудовитрати, необхідні для дотримання відповідних стандартів, можуть бути значними. У гнучких ІТ-проєктах операція контролю якості може виконуватися всіма членами команди протягом усього життєвого циклу ІТ-проєкту. В свою чергу, під час здійснення ІТ-проєктів на основі каскадної моделі операції контролю якості здійснюються спеціально призначеними членами команди у встановлені моменти часу на кінцевій стадії ІТ-проєкту або фази.

Оцінка трудовитрат – це процес оцінки ресурсів команди, необхідних для виконання робіт. Факторами, які враховуються під час визначення трудовитрат на керування якістю ІТ-проєкту, можуть бути вартість, розмір, графік, людські ресурси та складність.

Оцінка трудовитрат на забезпечення якості включає аналіз різних факторів, що впливають на якість ІТ-проєкту – від кваліфікації команди до специфікацій ІТ-проєкту та технологічних рішень. Рекомендується проводити оцінку регулярно для вчасного виявлення та виправлення погрішностей, які можуть з'являтися в ході реалізації ІТ-проєкту. Для проведення оцінки трудовитрат існують моделі та методи, які були сформовані на основі даних, отриманих при реалізації значної кількості ІТ-проєктів.

1.2 Аналіз методів оцінки трудовитрат на забезпечення якості ІТ-проєктів

У сучасному світі, де бізнес, наука та техніка безперервно розвиваються та еволюціонують, забезпечення якості ІТ-проєктів грає все більш значущу роль у досягненні успіху організацій. Один з ключових аспектів ефективного забезпечення якості полягає у вмінні оцінити та забезпечити правильний розподіл ресурсів для досягнення загальної продуктивності та успіху ІТ-проєкту. У цьому контексті методи оцінки трудовитрат відіграють важливу роль у плануванні та контролі процесів забезпечення якості.

Методи оцінки трудовитрат використовуються для визначення того, скільки часу, зусиль і ресурсів знадобиться для забезпечення якості ІТ-проєкту. До методів оцінки трудовитрат належать наступні методи: метод оцінки за Паркінсоном, метод експертної оцінки, метод оцінки за аналогією, метод точок використання, метод оцінки за трьома точками та метод Bottom-Up.

Метод оцінки трудовитрат за Паркінсоном, також відомий як метод емпіричних графіків або шаблонів Паркінсона, був розроблений економістом Сирілом Норткотом Паркінсоном, відомим також як автор закону Паркінсона: робота розтягується, щоб заповнити весь доступний час [4].

Метод оцінки трудовитрат за Паркінсоном використовує візуальний підхід та спирається на емпіричні спостереження з метою прогнозування трудовитрат для виконання робіт і завдань.

Основні етапи застосування методу оцінки трудовитрат за Паркінсоном:

– розбиття ІТ-проєкту на окремі завдання;

– розробка емпіричних графіків або шаблонів на основі досвіду і статистичних даних, які відображають залежність між потребами в трудових ресурсах та ступенем складності окремих завдань ІТ-проєкту;

– аналіз завдань ІТ-проєкту на предмет їх складності, коригування оптимальної кількості потрібних робочих, враховуючи різні параметри, такі як рівень навичок, кваліфікація, досвід;

– визначення тривалості кожного конкретного завдання на основі даних емпіричних графіків;

– обчислення загальної кількості трудових ресурсів і тривалості виконання всіх завдань і ІТ-проєкту в цілому.

Переваги методу оцінки трудовитрат за Паркінсоном полягають в обчисленнях, які засновані на емпіричних даних, накопичених з досвіду і знань, та які спираються на конкретні практичні результати. Також графічні зображення допомагають наочно ілюструвати залежність між ресурсами та складністю завдань.

Недоліками методу, в свою чергу, є те, що метод вважається досить старим та його актуальність може бути обмеженою в контексті сучасних умов праці. Він може бути слабко пристосованим до нетипових ІТ-проєктів, а методика розробки емпіричних графіків може бути складною і тривалою. Також метод передбачає значну участь експертів потрібної галузі для розробки якісних емпіричних графіків.

Отже, метод оцінки трудовитрат за Паркінсоном є корисним і зручним інструментом для оцінки ресурсів і тривалості ІТ-проєктів, проте його точність і актуальність залежить від адекватності використовуваних емпіричних графіків та шаблонів.

Метод експертної оцінки використовується у процесі забезпечення якості ІТ-проєктів для прийняття обґрунтованих рішень проєктного менеджменту. Метод полягає в зборі, аналізі і об’єднанні думок декількох експертів з метою оцінки різних параметрів процесу, включаючи трудові ресурси, ризики, реалізацію та ефективність рішень.

Основні етапи методу експертної оцінки включають [4]:

– добір спеціалістів з представників різних дисциплін, з відповідним досвідом і знаннями для оцінки параметрів;

– визначення параметрів процесу, ваги кожного з них, представлення експертам загальних показників якості ІТ-проєкту та критеріїв оцінки відповідних параметрів;

– забезпечення анонімності для зменшення відхилення у відповідях та запобігання груповому мисленню;

– збір оцінок експертів щодо представлених показників якості ІТ-проєкту;

– групування, аналіз та узагальнення отриманих даних; зазвичай це передбачає застосування статистичних методів, таких як середнє значення та формування висновків;

– використання результатів експертної оцінки для підтримки проєктного менеджменту та оптимізації показників якості ІТ-проєкту.

Переваги методу експертної оцінки базуються на знаннях і досвіді експертів, що дозволяє отримати більш точні та об’єктивні оцінки. Метод безпосередньо враховує специфіку ІТ-проєктів та їхню технічну особливість. Метод експертної оцінки можливо використовувати у комбінації з іншими методами аналізу та забезпечення якості.

Недоліки методу експертної оцінки включають:

– якість оцінки залежить від відбору експертів та варіативності оцінок;

– часові та фінансові витрати на організацію опитувань, інтерв’ю та аналіз результатів;

– відсутність точного алгоритму для опрацювання отриманих даних та формулювання рекомендацій.

Застосування методу експертної оцінки для оцінки трудовитрат забезпечення якості ІТ-проєктів допомагає оцінити ризики, керувати ресурсами та контролювати виконання процесу. Це є важливою складовою успішного проєктного менеджменту, що дозволяє вчасно реагувати на можливі проблеми та виконати ІТ-проєкт відповідно до своїх цілей та завдань.

Метод оцінки трудовитрат за аналогією, також відомий як метод порівняння або історичний, використовується в забезпеченні якості ІТ-проєктів для визначення трудовитрат, часу та ресурсів на основі аналізу даних аналогічних завершених ІТ-проєктів. Цей метод ґрунтується на ідейному зрізі, що якщо попередній досвід у вирішенні схожих задач був успішним, то й підхід до нового ІТ-проєкту за допомогою аналогічних методів може бути таким же ефективним. Оскільки керівник процесу базує свою оцінку на порівнянні, чим більше даних він має в своєму розпорядженні, тим точнішою буде його оцінка. Цей метод оцінки простий у розумінні та виконанні. Менеджери ІТ-проєктів часто використовують його, коли у них мало деталей про поточний ІТ-проєкт [5].

Оцінка за аналогією – це метод оцінки за принципом «зверху вниз», спочатку отримується оцінка вартості всього ІТ-проєкту, а потім вона розбивається на багато менших елементів ІТ-проєкту. Метод може бути використаний для оцінки вартості і тривалості всього ІТ-проєкту або тільки окремих завдань.

Основні етапи методу оцінки трудовитрат за аналогією:

– пошук та аналіз завершених ІТ-проєктів, які мають схожі технічні аспекти, технології, розміри, функціональність та вимоги до якості;

– перегляд завершених ІТ-проєктів, збір показників трудовитрат, часу, ресурсів та своєчасності виконання задач;

– дослідження ступені схожості між даними ІТ-проєкту та проєктами-аналогами в плані структури, об’ємів задач, процесів роботи, якості та технологій;

– проведення нормалізації даних для аналогічних ІТ-проєктів, враховуючи особливості та відмінності між ними;

– на основі зібраних даних, проводиться розрахунок трудовитрат нового ІТ-проєкту з урахуванням проаналізованих спостережень та закономірностей;

– проведення аналізу можливих ризиків та врахування їх при корекції оцінок.

Цей інструмент оцінки трудовитрат може швидко надати базову оцінку, використовуючи мінімальну кількість параметрів. При цьому оцінки засновані на реальних даних та практичному досвіді, що може зменшити ймовірність похибок. Також оцінка за аналогією заощаджує час та ресурси, оскільки вже існують дані від яких можна відштовхуватися. Але при використанні цього методу треба пам’ятати про його недоліки, а саме: ризик отримання неточних результатів через значні відмінності між ІТ-проєктом та аналогами, ймовірні проблеми з пошуком аналогічних ІТ-проєктів та висока залежність від зібраних даних та досвіду спеціалістів, які робили аналіз.

Для отримання найкращого результату варто поєднувати метод оцінки трудовитрат за аналогією з іншими методами оцінки, щоб розширити аналіз і підійти до визначення ресурсів ІТ-проєкту всебічно.

Метод точок використання (Use Case Points, UCP) є одним з методів оцінки трудовитрат на тестування програмного забезпечення. Він базується на аналізі використання системи, що вимагає детального розуміння бізнес-процесів, які система повинна підтримувати [6].

Суть методу полягає в наступному: визначаються всі варіанти використання системи; кожному варіанту присвоюється певна кількість балів в залежності від його складності; сума всіх балів дає загальну оцінку трудовитрат на тестування системи.

Варіант використання (Use Case) – це серія пов’язаних взаємодій між користувачем і системою, які дозволяють користувачеві досягти поставленої мети. Варіанти використання є одним із способів зафіксувати функціональні вимоги до системи. Користувач системи називається «Актором». Варіанти використання в основному мають текстову форму.

Оцінка за допомогою UCP вимагає, щоб всі варіанти використання були написані з певною метою і приблизно на одному рівні, з однаковим ступенем деталізації. Отже, перед оцінкою команда ІТ-проєкту повинна переконатися,

що вона написала варіанти використання з визначеними цілями і на детальному рівні. Варіант використання зазвичай завершується протягом одного сеансу, і після досягнення мети користувач може перейти до іншої діяльності.

Метод точок використання набуває все більшої популярності для оцінки трудомісткості програмного забезпечення. Цей метод може бути дуже корисним у випадку тендерних ІТ-проєктів, оскільки варіанти використання є однією з перших, а іноді і єдиною інформацією, доступною на початку програмного ІТ-проєкту.

