УДК 004.9

МЕТОД USE CASE В УПРАВЛІННІ ІТ-ПРОЄКТОМ НА ОСНОВІ МЕТОДОЛОГІЇ AGILE

Свінцицька О. М. – канд. економ. наук, доцент, доцент кафедри комп'ютерних наук, Державний університет Житомирська політехніка, м. Житомир, Україна.

Пулеко І. В. – канд. техн. наук, доцент, доцент кафедри комп'ютерної інженерії та кібербезпеки, Державний університет Житомирська політехніка, м. Житомир, Україна.

Граф М. С. – доктор філософії, завідувачка кафедри комп'ютерних наук, Державний університет Житомирська політехніка, м. Житомир, Україна.

Петросян Р. В. – старший викладач кафедри комп'ютерних наук, Державний університет Житомирська політехніка, м. Житомир, Україна.

АНОТАЦІЯ

Актуальність. Розглянуто роль та процес формування вимог користувача на основі методу Use Case в оцінці трудомісткості Agile проєкту на етапі попередньої оцінки менеджментом компанії. Вже з середини 70-х років відомо, що помилки у вимогах є найчисленнішими, найдорожчими та трудомісткими для виправлення в проєктах. У зв'язку з цим, підвищується ступінь важливості управління вимогами в ІТ-проєктах з використанням сучасних технології та методів їх формування та оцінки.

Мета. Формування та оцінка вимог користувача в управлінні ІТ-проєктами на основі методу Use Case та їх вплив на один з показників ефективності проєкту на етапі планування, зокрема трудомісткість.

Метод. В статті запропоновано новий авторський підхід до формування та оцінки вимог користувача в Agile-проєктах з урахуванням впливу ризиків та оцінки складності системи на основі методу Use Case, і як результат дослідження та пропозиції по досягненню поставленої мети, запропоновано математичну модель оцінки трудомісткості проєкту. Математичний шаблон моделі дозволяє враховувати додаткові змінні, які можуть впливати на проєкт, зокрема кількість рівнів користувачів, наявний функціонал та ризики технічного та організаційного характеру, є гнучкою і може адаптуватися до різних потреб конкретного проекту, що відповідає принципам методології Agile. Кількість компонентів у формулі можуть бути змінені, щоб врахувати важливість різних змінних, або розширена, щоб враховувати додаткові змінні, які можуть впливати на проєкт.

Результати. Розроблена та апробована на прикладі мобільного додатку математична модель оцінки трудомісткості проєкту на основі методу use case, що містить набір вихідних даних для розробки продукту та обмежень щодо зміни вимог користувачів та організаційно-технічних ризиків. Запропонована математична модель дозволяє швидко, точно та ефективно визначати сценарії трудомісткості витрат проєкту різного типу та рівня складності та може слугувати ефективним інструментом прийняття управлінських рішень.

Висновки. Загальні висновки, отримані після аналізу методів формування та оцінки вимог користувача в практиці Agile управління, наступні. На етапі планування робіт базову модель оцінки проекту, яка базується на експертній оцінці кожної функціональної вимоги, замінено на більш сучасну та складну, яка базується на методі use case та враховує зміни вимог користувачів та інші ризики розробки продукту. Для складання нової моделі використовуються графічні, аналітичні та математичні інструменти, зокрема, діаграма use case, коригуючі коефіцієнти, що враховують складність актора та use case, коефіцієнти, що враховують організаційний та технічний ризики, і як результат, отримуємо математичний формат розрахунку трудомісткості проєкту. Такий підхід дозволяє швидко адаптуватися до різних типів проектів, а за умови правильного визначення вихідних даних, модель дозволяє отримати досить точні оцінки на ранньому етапі планування проєкту. Практично отримані результати дослідження демонструють потенціал запропонованої математичної моделі, яка може мати логічне продовження через верифікацію моделі на більшій вибірці та оцінку її стійкості до різних типів проектів і ризиків.

КЛЮЧОВІ СЛОВА: вимоги користувачів, ІТ-продукт, Agile-проєкт, User Story, Use Case, мобільний додаток.

АБРЕВІАТУРИ

API – Application Programming Interface;
ACF – Actor Complexity Factor;
BA – Business analysis;
CR – Communication and Risk;
Dev – Development;
GUI – Graphical User Interface;
IT – інформаційні технології;
MVP – minimum viable product;
OF – Organizational Factor;
OCF – Organizational Complexity Factor;
PM – Project management;
TF – Technical Factor;
TCF – Technical Complexity Factor;
QA – Quality Assurance;
UI – User Interface;
UML – Unified Modeling Language;
UCCF – Use Case Complexity Factor.

НОМЕНКЛАТУРА

E – множина всіх етапів проекту;
F – множина фіч;
G – показник ризику;
i – етапи роботи над проектом;
J – множина use case;
Ki – коефіцієнт складності етапу;

DOI 10.15588/1607-3274-2025-1-17

K_{ACF} – середня складність актора;

K_{UCCF} – середня складність use case;

K_{TCF} – показник технічної складності проєкту;

K_{OCF} – показник організаційної складності проєкту;

PF – продуктивність розробки програмного забезпечення на j-му use case;

c_i – кількість акторів для i-го use case;

c_i – кількість use case для j-го актора;

T – загальна трудомісткість проекту;

$Ti(f)$ – час, необхідний для розробки фічі f на етапі i роботи над проектом;

X – множина акторів.