Метод оцінки трудовитрат за трьома точками – це статистичний підхід до оцінки трудовитрат, тривалості та ресурсів для ІТ-проєктів, який враховує найкращі, найгірші та найбільш ймовірні сценарії для кожної задачі. Метод забезпечує більш точний та закономірний результат оцінки, зменшуючи вплив невизначеності та ризику на результати оцінки. Оптимістична оцінка – це значення, яке відображає найкращий сценарій, який можна отримати, якщо при забезпеченні якості ІТ-проєкта та його реалізації не виникне жодних проблем. Песимістична оцінка – це значення відображає результати, якщо в ході реалізації ІТ-проєкту виникнуть несприятливі умови. Реалістична оцінка – це третій тип оцінки, при якій реалізація ІТ-проєкту проходить з певними проблемами, але здебільшого все йде гладко. Вона враховує непередбачувані обставини, які є ймовірними протягом усього ІТ-проєкту [7].

Однією з ключових переваг оцінки трудовитрат за трьома точками є те, що враховується ризик, пов’язаний з ІТ-проєктом. Вона також враховує думку членів команди при формуванні оцінок, що може підвищити рівень залученості. При використанні методу оцінки трудовитрат за трьома точками менеджери ІТ-проєктів пропонують усім членам команди ІТ-проєкту написати свої оптимістичні, реалістичні та песимістичні оцінки щодо роботи, яка від них вимагається, та часу, який вони очікують на завершення ІТ-проєкту. Потім менеджери можуть взяти накопичені підсумки і застосувати оцінки за трьома точками [7]:

$$\text{Оцінка} = (O + 4 * R + P) / 6, \quad (1.1)$$

де О – оптимістична оцінка, яка має вагу 1;

R – реалістична оцінка, яка має вагу 4;

Р – песимістична оцінка, яка має вагу 1.

Реалістичній оцінці надається більша вага, ніж іншим, що дозволяє отримати оцінку, яка враховує найкращий і найгірший сценарії, не стаючи при цьому нереалістичною.

Оцінку за трьома точками можна використовувати для планування задач. Наприклад, менеджер може використовувати формулу стандартного відхилення з методом оцінки та аналізу проєкту (PERT), щоб оцінити час, який можуть зайняти окремі задачі. PERT-діаграма допомагає впорядкувати задачі ІТ-проєкту з метою оцінки тривалості кожного з них. Це допомагає передбачити більш точний час початку і завершення конкретних задач. Планування задач особливо важливе для складних ІТ-проєктів, де дати початку і завершення можуть бути нечіткими.

Оскільки оцінка за трьома точками дає більш точну оцінку, вона допомагає зменшити ризик невдачі та знижує ймовірність надмірно оптимістичної або завищеної оцінки. Після того, як менеджери визначили ризики, вони можуть проаналізувати їх і визначити пріоритети, застосувавши метод оцінки за трьома точками. Оптимістична оцінка передбачає відсутність ризиків, тоді як реалістична оцінка визнає наявність помірних ризиків. Песимістична оцінка означає, що існує висока ймовірність виникнення ризику.

Як і інші методи оцінки трудовитрат, оцінка за трьома точками допомагає визначити тривалість робіт по ІТ-проєкту. Вона забезпечує реалістичну оцінку часових рамок ІТ-проєкту, враховуючи відповідні змінні ІТ-проєкту. Менеджери зможуть краще спланувати ІТ-проєкт та роботи по ньому, якщо матимуть приблизне уявлення про те, скільки часу він може

зайняти, що допоможе приймати більш обґрунтовані рішення, наприклад, орендувати додаткове обладнання або найняти більше робітників.

Отже, метод оцінки трудовитрат за трьома точками є цінним інструментом для точного прогнозування трудовитрат та часу, необхідних для забезпечення якості ІТ-проєктів. Він допомагає краще керувати ІТ-проєктами й контролювати процес забезпечення якості, дозволяючи вчасно реагувати на можливі ризики та зміни.

Метод оцінки трудовитрат Bottom-Up (знизу вгору) – це підхід до оцінки трудовитрат, часу та ресурсів на основі детального розгляду всіх складових ІТ-проєктів. Він базується на принципі поетапного об’єднання вартості, часу та ресурсів окремих фрагментів ІТ-проєкту до рівня пріоритетів, процесів, функціональних модулів та остаточного ІТ-проєкту [8].

Оцінка «знизу вгору» важлива, тому що вона може дати бізнесу, командам і менеджерам ІТ-проєктів уявлення про те, скільки може коштувати завершення ІТ-проєкту. Охоплюючи всі відповідні компоненти, менеджери ІТ-проєктів можуть отримати точні та вичерпні оцінки. Будь-хто, хто бере участь у ІТ-проєкті з декількома компонентами, може використовувати метод оцінювання Bottom-Up. Це може бути актуальною стратегією для багатьох галузей і ситуацій. Як правило, оцінювачі – це люди, які безпосередньо залучені до проєктних команд і мають практичні знання про обсяг ІТ-проєкту. Це може допомогти їм зрозуміти всі компоненти, задіяні в ІТ-проєкті, щоб переконатися, що вони точно оцінюють загальну вартість та тривалість.

Bottom-Up оцінка відрізняється від іншої популярної методики оцінки, яка називається «зверху вниз», тим, що оцінка «знизу вгору» включає всі пов’язані з ІТ-проєктом витрати і критерії. При оцінюванні «зверху вниз» замість того, щоб починати з різних компонентів ІТ-проєкту, менеджери починають з кінцевої мети. Bottom-Up підхід починається з окремих елементів і об’єднує їх разом. При підході «зверху вниз» отримати точну оцінку може бути складніше. Оскільки оцінка Bottom-Up передбачає детальний аналіз, вона часто може призвести до кращого загального розуміння обсягу ІТ-проєкту. Це

може полегшити обґрунтування результатів і покладання на прогнози. Для оцінки часу, однак, може бути корисною оцінка «зверху вниз» або оцінка за аналогією, оскільки оцінювачі можуть використовувати історичні дані з попередніх ІТ-проєктів для прогнозування тривалості ІТ-проєкту.

Метод оцінки трудовитрат Bottom-Up є ефективним й детальним інструментом для планування ресурсів, часу та бюджету в ІТ-проєктах. Він дає можливість ретельно проаналізувати вклад кожної задачі та компонента в загальний успіх ІТ-проєкту. Однак, цей метод має високу трудомісткість та залежить від досвіду фахівців.

Отже, багато методів оцінки трудовитрат потребують залучення спеціалістів, які мають досвід в реалізації ІТ-проєктів та проведені процесу забезпечення якості, для отримання найбільш точної оцінки. При використані подібних методів завжди існує вірогідність суб’єктивізму, що може призвести до отримання невірних оцінок трудовитрат.

1.3 Аналіз моделей оцінки трудовитрат на забезпечення якості ІТ-проєктів

Моделі оцінки трудовитрат являють собою математичні моделі, які використовуються для кількісного прогнозування трудовитрат на основі різних параметрів і факторів ІТ-проєкту. Серед таких моделей найчастіше використовуються і є добре задокументованими модель Putnam SLIM та модель COCOMO.

Модель Putnam SLIM (Software Life Cycle Management) – це математична модель для прогнозування трудовитрат, тривалості й витрат ресурсів розробки програмного забезпечення. Модель базується на принципі системної абстракції та спрощення процесів розробки програмного

забезпечення та співвідношенні різних параметрів розробки для визначення точних оцінок та прогнозів [9].

Створена Лоуренсом Патнемом-старшим модель Патнема описує час і зусилля, необхідні для завершення програмного ІТ-проєкту заданого розміру. Це одна з найбільш ранніх моделей такого типу, що була розроблена і є однією з найпоширеніших. Тісно пов'язаними з програмним забезпеченням параметричними моделями є конструктивна модель вартості (COCOMO) та параметричний аналіз інформації для визначення вартості та оцінки програмного забезпечення (PRICE-S).

Однією з ключових переваг цієї моделі є простота її калібрування. Більшість організацій, що займаються розробкою програмного забезпечення, незалежно від рівня зрілості можуть легко зібрати дані про розмір, зусилля і тривалість минулих ІТ-проєктів. Продуктивність процесу, будучи експоненціальною за своєю природою, зазвичай перетворюється на лінійний індекс продуктивності, який організація може використовувати для відстеження власних змін у продуктивності та застосовувати для оцінки майбутніх зусиль.

Модель оцінки трудовитрат Putnam SLIM є корисним інструментом для прогнозування тривалості, трудовитрат і ресурсів у ІТ-проєктах. Вона допомагає ефективно об'єднувати різні параметри та керувати процесом забезпечення якості, враховуючи реальний стан ресурсів та розвиток ІТ-проєкту. Однак, для отримання точних оцінок, необхідно багато даних та досвіду, і завжди повинні враховуватися ризики та відповідні корекції оцінок.

Модель COCOMO – це процедурна модель оцінки вартості програмних ІТ-проєктів, яка часто використовується для надійного прогнозування різних параметрів, пов'язаних із створенням ІТ-проєкту, таких як розмір, трудовитрати, вартість, час і якість. Вона була запропонована Баррі Боемом і базується на дослідженні 63 проєктів, що робить її однією з найбільш задокументованих моделей [10].

Ключовими параметрами, які визначають якість будь-якого програмного продукту, і які також є результатом COCOMO, є насамперед трудовитрати та розклад:

– трудовитрати – це кількість праці, яка буде необхідна для виконання завдання, вимірюється в людино-місяцях;

– розклад – це кількість часу, необхідного для завершення роботи, яка пропорційна докладеним трудовитратам, вимірюється в таких одиницях часу, як тижні та місяці.

Для прогнозування оцінки вартості на різних рівнях були запропоновані різні моделі COCOMO залежно від необхідного рівня точності та коректності. Всі ці моделі можуть бути застосовані до різних IT-проєктів, характеристики яких визначають значення константи, що використовується в подальших розрахунках.

Базова модель COCOMO – це найпростіший підхід, який ґрунтується на ідеї про зв’язок між розміром коду програмного забезпечення та кількості використовуваних робочих годин.

Базова модель COCOMO передбачає, що трудомісткість є лише функцією кількості рядків коду та деяких констант, які оцінюються відповідно до різних програмних систем. Однак, в дійсності, жодна система не може бути розрахована виключно на основі кількості рядків коду. Для цього необхідно враховувати інші фактори, наприклад, надійність та досвід команди. Ці фактори відомі як фактори вартості, і проміжна модель COCOMO використовує такі як необхідний рівень надійності програмного забезпечення, розмір бази даних, складність продукту, обмеження продуктивності під час виконання, аналітичні можливості, досвід роботи з віртуальними машинами, досвід роботи з мовами програмування, використання програмних інструментів, застосування методів програмної інженерії, необхідний розклад розробки.

Детальна модель COCOMO включає всі характеристики проміжної версії з оцінкою впливу факторів вартості на кожен крок процесу розробки

програмного забезпечення. Детальна модель використовує різні мультиплікатори зусиль для кожного атрибуту фактору вартості. У детальній моделі COCOMO все програмне забезпечення поділяється на різні модулі, проводиться оцінка зусиль для кожного модуля, а потім підсумовуються усі необхідні зусилля.

Переваги моделі COCOMO:

– забезпечує систематичний спосіб оцінки вартості та зусиль програмного ІТ-проєкту;

– може бути використана для оцінки вартості та трудовитрат програмного ІТ-проєкту на різних етапах процесу розробки;

– допомагає визначити фактори, які мають найбільший вплив на вартість і трудовитрати програмного ІТ-проєкту;

– може використовуватися для оцінки здійсненності програмного ІТ-проєкту шляхом оцінки вартості та зусиль, необхідних для його завершення.

Недоліки моделі COCOMO:

– припускає, що розмір програмного забезпечення є основним фактором, який визначає вартість та зусилля програмного ІТ-проєкту, що не завжди може бути правдою;

– не враховує специфічні характеристики команди розробки, які можуть мати значний вплив на вартість та трудомісткість програмного ІТ-проєкту;

– не дає точної оцінки вартості та трудовитрат програмного ІТ-проєкту, оскільки ґрунтується на припущеннях та середніх значеннях.

Модель оцінки трудовитрат COCOMO є корисним та широко використовуваним інструментом для прогнозування тривалості ІТ-проєкту, витрат ресурсів та вартості ІТ-проєктів. Вона ґрунтується на емпіричному підході до оцінювання ефективності програмних ІТ-проєктів з використанням різноманітних атрибутів та характеристик. Однак, модель має певні недоліки,

такі як зосередження на розмірі програмного коду та історичних даних, які можуть впливати на точність оцінок.

Правильне визначення, розподіл та контроль трудовитрат є важливими аспектами для досягнення ефективності при забезпеченні якості ІТ-проєктів. Моделі та методи оцінки трудовитрат є корисними техніками, необхідними для отримання точної оцінки трудовитрат, що дозволяє збільшити ефективність процесу забезпечення якості і отримати якісний та успішно реалізований ІТ-проєкт.

1.4 Висновки та постановка задачі дослідження

Забезпечення якості ІТ-проєкту є ключовим аспектом успішної реалізації ІТ-проєктів, який відіграє важливу роль у задоволенні потреб клієнтів і формуванні конкурентоспроможності на ринку. В свою чергу, ефективна оцінка трудовитрат є важливим інструментом керування процесами на різних етапах реалізації ІТ-проєкту, що дозволяє раціоналізувати витрати, оптимізувати ресурси і досягти вимог замовника щодо якості продукту.

Сучасні моделі та методи оцінки трудовитрат враховують ряд факторів, які визначають складність і витрати ІТ-проєктів, включаючи технологічні, організаційні та управлінські аспекти. Однак, в процесі застосування моделей та методів можуть виникати проблеми, пов’язані з недостатньою точністю оцінки або неадекватністю застосування у конкретних умовах ІТ-проєкту.

Дослідивши моделі та методи оцінки трудовитрат на забезпечення якості ІТ-проєктів, було виділено існуючі проблеми, які потрібно опрацювати. По-перше, ІТ-проєкти часто є унікальними та неповторними, що ускладнює точне прогнозування трудовитрат. Попередній досвід та історичні дані можуть бути корисними, але не завжди точно відображати реальні затрати на забезпечення якості. Визначення аналогічних ІТ-проєктів, постійний збір, аналіз та

узагальнення інформації може допомогти збільшити точність оцінювання.

По-друге, оцінка трудовитрат сильно залежить від досвіду, навичок та продуктивності команди, яка працює над ІТ-проєктом. Оцінка може виявитися неточною, якщо команда не враховує можливі проблеми з плануванням, комунікацією та розумінням вимог. Інвестування у навички, досвід та професійний розвиток команди може підвищити продуктивність та знизити трудовитрати, а також забезпечити точніше розуміння процесів забезпечення якості.

Наступна проблема пов’язана з моделями життєвого циклу програмного забезпечення. Різні моделі, такі як каскадна, ітераційна та інші, вимагають різного підходу до забезпечення якості та можуть мати свої особливості щодо оцінки трудовитрат. Вибір правильної моделі життєвого циклу для конкретного ІТ-проєкту допоможе спростити забезпечення якості та сприятиме зменшенню невизначеності трудовитрат.

Метою роботи є дослідження та удосконалення існуючих моделей і методів оцінки трудовитрат на забезпечення якості ІТ-проєктів.

Для досягнення мети кваліфікаційної роботи необхідно вирішити наступні задачі дослідження:

– проаналізувати сучасні моделі та методи оцінки трудовитрат на забезпечення якості ІТ-проєктів;

– провести модифікацію методу оцінки трудовитрат на забезпечення якості;

– розробити методику застосування модифікованого методу;

– експериментально перевірити отримані результати оцінки трудовитрат на забезпечення якості при використанні модифікованого методу.

2 МОДИФІКАЦІЯ МЕТОДУ ТОЧОК ВИКОРИСТАННЯ ДЛЯ ОЦІНКИ ТРУДОВИТРАТ НА ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ІТ-ПРОЄКТІВ

2.1 Метод точок використання для оцінки трудовитрат

Метод точок використання (Use Case Points) використовується для оцінки трудовитрат за допомогою розміру ІТ-проєктів на основі діаграм UML (Unified Modeling Language) і методології RUP (Rational Unified Process). Метод точок використання базується на тих самих принципах, що й метод функціональних точок. Головна відмінність полягає в заміні одиниць вимірювання з функціональних точок на варіанти використання (Use Cases) [11].

Цей метод оцінки фактично є більш точним, ніж метод функціональних точок [11]. Основна ідея методу полягає в тому, що для проєктування, розробки, тестування та впровадження складного варіанту використання необхідно витратити більше часу ніж для простого варіанту використання. Застереженням тут є те, що ІТ-проєкт має реалізовуватись за класичною методологією (каскадна, V-подібна), а створення варіантів використання повинно починатися вже на етапі збору вимог. Проте метод може бути використаний на ІТ-проєктах з гнучкою методологією, але при умові, що всі функціональні вимоги описані в вигляді варіантів використання.

Першим кроком для кожного методу оцінки трудовитрат є розрахунок обсягу діяльності, яку необхідно виконати. У методі точок використання є 3 основні компоненти, які є важливими для визначення розміру ІТ-проєкту:

- загальна вага акторів (AW);
- загальна вага варіантів використання (UCW);
- фактори технічної складності і складності середовища (TEF).

Актори класифікуються на три різні типи: простий, середній та складний. В таблиці 2.1 описані критерії для кожного типу акторів, а також надані ваги акторів в числовому вигляді. Загальна вага акторів (AW)

розраховується шляхом розподілу всіх акторів на прості, середні та складні, а потім підсумуванням ваг цих акторів [11].

Таблиця 2.1 – Ваги акторів

Тип актора	Вага	Опис
Простий	1	Представляє іншу систему з визначеним API (application programming interface)
Середній	2	Представляє іншу систему, що взаємодіє через мережеві протоколи, наприклад TCP/IP, або людину, яка взаємодіє через текстовий інтерфейс
Складний	3	Представляє людину, яка взаємодіє з системою через графічний інтерфейс (GUI)

Вагу варіанту використання можна обчислити так само, як і вагу актора. Всі варіанти використання потрібно розділити на прості, середні та складні, розподіл представлений в таблиці 2.2 [11]. Загальна вага варіантів використання (UCW) – це сума ваг для всіх категоризованих варіантів використання. Кількість транзакцій, які задіяні в сценарії, впливають на складність варіанту використання та, відповідно, на вагу. Транзакція еквівалентна кроку у варіанті використання. Тому можна визначити кількість транзакцій шляхом підрахунку кроків у варіанті використання.

Існують також альтернативні критерії для оцінювання складності варіантів використання, наприклад, кількість класів, що реалізуються в межах одного варіанта використання, або кількість об’єктів у базі даних, що змінюються в рамках одного варіанта використання.

Таблиця 2.2 – Ваги варіантів використання

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

Після визначення загальних ваг акторів (AW) та варіантів використання (UCW) необхідно їх підсумувати для отримання загальної кількості нескоригованих точок використання (UUCP).

Для отримання загальної кількості скоригованих точок використання (AUCP) необхідно оцінити та врахувати значимість факторів технічної складності і складності середовища для проєкту. Фактори технічної складності і складності середовища представлені в таблиці 2.3 [11]. Технічні фактори використовуються для визначення складності архітектури програмного забезпечення. Фактори середовища використовуються для визначення впливу організаційних ризиків на тестування. Кожен фактор треба оцінити за шкалою від 0 (фактор не значущий) до 5 (фактор має суттєве значення), після чого помножити оцінку фактора на присвоєне значення. Всі добутки сумуються, щоб отримати загальне значення факторів складності (TEF).

Таблиця 2.3 – Фактори технічної складності і складності середовища

Фактор	Опис	Присвоєне значення
T1	Інструменти тестування	5
T2	Задокументовані вхідні дані	5
T3	Середовище розробки	2
T4	Тестове середовище	3
T5	Повторне використання тестового ПЗ	3
T6	Розподілена система	4

Кінець таблиці 2.3

Фактор	Опис	Присвоєне значення
T7	Цілі ефективності	2
T8	Функції безпеки	4
T9	Складна взаємодія	5

Для розрахунку загального значення скоригованих точок використання (AUCP) використовується така ж формула, як і в методі точок використання для розробки програмного забезпечення [11]:

$$AUCP = UUCP * (0.065 + (0.00062 * TEF)), \quad (2.1)$$

де AUCP – значення скоригованих точок використання;

UUCP – значення нескоригованих точок використання;

TEF – загальне значення факторів складності.

Після отримання загального значення скоригованих точок використання необхідно помножити це значення на коефіцієнт перерахунку (CF) для отримання оцінки трудовитрат. Коефіцієнт перерахунку – це співвідношення кількості людино-годин на кожну точку використання. Коефіцієнт перерахунку визначається як загальний час тестування, необхідний для тестування однієї точки використання. Командам необхідно самим визначити коефіцієнт перерахунку, що можна зробити шляхом введення історичних даних минулих IT-проєктів в шаблон оцінки для різних технологій. Якщо не було зібрано жодних історичних даних, експерти галузі пропонують значення від 1 до 5 [12].

Оцінка трудовитрат за методом точок використання розраховується за наступною формулою [11]:

$$Effort = AUCP * CF, \quad (2.2)$$

де Effort – оцінка трудовитрат;

AUCP – значення скоригованих точок використання;

CF – значення коефіцієнта перерахунку.

Перевагами методу є те, що він базується на варіантах використання та може застосовуватися на ранніх етапах життєвого циклу реалізації ІТ-проєктів. Також за цим методом враховуються фактори технічної складності та складності середовища, які можуть бути уточнені в подальшому для досягнення більш точних оцінок. В цей же час головний недолік методу полягає в тому, що цей метод може застосовуватися тільки, коли вимоги до програмного забезпечення написані в вигляді варіантів використання [13].

2.2 Проблематика методу точок використання для оцінки трудовитрат на забезпечення якості ІТ-проєктів

За методом точок використання для розрахунку значення скоригованих точок використання (AUCP) необхідно враховувати фактори технічної складності та складності середовища. Однак факторів, які представлені в таблиці 2.3, недостатньо для коригування значення точок використання в контексті сучасних інформаційних технологій. Хоча технічних факторів може бути достатньо, з факторів середовища представлені лише середовище розробки та тестове середовище, які не відображають весь спектр факторів середовища, що впливають на значення точок використання.

До того ж один з представлених факторів середовища є зайвим, бо цей фактор не впливає достатнім чином на процес тестування, та його використання може призвести до некоректного коригування значень точок використання. Цим фактором є середовище розробки. Після виключення цього фактору з факторів середовища залишається лише тестове середовище, якого

недостатньо для врахування більшості змінних середовища, таких як людський фактор та фактори, що стосуються тестових даних.

Крім того, через постійний розвиток технологій та методик тестування, присвоєні значення кожного фактору є застарілими та потребують перерахування на основі досвіду спеціалістів із забезпечення якості, які мають експертизу в проведені процесу оцінки обсягу робіт з тестування.

Іншою проблемою методу є етап перетворення значення скоригованих точок використання на оцінку трудовитрат. В методі пропонується використовувати історичні дані з минулих IT-проєктів для визначення коефіцієнта перерахунку (CF), а якщо історичні дані відсутні, то брати значення з діапазону від 1 до 5, при цьому обране значення з діапазону нічим не обґрунтовується та залежить лише від досвіду спеціаліста, який використовує метод. Тому процес визначення коефіцієнта перерахунку може бути важким, трудомістким та неточним, що може відштовхнути від використання даного методу. Через це перспективним варіантом вважається створення методу визначення коефіцієнта перерахунку, для якого не треба проводити збір та аналіз історичних даних або обирати значення з діапазону.

2.3 Модифікація методу точок використання для оцінки трудовитрат на забезпечення якості IT-проєктів

Першим етапом модифікації існуючого метода точок використання буде визначення додаткових факторів складності середовища та перерахунок присвоєних значень для усіх факторів. Для зручності використання пропонується розділити фактори складності окремо на технічні фактори та на фактори середовища. Список технічних факторів залишається незмінним, для них лише будуть перераховані присвоєні значення.

Список факторів середовища пропонується оновити шляхом виключення з нього фактору середовища розробки, залишивши лише тестове середовище та додавши додаткові фактори:

- знання програмного забезпечення;
- тестові дані;
- здібності Test Lead спеціаліста;
- мотивація;
- стабільні вимоги;
- працівники з неповною зайнятістю.

Якщо команда тестування вже знайома з програмним забезпеченням, що тестує, то це може зменшити тривалість тестування. Якщо необхідно створити велику кількість тестових даних або якщо тестові дані складні для створення, це може збільшити тривалість тестування. Здібності Test Lead спеціаліста впливають на координацію та розуміння задач всередині команди тестування. Мотивована команда показує кращі показники продуктивності, що може зменшити кількість точок використання. Наявність вимог до системи, що часто змінюються, призводить до збільшення часу тестування. Якщо команда тестування складається з працівників, які працюють неповний робочий день або займаються декількома IT-проєктами одночасно, це може збільшити тривалість тестування.

Після визначення оновлених списків технічних факторів та факторів середовища було проведено опитування експертів із забезпечення якості, які мають рівень не нижче за Senior, для визначення нових присвоєних значень для кожного з факторів складності. Експерти присвоїли факторам значення від 0 до 3, де 0 означає, що фактор незначний та не впливає на тривалість робіт з тестування, а 3 показує, що фактор є критичним та може сильно вплинути на процес тестування.

В результаті опитування були отримані нові присвоєні значення для факторів та розраховані середні значення для кожного з них. Лише середнє

значення фактору «працівники з неповною зайнятістю» було перетворене на від’ємне значення, бо цей фактор завжди негативно впливає на тривалість робіт з тестування. Результати представлені в таблиці 2.4.

Таблиця 2.4 – Нові присвоєні значення факторів складності

Фактори	Оцінки експертів										Середнє значення
	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	
Технічні фактори											
Інструменти тестування	3	1	2	2	2	2	3	1	1	3	2
Задокументовані вхідні дані	2	2	1	1	3	1	3	3	2	2	2
Повторне використання тестового ПЗ	0	3	1	2	1	0	1	0	1	1	1
Розподілена система	3	3	1	1	2	2	2	2	2	2	2
Цілі ефективності	1	1	1	1	1	1	1	1	1	1	1
Функції безпеки	2	0	1	1	2	1	0	2	1	0	1
Складна взаємодія	1	1	1	1	1	1	1	1	1	1	1
Фактори середовища											
Знання програмного забезпечення	2	1	1	2	1	2	0	0	1	0	1
Тестове середовище	2	3	1	1	1	2	3	2	3	2	2
Тестові дані	0	1	2	1	1	0	1	2	1	1	1
Здібності Test Lead спеціаліста	0	1	1	0	0	1	0	1	0	1	0,5
Мотивація	1	1	1	2	0	1	1	1	1	1	1
Стабільні вимоги	2	2	1	1	2	2	3	2	2	3	2
Працівники з неповною зайнятістю	0	2	1	2	0	1	1	2	0	1	-1

Після отримання нових присвоєних значень для технічних факторів та факторів середовища необхідно розділити їх на окремі групи, що представлені в таблицях 2.5-2.6.

Таблиця 2.5 – Технічні фактори складності

Фактор	Опис	Присвоєне значення
T1	Інструменти тестування	2
T2	Задокументовані вхідні дані	2
T3	Повторне використання тестового ПЗ	1
T4	Розподілена система	2
T5	Цілі ефективності	1
T6	Функції безпеки	1
T7	Складна взаємодія	1

Для подальшого використання в методі точок використання кожен технічний фактор має бути оцінений в діапазоні від 0 до 5, де 0 показує відсутність значимості фактору для ІТ-проєкту, а 5 показує велике значення фактору. Після проведення оцінки кожного технічного фактору, отримані оцінки мають бути помножені на присвоєні значення конкретного фактору та підсумовані для отримання суми зважених значень технічних факторів (TFactor). TFactor потім використовується для розрахунку загального значення факторів технічної складності (TCF) за наступною формулою [14]:

$$TCF = 0.6 + (0.014 * TFactor), \quad (2.3)$$

де TCF – загальне значення факторів технічної складності;

TFactor – сума зважених значень технічних факторів.

Таблиця 2.6 – Фактори складності середовища

Фактор	Опис	Присвоєне значення
E1	Знання програмного забезпечення	1
E2	Тестове середовище	2
E3	Тестові дані	1
E4	Здібності Test Lead спеціаліста	0.5
E5	Мотивація	1
E6	Стабільні вимоги	2
E7	Працівники з неповною зайнятістю	-1

Аналогічним чином кожен фактор середовища має бути оцінений в діапазоні від 0 до 5, але не для всіх факторів значення оцінки є однаковим. Для факторів E1-E4 0 означає відсутність, 3 – середній рівень, 5 – високий рівень. Для фактору E5 0 означає відсутність мотивації, 3 – середній рівень, 5 – високий рівень мотивації. Для фактору E6 0 означає високу нестабільність вимог, 3 – середню, 5 – стабільні вимоги. Для фактору E7 0 означає відсутність спеціалістів з частковою зайнятістю, 3 – середній рівень, 5 – всі спеціалісти з частковою зайнятістю.

Після проведення оцінки кожного фактору середовища отримані оцінки мають бути помножені на присвоєні значення конкретного фактору та підсумовані для отримання суми зважених значень факторів середовища (EFactor). EFactor потім використовується для розрахунку загального значення факторів складності середовища (ECF) за наступною формулою [14]:

$$ECF = 1.14 + (-0.0362 * EFactor), \quad (2.4)$$

де ECF – загальне значення факторів складності середовища;

EFactor – сума зважених значень факторів середовища.

Отримані значення TCF та ECF потім використовуються для отримання значення скоригованих точок використання (AUCP) за наступною формулою [14]:

$$AUCP = UUCP * TCF * ECF, \quad (2.5)$$

де AUCP – значення скоригованих точок використання;

UUCP – значення нескоригованих точок використання;

TCF – загальне значення факторів технічної складності;

ECF – загальне значення факторів зовнішньої складності.

Другим етапом модифікації методу є створення методу визначення коефіцієнта перерахунку (CF), в якому враховуються рівень кваліфікації команди забезпечення якості (QA) та складність IT-проєкту.

Для розрахунку показника загального рівня кваліфікації команди QA пропонується використовувати наступну формулу, за якою враховується кваліфікація кожного члена команди QA:

$$QA\_exp = \frac{0.7*Lead\_QA+1*Senior\_QA+0.9*Middle\_QA+0.5*Junior\_QA}{Total\_QA}, \quad (2.6)$$

де QA_exp – показник рівня кваліфікації команди QA;

Lead_QA, Senior_QA, Middle_QA, Junior_QA – кількість QA інженерів відповідного рівня в команді;

Total_QA – загальна кількість QA інженерів.

Коефіцієнти для кожного рівня QA інженерів в команді були визначені після проведення аналізу досвіду провідних IT-компаній. Спираючись на цей досвід, було визначено, що у співробітника рівня Middle або Senior продуктивність на IT-проєкті буде на рівні 90-100%. Продуктивність співробітника залежить від його вмінь, що дозволяють йому виконувати поставлені задачі у визначений термін. Усі інші фахівці поступаються за

відсотком. Далі для зручності в підрахунках значення продуктивності в 100% буде прийматись за 1. Відповідно, якщо продуктивність співробітника менша за 1, то це буде 0.9, 0.8 і так далі. На робочі мітинги у Lead спеціалістів закладається 30%. Так відбувається, тому що всі вхідні дані проходять через Lead спеціалістів, і фахівцеві рівнем нижче в роботу приходить уже декомпозоване завдання з мінімумом невизначеностей. Продуктивність Lead спеціалістів на IT-проєкті тим самим знижується до 0.7 [15].

Складність IT-проєкту може бути визначена після проведення оцінки факторів складності середовища, які представлені в таблиці 2.6. Складність проєкту буде впливати на початкове значення коефіцієнта перерахунку (ICF), якщо IT-проєкт визначений як складний, то початкове значення коефіцієнта буде дорівнювати 3, якщо IT-проєкт простий, то значення буде дорівнювати 1.

Для визначення складності IT-проєкту необхідно після отримання оцінок факторів складності середовища проаналізувати їх та визначити, скільки оцінок факторів E1-E6 мають значення менше 3 і чи має фактор E7 значення більше 3. Потім треба підсумувати кількість оцінок, які потрапляють під визначені вище умови. Якщо загальна кількість менша або дорівнює 3, то IT-проєкт можна вважати простим, якщо загальна кількість більше 3, то IT-проєкт є складним [12].

Для розрахунку значення коефіцієнта перерахунку (CF) треба отримане початкове значення коефіцієнта (ICF) помножити на показник рівня кваліфікації команди QA. Цей метод визначення коефіцієнта перерахунку пропонується використовувати командам, в яких немає визначеного стандартного значення цього коефіцієнта. Розрахунок представлений в вигляді наступної формули:

$$CF = ICF * (2 - QA\_exp), \tag{2.7}$$

де CF – значення коефіцієнта перерахунку;

ICF – початкове значення коефіцієнта перерахунку;

QA_exp – показник рівня кваліфікації команди QA.

3 РОЗРОБКА МЕТОДИКИ ДЛЯ МОДИФІКОВАНОГО МЕТОДУ ТОЧОК ВИКОРИСТАННЯ ДЛЯ ОЦІНКИ ТРУДОВИТРАТ НА ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ІТ-ПРОЄКТІВ

3.1 Методика використання оригінального методу точок використання

Фундаментальною особливістю використання методу точок використання є те, що всі варіанти використання повинні бути написані на рівні цілей користувача (user goal level). Ціль варіанту використання на рівні цілей користувача є фундаментальною одиницею бізнес-цінності [14]. Приклад такого варіанту використання представлений в таблиці 3.1 [16]. Цей варіант використання був створений для онлайн платформи навчання. В ньому описується ситуація, коли студентам необхідно проходити тести та отримувати миттєво результати.

Таблиця 3.1 – Варіант використання «Quiz Instant Results»

Параметр	Детальний опис
Опис	Функціонал, який дозволяє студентам проходити тести та отримувати миттєво результати
Актори	Студент, система
Цілі	Пройти тест, переглянути результати тесту
Передумови	Студент має бути авторизований в системі, студент повинен мати доступ до тесту
Основний шлях	1. Студент переходить в секцію «Тести». 2. Студент обирає опцію пройти тест. 3. Система представляє студенту питання тесту. 4. Студент відповідає на запитання. 5. Студент зберігає відповіді. 6. Система оцінює відповіді студента. 7. Система робить результати доступними для перегляду.

Кінець таблиці 3.1

Параметр	Детальний опис
Основний шлях	8. Студент переглядає результати тесту.
Альтернативний шлях	5а. 5а1. Студент не зберігає відповіді. 5а2. Система очищує відповіді студента. Варіант використання продовжується з кроку 2 основного шляху.

Етап 1. Визначення загальної ваги акторів (AW). Актором у варіанті використання може бути людина, інша програма, апаратне забезпечення тощо. Деякі актори, такі як користувач, що працює з простим інтерфейсом командного рядка, мають дуже прості потреби і збільшують складність варіанту використання лише незначною мірою. Інші учасники, такі як користувач, що працює з інтерактивним графічним інтерфейсом (GUI), мають набагато більший вплив на зусилля з тестування варіанту використання. Щоб відобразити ці відмінності, кожен актор в системі класифікується як простий, середній або складний, і йому присвоюється вага.

Для визначення загальної ваги акторів необхідно проаналізувати кожен варіант використання та визначити усіх акторів. Після цього треба визначити тип кожного актора та його вагу за таблицею 2.1. Для отримання значення AW треба підсумувати значення ваг усіх акторів.

Для прикладу визначення ваг акторів взято варіант використання, який представлений в таблиці 3.1. В даному варіанті використання представлені два актори. Першим актором є студент, який взаємодіє з системою через графічний інтерфейс (GUI) при проходженні тесту та перегляду результатів, тому його тип – складний (вага 3). Другий актор представляє систему з визначеним API, тому його тип – простий (вага 1). Загальна вага акторів (AW) буде дорівнювати 4.

Етап 2. Визначення загальної ваги варіантів використання (UCW). Кожен варіант використання оцінюється на основі кількості транзакцій в

межах варіанту використання. Транзакція (при роботі з варіантами використання на рівні цілей користувача) еквівалентна кроку у варіанті використання. Також треба враховувати не тільки кроки з основного шляху, а й кроки, які представлені в альтернативному шляху. Кількість кроків, які задіяні в варіанті використання, впливають на його складність та відповідно на вагу.

Для визначення загальної ваги варіантів використання необхідно проаналізувати кожен варіант використання та визначити кількість кроків в кожному. Після цього треба визначити тип кожного варіанта використання та його вагу за таблицею 2.2. Для отримання значення UCW треба підсумувати значення ваг усіх варіантів використання.

Для прикладу визначення ваги варіанту використання взято варіант використання, який представлений в таблиці 3.1. Тип варіанту використання в даному прикладі – складний. У варіанті використання представлено 10 транзакцій: 8 в основному шляху та 2 в альтернативному шляху. Тому вага такого варіанта використання буде дорівнювати 15.

Етап 3. Визначення загальної кількості нескоригованих точок використання (UUCP). Після визначення загальної ваги акторів (AW) та варіантів використання (UCW), необхідно їх підсумувати для отримання загальної кількості нескоригованих точок використання (UUCP).

Етап 4. Оцінка значимості факторів технічної складності і складності середовища для IT-проєкту. Фактори технічної складності і складності середовища представлені в таблиці 2.3. Технічні фактори використовуються для визначення складності архітектури програмного забезпечення. Фактори середовища використовуються для визначення впливу організаційних ризиків на тестування. Кожен фактор треба оцінити за шкалою від 0 (фактор не значущий) до 5 (фактор має суттєве значення), після чого помножити оцінку фактора на його присвоєне значення. Всі добутки підсумовуються, щоб отримати загальне значення факторів складності (TEF). Отримане значення необхідне для наступного кроку.

Для прикладу оцінки фактору складності можна взяти фактор «розподілена система» для варіанту використання, який представлений в таблиці 3.1. Розподілена система є важливим фактором для IT-проєкту платформи навчання, оскільки вона забезпечує масштабованість, високу доступність (якщо один сервер стає недоступним, інші сервери можуть продовжувати обслуговувати користувачів), відмовостійкість та ефективне використання ресурсів. Однак, тестування розподіленої системи є більш складним. Це пов’язано з тим, що необхідно тестувати не тільки окремі компоненти системи, але й їх взаємодію між собою. Тому оцінка має дорівнювати 4.

Етап 5. Визначення загального значення скоригованих точок використання (AUCP). Для розрахунку загального значення скоригованих точок використання (AUCP) використовується формула 2.1, в якій використовуються отримані на минулих кроках значення UUCP та TEF.

Етап 6. Отримання значення оцінки трудовитрат (Effort). Після отримання загального значення скоригованих точок використання (AUCP) необхідно помножити це значення на коефіцієнт перерахунку (CF) для отримання оцінки трудовитрат. Командам необхідно самим визначити коефіцієнт перерахунку, що можна зробити шляхом введення історичних даних минулих IT-проєктів в шаблон оцінки для різних технологій. Якщо не було зібрано жодних історичних даних, експерти галузі пропонують значення від 1 до 5.

Оцінка трудовитрат за методом точок використання розраховується за формулою 2.2, в якій враховуються загальне значення скоригованих точок використання та коефіцієнт перерахунку.

3.2 Методика використання модифікованого методу точок використання

Етап 1. Визначення загальної ваги акторів (AW). Для визначення загальної ваги акторів необхідно проаналізувати кожен варіант використання та визначити усіх акторів. Після цього треба визначити тип кожного актора та його вагу за таблицею 2.1. Для отримання значення AW треба підсумувати значення ваг усіх акторів.

Етап 2. Визначення загальної ваги варіантів використання (UCW). Для визначення загальної ваги варіантів використання необхідно проаналізувати кожен варіант використання та визначити кількість кроків в кожному. Після цього треба визначити тип кожного варіанта використання та його вагу за таблицею 2.2. Для отримання значення UCW треба підсумувати значення ваг усіх варіантів використання.

Етап 3. Визначення загальної кількості нескоригованих точок використання (UUCP). Після визначення загальної ваги акторів (AW) та варіантів використання (UCW), необхідно їх підсумувати для отримання загальної кількості нескоригованих точок використання (UUCP).

Етап 4. Оцінка значимості факторів технічної складності для IT-проєкту. Для подальшого використання в методі точок використання кожен технічний фактор має бути оцінений в діапазоні від 0 до 5, де 0 показує відсутність значимості фактору для IT-проєкту, а 5 показує велике значення фактору. Фактори технічної складності представлені в таблиці 2.5. Після проведення оцінки кожного технічного фактору отримані оцінки мають бути помножені на присвоєні значення конкретного фактору та підсумовані для отримання суми зважених значень технічних факторів (TFactor). TFactor потім використовується для розрахунку загального значення факторів технічної складності (TCF) за формулою 2.3.

Етап 5. Оцінка значимості факторів складності середовища для IT-проєкту. Аналогічним чином кожен фактор середовища, який

представлений в таблиці 2.6, має бути оцінений в діапазоні від 0 до 5, але не для всіх факторів значення оцінки є однаковим. Для факторів E1-E4 0 означає відсутність, 3 – середній рівень, 5 – високий рівень. Для фактору E5 0 означає відсутність мотивації, 3 – середній рівень мотивації, 5 – високий рівень мотивації. Для фактору E6 0 означає високу нестабільність вимог, 3 – середню, 5 – стабільні вимоги. Для фактору E7 0 означає відсутність спеціалістів з частковою зайнятістю, 3 – середній рівень, 5 – всі спеціалісти з частковою зайнятістю.

Після проведення оцінки кожного фактору середовища отримані оцінки мають бути помножені на присвоєні значення конкретного фактору та підсумовані для отримання суми зважених значень факторів середовища (EFactor). EFactor потім використовується для розрахунку загального значення факторів складності середовища (ECF) за формулою 2.4.

Етап 6. Визначення загального значення скоригованих точок використання (AUCP). Для розрахунку загального значення скоригованих точок використання (AUCP) використовується формула 2.5, в якій використовуються отримані на минулих кроках значення UUCP, TCF та ECF.

Етап 7. Визначення показника загального рівня кваліфікації команди QA (QA_exp). Необхідно проаналізувати команду забезпечення якості на IT-проєкті, визначити кількість співробітників кожного рівня (Lead, Senior, Middle, Junior) та підставити ці значення в формулу 2.6 для розрахунку показника загального рівня кваліфікації команди.

Етап 8. Визначення початкового значення коефіцієнта перерахунку (ICF). Для цього необхідно спочатку визначити складність IT-проєкту, що можна зробити після проведення оцінки факторів складності середовища. Якщо IT-проєкт визначений як складний, то початкове значення коефіцієнта буде дорівнювати 3, якщо IT-проєкт простий, то значення буде дорівнювати 1.

Для визначення складності IT-проєкту необхідно після отримання оцінок факторів складності середовища проаналізувати їх та визначити, скільки оцінок факторів E1-E6 мають значення менше 3, і чи має фактор E7

значення більше 3. Потім треба підсумувати кількість оцінок, які потрапляють під визначені вище умови. Якщо загальна кількість менша або дорівнює 3, то IT-проєкт можна вважати простим, якщо загальна кількість більше 3, то IT-проєкт є складним.

Фактори складності середовища були оцінені наступним чином:

– фактор E1 (знання програмного забезпечення) має оцінку 5;
– фактор E2 (тестове середовище) має оцінку 3;
– фактор E3 (тестові дані) має оцінку 1;
– фактор E4 (здібності Test Lead спеціаліста) має оцінку 1;
– фактор E5 (мотивація) має оцінку 5;
– фактор E6 (стабільні вимоги) має оцінку 2;
– фактор E7 (працівники з неповною зайнятістю) має оцінку 2.

Після проведення аналізу отриманих оцінок можна визначити, що серед оцінок факторів E1-E6 мають значення менше 3 тільки три фактори, та оцінка фактору E7 має значення 2, тому вона не підпадає під визначені вимоги. Таким чином можна визначити, що IT-проєкт є простим (ICF = 1), бо загальна кількість оцінок, які потрапляють під визначені вище умови, дорівнює 3.

Етап 9. Розрахунок значення коефіцієнта перерахунку (CF). Для проведення цього розрахунку треба отримане початкове значення коефіцієнта (ICF) помножити на показник рівня кваліфікації команди QA, розрахунок представлений формулою 2.7.

Етап 10. Отримання значення оцінки трудовитрат (Effort). Після отримання загального значення скоригованих точок використання (AUCP) необхідно помножити це значення на коефіцієнт перерахунку (CF) для отримання оцінки трудовитрат, що розраховується за формулою 2.2.

Таким чином можна отримати точну оцінку трудовитрат, необхідних для виконання задач по тестуванню, які враховують значимість факторів технічної складності і складності середовища, складність IT-проєкту та особливості конкретної команди QA.

4 АПРОБАЦІЯ ОРИГІНАЛЬНОГО ТА МОДИФІКОВАНОГО МЕТОДУ ТОЧОК ВИКОРИСТАННЯ ДЛЯ ОЦІНКИ ТРУДОВИТРАТ НА ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ІТ-ПРОЄКТІВ

4.1 Опис ІТ-проєкту

Продукт, який створюється в ході ІТ-проєкту, надає можливість менеджерам записувати прийняті рішення щодо проєкту в журнал рішень. Така можливість є дуже корисною у разі виникнення потенційної конфліктної ситуації. Щоразу, коли виникає питання, чи було прийнято рішення, що саме було вирішено або чи всі аспекти були враховані, журнал рішень буде містити відповідь і забезпечувати основу для нових рішень. Рішення можуть додаватися різними стейкхолдерами, а особа, яка є рецензентом, має погодитися з ними або відхилити. Оскільки дуже важливо відстежувати всі зміни щодо рішень, вони будуть зберігатись в базі даних.

Конкретними групами користувачів продукту є менеджери різних типів: менеджери проєктів, менеджери з постачання, менеджери по роботі з клієнтами, віце-президенти тощо.

Продукт являє собою веб-додаток з графічним інтерфейсом для користувачів персональних комп’ютерів і мобільних користувачів, що дозволяє записувати прийняті рішення з розбивкою по проєктах і компаніях. Веб-додаток розробляється таким чином, щоб бути зручним та інтуїтивно зрозумілим з мінімальним часом навчанням, необхідним для початку роботи, і з мінімальною кількістю функцій, що дозволяють задовольнити бізнес-потреби.

Веб-додатки використовують клієнт-серверну архітектуру, де браузер є клієнтом, а веб-сервер є сервером. Клієнт відправляє запит до сервера, а сервер відповідає на цей запит, надсилаючи запитані дані назад до клієнта [17].

Особи, які приймають рішення, мають можливість переглянути і погодитися або не погодитися з рішеннями, які були записані. У разі відхилення рішення необхідно буде надати коментарі.

Проблеми, які вирішуються, за допомогою створюваного продукту:

– непорозуміння під час проєктних зустрічей або переговорів: кожен учасник повинен бути впевнений, що він все правильно зрозумів та його правильно зрозуміли;

– виникнення конфліктів із замовником, коли замовник не погоджується з тим, що саме було вирішено під час попередніх зустрічей і переговорів.

Цілі створення продукту:

– надати командам централізоване місце для документування та впорядкування рішень, прийнятих під час проєкту;

– забезпечити чітке документування деталей рішень, включаючи осіб, які приймають рішення, і статус рішення;

– забезпечити легкий доступ до інформації про рішення для всіх членів команди;

– забезпечити контроль доступу на основі ролей (RBAC) для контролю змін у рішеннях.

Команда забезпечення якості складається з наступних спеціалістів: один спеціаліст рівня Lead, один спеціаліст рівня Senior та два спеціалісти рівня Middle. Саме ці спеціалісти будуть проводити тестування функціоналу застосунку, який описаний в варіантах використання.

Визначення трудовитрат, необхідних для тестування, за оригінальним та модифікованим методом точок використання буде проходити на основі п’яти варіантів використання, які представлені в таблицях 4.1-4.5.

Таблиця 4.1 – Варіант використання «User Registration»

Параметр	Детальний опис
Опис	Функціонал, який дозволяє користувачам пройти процес реєстрації та створити новий аккаунт
Актори	Користувач, система
Цілі	Зареєструватися в додатку
Передумови	Користувач отримує посилання-запрошення і переходить до додатку за допомогою браузера
Основний шлях	1. Користувач переглядає форму авторизації. 2. Користувач натискає на кнопку «Sign Up». 3. Користувач переглядає форму реєстрації з вимкненою кнопкою «Sign Up». 4. Користувач вводить ім’я в текстове поле «Name». 5. Користувач вводить електрону адресу в текстове поле «E-mail». 6. Користувач вводить пароль в текстові поля «Password» та «Re-type Password». 7. Користувач натискає на іконку з оком, щоб переглянути введений пароль. 8. Користувач натискає на кнопку «Sign Up». 9. Система перевіряє введені дані. 10. Система надсилає повідомлення на вказану користувачем електронну пошту з посиланням для активації облікового запису (посилання активне протягом 24 годин). 11. Користувач відкриває повідомлення у своїй поштовій скриньці. 12. Користувач натискає на посилання, щоб перейти на сторінку авторизації. 13. Система активує обліковий запис користувача. 14. Система надає користувачеві доступ до повної функціональності додатку. 15. Система надсилає повідомлення в область сповіщень «Your account has successfully activated».

Кінець таблиці 4.1

Параметр	Детальний опис
Альтерна-тивний шлях	9a. Введені дані є неправильними: 9a1. Система надсилає повідомлення в область сповіщень «Please check your authorization details». 9a2. Система позначає поля з неправильними даними червоним кольором. 9a3. Система вимикає кнопку «Sign Up», поки користувач не введе правильні дані. Варіант використання продовжується з кроку 4 основного шляху. 9b. Користувач з введеною електронною адресою вже існує: 9b1. Система надсилає повідомлення в область сповіщень «Entered e-mail address is already in use. Please log in or use a different e-mail address.». 9b2. Система позначає поле «E-mail» червоним кольором. 9b3. Система вимикає кнопку «Sign Up», поки користувач не введе правильну електронну адресу. Варіант використання продовжується з кроку 5 основного шляху. 12a. Термін дії посилання закінчився: 12a1. Система не активує обліковий запис користувача. 12a2. Система надсилає повідомлення в область сповіщень «Failed to activate account. Please try to sign up one more time.». Варіант використання продовжується з кроку 2 основного шляху.
Час тестування	7 людино-годин

Вимоги до текстових полів в варіанті використання «User Registration» є наступними:

a) текстове поле «Name»;

1) дозволено використовувати великі та малі літери;

2) спеціальні символи не допускаються;

3) допускається введення символів підкреслення, короткого дефісу та крапки;

4) дозволено вводити цифри;

5) дозволено вводити тільки латинські символи;

б) текстове поле «E-mail»;

1) присутній символ «@»;

2) до символу «@» два символи – мінімальна довжина;

3) спеціальні символи не допускаються;

4) допускаються тільки латинські символи;

в) текстові поля «Password» та «Re-type Password»;

1) мінімальна довжина: 8 символів;

2) має містити великі та малі літери;

3) має містити принаймні 1 цифру;

4) має містити принаймні 1 нелітерний символ.

Таблиця 4.2 – Варіант використання «Log In»

Параметр	Детальний опис
Опис	Функціонал, який дозволяє користувачам зі створеними обліковими записами пройти процес авторизації
Актори	Користувач, система
Цілі	Авторизуватися в додатку
Передумови	1. Користувач перейшов до додатку за посилання у браузері. 2. Користувач був перенаправлений на форму авторизації в систему. 3. Користувач зареєстрований в системі.
Основний шлях	1. Користувач переглядає форму авторизації. 2. Користувач вводить зареєстровану адресу електронної пошти та пароль. 3. Користувач натискає на «Sign In» кнопку.

Кінець таблиці 4.2

Параметр	Детальний опис
Основний шлях	4. Система перевіряє правильність введеної адреси електронної пошти та паролю. 5. Система перенаправляє користувача на головний екран додатку.
Альтерна-тивний шлях	4a. Введені дані є неправильними: 4a1. Система надсилає повідомлення в область сповіщень «Failed to sign in. Please check your authorization details». 4a2. Система не авторизує користувача. Варіант використання продовжується з кроку 2 основного шляху.
Час тестування	3 людино-години

Таблиця 4.3 – Варіант використання «Decision Log Creation»

Параметр	Детальний опис
Опис	Функціонал, який дозволяє користувачам створювати журнал рішень
Актори	Користувач, система
Цілі	Створити журнал рішень
Передумови	1. Користувач авторизований в системі. 2. Користувач знаходиться на головному екрані додатку.
Основний шлях	1. Користувач натискає кнопку «New Decision Log». 2. Система пропонує ввести необхідні дані: назву журналу рішень, опис журналу рішень та права доступу. 3. Користувач заповнює обов’язкове поле – назва журналу рішень (кількість символів в діапазоні від 1 до 300 включно, допускаються будь-які символи). 4. Користувач заповнює необов’язкові поля: опис журналу рішень (кількість символів у діапазоні від 1 до 1000 включно, допускаються будь-які символи) та права доступу. 5. Користувач натискає кнопку «Create».

Кінець таблиці 4.3

Параметр	Детальний опис
Основний шлях	6. Система перевіряє введені дані. 7. Система створює новий журнал рішень. 8. Система додає створений журнал рішень в початок списку журналів.
Альтерна-тивний шлях	6а. Введені дані є неправильними: 6а1. Система надсилає повідомлення в область сповіщень «Oops! It looks like there was an error with the information you provided. Please review your input and ensure that all fields contain valid data.». 6а2. Система виділяє поля введення червоним кольором. Варіант використання продовжується з кроку 3 основного шляху.
Час тестування	4 людино-години

Таблиця 4.4 – Варіант використання «Creation of a Separate Decision»

Параметр	Детальний опис
Опис	Функціонал, який дозволяє користувачам створювати рішення всередині журналу рішень
Актори	Користувач, система
Цілі	Створити рішення
Передумови	1. Користувач авторизований в системі. 2. Користувач знаходиться на сторінці журналу рішень. 3. Користувач є власником журналу рішень.
Основний шлях	1. Користувач натискає кнопку «New Decision». 2. Система пропонує ввести необхідні дані: назву, деталі рішення, міркування, термін надання рецензії та перемикач автоматичного опрацювання рішення в випадку спливу терміну рецензії («Approved» вибрано за замовчуванням).

Кінець таблиці 4.4

Параметр	Детальний опис
Основний шлях	3. Користувач заповнює обов’язкові поля: назва (кількість символів від 1 до 300), деталі рішення (кількість символів від 1 до 1000), термін надання рецензії та перемикач автоматичного опрацювання рішення. 4. Користувач заповнює необов’язкове поле – міркування (кількість символів від 1 до 1000). 5. Користувач натискає кнопку «Create». 6. Система перевіряє введені дані. 7. Система створює нове рішення. 8. Система додає створене рішення в початок списку рішень.
Альтерна-тивний шлях	6а. Введені дані є неправильними: 6а1. Система надсилає повідомлення в область сповіщень «Oops! It looks like there was an error with the information you provided. Please review your input and ensure that all fields contain valid data.». 6а2. Система виділяє поля введення червоним кольором. Варіант використання продовжується з кроку 3 основного шляху.
Час тестування	4 людино-години

Таблиця 4.5 – Варіант використання «Decision Approval Process»

Параметр	Детальний опис
Опис	Функціонал, який дозволяє користувачам затверджувати рішення всередині журналу рішень
Актори	Користувач, система
Цілі	Затвердити рішення
Передумови	1. Користувач авторизований в системі. 2. Користувач знаходиться на сторінці журналу рішень.

Кінець таблиці 4.5

Параметр	Детальний опис
Передумови	3. Користувач має права рецензента.
Основний шлях	1. Користувач переглядає список рішень з наступними атрибутами: назва рішення, дата додавання, рецензенти, статус, дії (редагування та архівування). 2. Користувач натискає на назву необхідного рішення (приховане посилання). 3. Система перенаправляє користувача на сторінку обраного рішення на якій представлені: назва рішення, поле деталей рішення, імена рецензентів, кнопки «Approve» та «Decline», порожній розділ «Comments» та кнопка «Add Comment». 4. Користувач натискає кнопку «Approve». 5. Система помічає рішення як затверджене, кнопки «Approve» та «Decline» стають неактивними, в розділі «Comments» з’являється повідомлення про те, що рішення було затверджено. 6. Користувач повертається до журналу рішень. 7. Система оновлює список рішень та змінює статус опрацьованого рішення.
Альтерна-тивний шлях	4a. 4a1. Користувач натискає кнопку «Decline». 4a2. Система показує спливаюче вікно з полем «Please add decline reason» та кнопками «Cancel» і «Decline». 4a3. Користувач заповнює причину відхилення (1-1000 символів, дозволені будь-які літери) та натискає кнопку «Decline». 4a4. Система помічає рішення як відхилене, кнопки «Approve» та «Decline» стають неактивними, в розділі «Comments» з’являється повідомлення про те, що рішення було відхилене. Варіант використання продовжується з кроку 6 основного шляху.
Час тестування	5 людино-години

Представлені варіанти використання будуть використані для визначення трудовитрат за допомогою оригінального методу (підрозділ 4.2) та модифікованого методу точок використання (підрозділ 4.3). Після чого отримані результати будуть порівняні з фактичним часом, який був витрачений на тестування даних варіантів використання, для визначення точності оригінального та модифікованого методів. Фактичний час, який був витрачений на тестування, складає 23 людино-години.

4.2 Приклад використання оригінального методу точок використання

У прикладі зроблені розрахунки на основі оригінального методу точок використання. Першим етапом є визначення загальної ваги акторів (AW). Для цього необхідно проаналізувати кожен варіант використання, визначити акторів та класифікувати їх за таблицею 2.1. У варіантах використання (таблиці 4.1-4.5) представлені два актора: користувач та система. Користувач є складним актором, бо він взаємодіє з додатком через графічний інтерфейс, тому його вага буде дорівнювати 3. Другий актор представляє систему з визначеним API, тому його тип простий та вага дорівнює 1. Таким чином, вага акторів для одного варіанту використання буде дорівнювати 4. Але, оскільки визначені актори представлені в усіх п’яти варіантах використання, необхідно 4 помножити на кількість варіантів використання для отримання загальної ваги акторів. Загальна вага акторів (AW) буде дорівнювати 20.

Другим етапом є визначення загальної ваги варіантів використання (UCW). Для цього необхідно проаналізувати варіанти використання та визначити кількість транзакцій в кожному. Після цього треба визначити тип кожного варіанта використання та його вагу за таблицею 2.2. Результати аналізу варіантів використання представлені в таблиці 4.6.

Таблиця 4.6 – Аналіз варіантів використання

Варіант використання	Кількість транзакцій	Тип	Вага
User registration	23	Складний	15
Log In	7	Середній	10
Decision Log Creation	10	Складний	15
Creation of a Separate Decision	10	Складний	15
Decision Approval Process	11	Складний	15
		UCW	70

Третім етапом є визначення загальної кількості нескоригованих точок використання (UUCP). Після визначення загальної ваги акторів (AW = 20) та варіантів використання (UCW = 70) необхідно їх підсумувати. Таким чином, загальна кількість нескоригованих точок використання (UUCP) буде дорівнювати 90.

Четвертим етапом є оцінка значимості факторів технічної складності і складності середовища для ІТ-проєкту. Необхідно оцінити значимість факторів, які представлені в таблиці 2.3. Після чого отримані оцінки факторів помножити на присвоєне значення відповідного фактору та підсумувати добутки для отримання загального значення факторів складності (TEF). Результати оцінки факторів представлені в таблиці 4.7.

Таблиця 4.7 – Оцінка факторів складності

Фактор	Опис	Присвоєне значення	Оцінка	Розрахований фактор
T1	Інструменти тестування	5	3	15
T2	Задокументовані вхідні дані	5	3	15
T3	Середовище розробки	2	0	0
T4	Тестове середовище	3	3	9

Кінець таблиці 4.7

Фактор	Опис	Присвоєне значення	Оцінка	Розрахований фактор
T5	Повторне використання тестового ПЗ	3	0	0
T6	Розподілена система	4	4	16
T7	Цілі ефективності	2	2	4
T8	Функції безпеки	4	1	4
T9	Складна взаємодія	5	1	5
			TEF	68

П’ятим етапом є визначення загального значення скоригованих точок використання (AUCP). Розрахунок відбувається за формулою 2.1. Результати розрахунків представлені нижче:

$$AUCP = 90 * \big(0.065 + (0.00062 * 68)\big) = 90 * 0.1077 = 9.693$$

Шостим етапом є розрахунок значення оцінки трудовитрат (Effort). Після отримання загального значення скоригованих точок використання (AUCP) необхідно помножити це значення на коефіцієнт перерахунку (CF) для отримання оцінки трудовитрат. Коефіцієнт перерахунку буде дорівнювати 2, бо це значення знаходиться між значеннями ICF для простого та складного ІТ-проєктів. Таким чином, оцінка трудовитрат буде дорівнювати 19.39 людино-годин.

4.3 Приклад використання модифікованого методу точок використання

У прикладі зроблені розрахунки на основі модифікованого методу точок використання.

Етапи визначення загальної ваги акторів (AW) та загальної ваги варіантів використання (UCW) не відрізняються між оригінальним та модифікованим методом. Тому значення AW та UCW залишаються незмінними. Таким чином, загальна кількість нескоригованих точок використання (UUCP) залишається рівною 90. Розрахунки будуть продовжені з четвертого етапу.

Четвертим етапом є оцінка значимості факторів технічної складності для IT-проєкту. Необхідно оцінити значимість факторів технічної складності, які представлені в таблиці 2.5. Після чого отримані оцінки факторів помножити на присвоєне значення відповідного фактору та підсумувати добутки для отримання суми зважених значень технічних факторів (TFactor). Результати оцінки факторів представлені в таблиці 4.8.

Таблиця 4.8 – Оцінка факторів технічної складності

Фактор	Опис	Присвоєне значення	Оцінка	Розрахований фактор
T1	Інструменти тестування	2	3	6
T2	Задокументовані вхідні дані	2	3	6
T3	Повторне використання тестового ПЗ	1	0	0
T4	Розподілена система	2	4	8
T5	Цілі ефективності	1	2	2
T6	Функції безпеки	1	1	1
T7	Складна взаємодія	1	1	1
			TFactor	24

TFactor використовується для розрахунку загального значення факторів технічної складності (TCF) за формулою 2.3. Результати розрахунків представлені нижче:

$$TCF = 0.6 + (0.014 * 24) = 0.6 + 0.336 = 0.936$$

П’ятим етапом є оцінка значимості факторів складності середовища для ІТ-проєкту. Необхідно оцінити значимість факторів складності середовища, які представлені в таблиці 2.6. Після чого отримані оцінки факторів помножити на присвоєне значення відповідного фактору та підсумувати добутки для отримання суми зважених значень факторів середовища (EFactor). Результати оцінки факторів представлені в таблиці 4.9.

Таблиця 4.9 – Оцінка факторів складності середовища

Фактор	Опис	Присвоєне значення	Оцінка	Розрахований фактор
E1	Знання програмного забезпечення	1	4	4
E2	Тестове середовище	2	3	6
E3	Тестові дані	1	2	2
E4	Здібності Test Lead спеціаліста	0.5	4	2
E5	Мотивація	1	4	4
E6	Стабільні вимоги	2	3	6
E7	Працівники з неповною зайнятістю	-1	0	0
			EFactor	24

EFactor використовується для розрахунку загального значення факторів складності середовища (ECF) за формулою 2.4. Результати розрахунків представлені нижче:

$$ECF = 1.14 + (-0.0362 * 24) = 1.14 - 0.8688 = 0.2712$$

Шостим етапом є визначення загального значення скоригованих точок використання (AUCP) за формулою 2.5. Результати розрахунків представлені нижче:

$$AUCP = UUCP * TCF * ECF = 90 * 0.936 * 0.2712 = 22.8$$

Сьомим етапом є визначення показника загального рівня кваліфікації команди QA (QA_exp). Для цього необхідно проаналізувати команду забезпечення якості на IT-проєкті, визначити кількість співробітників кожного рівня та підставити ці значення в формулу 2.6. Команда забезпечення якості складається з наступних спеціалістів: один спеціаліст рівня Lead, один спеціаліст рівня Senior та два спеціалісти рівня Middle.

Результати розрахунків представлені нижче:

$$QA_{exp} = \frac{0.7 * 1 + 1 * 1 + 0.9 * 2 + 0.5 * 0}{4} = \frac{3.5}{4} = 0.875$$

Восьмим етапом є визначення початкового значення коефіцієнта перерахунку (ICF). Спочатку треба проаналізувати оцінки факторів складності середовища. Після проведення аналізу отриманих оцінок, було визначено, що серед оцінок факторів E1-E6 має значення менше 3 тільки один фактор, та оцінка фактору E7 має значення 0, тому вона не підпадає під визначені вимоги. Таким чином було визначено, що IT-проєкт є простим та початкове значення коефіцієнта перерахунку (ICF) дорівнює 1.

Дев’ятим етапом є розрахунок значення коефіцієнта перерахунку (CF) за формулою 2.7. Результати розрахунків представлені нижче:

$$CF = 1 * (2 - 0.875) = 1 * 1.125 = 1.125$$

Останнім етапом є розрахунок значення оцінки трудовитрат (Effort). Після отримання загального значення скоригованих точок використання (AUCP) необхідно помножити це значення на визначений коефіцієнт перерахунку (CF) для отримання оцінки трудовитрат. Таким чином, оцінка трудовитрат буде дорівнювати 25.65 людино-годин.

Фактичний час, який був витрачений на тестування варіантів використання, що представлені в таблицях 4.1-4.5, складає 23 людино-години. Оцінка трудовитрат, отримана за оригінальним методом, дорівнює 19.39 людино-годин. Оцінка трудовитрат, отримана за модифікованим методом, дорівнює 25.65 людино-годин.

Після проведення розрахунків можна визначити, що відхилення оцінки трудовитрат за оригінальним методом від фактичних трудовитрат складає 15.7%, а точність дорівнює 84.3%. В свою чергу, відхилення оцінки трудовитрат за модифікованим методом складає 11.52%, а точність модифікованого методу дорівнює 88.48%. Вважається, що відхилення в межах 20% є прийнятним [18]. Оцінка трудовитрат за модифікованим методом має відхилення, що є в межах прийнятної норми. Це свідчить про те, що модифікований метод може бути ефективно використаний для оцінки трудовитрат на тестування.

Відхилення за оригінальним методом теж знаходиться в межах прийнятної норми, але отримана оцінка трудовитрат є меншою за фактичні трудовитрати, що є гіршим результатом ніж оцінка, яка є більшою. Занадто низькі оцінки можуть призвести до зниження якості, можливої переробки роботи на наступних етапах, і вищих ризиків провалу ІТ-проєкту [19]. Якщо оцінка трудовитрат вища за фактичні трудовитрати, це створює буфер часу для несподіваних затримок або проблем, які можуть виникнути під час ІТ-проєкту.

Підсумовуючи отримані оцінки трудовитрат за оригінальним та модифікованим методом, можна визначити, що модифікований метод точок використання дозволяє отримувати більш точні оцінки трудовитрат ніж оригінальний метод точок використання.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проведено аналіз сучасних моделей та методів оцінки трудовитрат на забезпечення якості ІТ-проєктів. Цей аналіз дозволив виявити недоліки та обмеження існуючих методів, що стали основою для подальшої модифікації. Перспективним методом для модифікації був обраний метод точок використання.

В ході роботи була проведена модифікація методу точок використання, яка включала в себе оновлення списку факторів технічної складності та складності середовища, перерахунок присвоєних значень кожного фактору за допомогою досвіду експертів галузі, створення методу визначення коефіцієнта перерахунку (CF), в якому враховуються рівень кваліфікації команди забезпечення якості (QA) та складність ІТ-проєкту. Після завершення процесу модифікації була розроблена методика застосування модифікованого методу для спрощення його використання спеціалістами у своїй роботі.

Експериментальна перевірка показала, що отримана оцінка трудовитрат на забезпечення якості в результаті використання модифікованого методу є більш точною ніж оцінка отримана за оригінальним методом точок використання. Це підтверджує ефективність проведених модифікацій і вказує на перспективність подальшого використання та розвитку цього методу.

Таким чином, мета роботи була досягнута, а поставлені задачі успішно вирішені. Результати дослідження можуть бути використані для покращення процесу оцінки трудовитрат на забезпечення якості в різних ІТ-проєктах.

Результати дослідження були представлені на конференції [20] та обговорені на Молодіжному форумі [21].

Робота була виконана відповідно до вимог методичних вказівок [22].

Для оформлення роботи використовувалися державні стандарти України [23, 24].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Якість програмного забезпечення та тестування: базовий курс. Навчальний посібник / За ред. Крепич С.Я., Співак І.Я. / для бакалаврів галузі знань 12 «Інформаційні технології» спеціальності 121 «Інженерія програмного забезпечення». – Тернопіль: ФОП Паляниця В.А., 2020. – 478с.

2. Project Costs and Risks Estimation Regarding Quality Management System Implementation. URL: https://cdn.intechopen.com/pdfs/38084/InTech-Project_costs_and_risks_estimation_regarding_quality_management_system_implementation.pdf (дата звернення 04.04.2024).

3. Bourque P., Society I. C., Fairley R. E. Guide to the Software Engineering Body of Knowledge: Version 3.0. IEEE Computer Society Press, 2014. 346 p.

4. 11 Cost Estimating Methods (With Formulas and Examples). URL: https://www.indeed.com/career-advice/career-development/cost-estimating-methods (дата звернення 10.04.2024).

5. Analogous Estimation: Definition, Uses and Examples. URL: https://www.indeed.com/career-advice/career-development/analogous-estimating (дата звернення 10.04.2024).

6. Use case point (Software Estimation Technique). URL: https://www.slideshare.net/slideshow/use-case-point-software-estimation-technique/128939552 (дата звернення 11.04.2024).

7. Three-Point Estimating: Definition, Formula and Example. URL: https://www.indeed.com/career-advice/career-development/three-point-estimating (дата звернення 12.04.2024).

8. What Is Bottom-Up Estimating? (And How It Differs from Top-Down). URL: https://www.indeed.com/career-advice/career-development/bottom-up-estimating (дата звернення 13.04.2024).

9. Putnam model. URL: https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/WIKIPEDI/W120619P.pdf (дата звернення 15.04.2024).

10. COCOMO Model – Software Engineering. URL: https://www.geeksforgeeks.org/software-engineering-cocomo-model/ (дата звернення 18.04.2024).

11. Nageswaran S. Test effort estimation using use case points. Quality Week. San Francisco. 2001. P. 22-28.

12. Software Cost Estimation With Use Case Points – Final Calculations. URL: https://tynerblain.com/blog/2007/02/19/software-cost-estimation-ucp-6/ (дата звернення 22.04.2024).

13. Estimation Techniques - Use-Case Points. URL: https://www.tutorialspoint.com/estimation_techniques/estimation_techniques_use_case_points.htm (дата звернення 23.04.2024).

14. Estimating With Use Case Points. URL: https://www.cs.cmu.edu/~jhm/DMS%202011/Presentations/Cohn%20-%20Estimating%20with%20Use%20Case%20Points_v2.pdf (дата звернення 26.04.2024).

15. Capacity of a product project team: how to calculate and what it affects. URL: https://habr.com/eu/companies/simbirsoft/articles/756626/ (дата звернення 28.04.2024).

16. Use Cases: Diagram & Examples (Updated 2024). URL: https://www.inflectra.com/Ideas/Topic/Use-Cases.aspx (дата звернення 06.05.2024).

17. Що таке веб додаток? Різниця між сайтом, веб-додатком, SPA і PWA. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення 10.05.2024).

18. A Guide to Making Estimates for Software Development Projects | Fora Soft Blog. URL: https://www.forasoft.com/blog/article/guide-to-software-estimating-95 (дата звернення 20.05.2024).

19. What we know and don't know about estimating effort in software development. URL: https://habr.com/en/articles/235271/ (дата звернення 20.05.2024).

20. Панфьорова І.Ю., Слєпцов В.А., Дослідження моделей та методів оцінки трудовитрат при керуванні якістю проєктів // Scientific progress: innovations, achievements and prospects. Proceedings of the 9th International scientific and practical conference. MDPC Publishing. Munich, Germany. 2023. P. 177-180.

21. Слєпцов В.А., Аналіз моделей та методів оцінки трудовитрат на забезпечення якості ІТ-проєктів. 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6., – Харків: ХНУРЕ. 2024. С. 261-262.

22. 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.

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

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