ВСТУП

Проєкти програмного забезпечення мають дуже високу ймовірність провалу, і основною причиною цього є поганий процес розробки та оцінки вимог на етапі планування. В Agile методології формування вимог є основним і найважливішим етапом життєвого циклу розробки програмного забезпечення [3]. А помилка в оцінці вимог може збільшити витрати та час виконання та призвести до збою проєкту програмного забезпечення. Подолання цих недоліків потребує застосування нового підходу до вибору методів збору, формування та оцінки вимог користувача. Незважаючи на широкий спектр існуючих методів, таких як інтерв'ю, анкетування, прототипування, використання прецедентів та ін., сучасні IT-проекти вимагають більш гнучких та ефективних підходів. Саме тому набули популярності такі методи, як сценарії користування (use case) та історії користувачів (user story), які дозволяють більш гнучко описувати функціональність системи та легко адаптувати її до змінних вимог. В статті проаналізовано їх переваги не недоліки, здійснено вибір методу use case для побудови математичної моделі прогнозування оцінки трудомісткості проєкту на основі вимог користувача, яку практично апробовано на проєкті мобільного додатку.

Мета дослідження. Моделювання трудомісткості IT-проєкту на основі процесу формування та оцінки вимог користувача методом Use Case.

Об'єкт дослідження є процес побудови математичної моделі оцінки трудомісткості Agile проєкту, що враховує вимоги користувача IT-продукту, ризик розробки та інструменти методу Use Case.

Предмет дослідження. Поєднання традиційних методів формування та оцінки вимог користувача з інноваційними, які побудовані на основі використання природної мови, концептуальних моделей, які допомагають описати математично та візуалізувати процеси в Agile-проєктах, а також знаходять своє відображення в показниках трудомісткості.

1 ПОСТАНОВКА ПРОБЛЕМИ

Розробка та оцінка вимог є ключовою та центральною для кожного успішного проєкту. Є кілька причин, чому IT-проєкти зазнаються невдач, серед них є причини, які пов'язані з вимогами, зокрема, методами їх збору, оцінки та документування. Існуючі методи часто мають ряд обмежень, таких як суб'єктивність, залежність від експертних оцінок та недостатня гнучкість для адаптації до змінних вимог користувача. Тому виникає потреба у розробці нових підходів до оцінки проєктів, які б дозволили забезпечити більш точне визначення трудомісткості проекту на ранніх етапах, при цьому враховували фактори ризику та зміну вимог користувача. Рішення даної проблеми може полягати у створенні моделі, яка б дозволила адаптувати до таких умов ранню оцінку тривалості та вартості проєкту, а також мінімізувати вплив людського фактору на точність оцінки, використовуючи більш об'єктивні методи розрахунку.

Одним із перспективних підходів до вирішення цієї проблеми є застосування методу use case як основи для побудови математичної моделі оцінки трудомісткості проєкту. Це дозволить врахувати складність кожного use case, його взаємозв'язки з акторами, а також врахувати відповідні ризики, пов'язані зі зміною вимог користувача. Така модель дозволить створити більш точну, гнучку та об'єктивну систему оцінки проєктів на ранньому етапі реалізації.

Для прийняття рішення про застосування даного підходу доцільно провести дослідження на основі порівняння двох варіантів оцінки проєкту. Перший – традиційний, шляхом прямого підрахунку витрат годин на розробку проекту та застосування до них коригуючих коефіцієнтів, що сформовані експертним шляхом на основі попереднього досвіду. Задача тут полягає у розрахунку загальної тривалості проекту на основі показника трудомісткості T, враховуючи набір фіч F та їхні характеристики, що сформовані на основі вимог користувача, які враховують етапи розробки проєкту, i (UI, Dev, QA, PM, BA, CR) та відповідний коефіцієнт їх складності, Ki. Для цього пропонується формалізувати цей варіант оцінки через шаблон математичної моделі. Тоді, математична модель визначення загальної тривалості проекту T, матиме такий вигляд (формула (1)):

$$T = \sum(f \in F)\sum(i \in E)\cdot Ti(f)\cdot Ki\,, \qquad (1)$$

де $F = \{f_1, f_2, ..., f_m\}$.

Поряд цим, такий формат розрахунку є достатньо спрощеним, має ряд обмежень, таких як: суб'єктивність та залежність від експертних оцінок, висока трудомісткість самих оціночних робіт, відсутність гнучкості в залежності від змін впливу зовнішнього контексту. Застосування методу use case дозволить створити нову математичну модель, яка враховуватиме такі фактори, як: складність use case, ризики проекту та продуктивність розробки.

DOI 10.15588/1607-3274-2025-1-17

Отже, постановка задачі полягає у створенні математичної моделі для прогнозування оцінки тривалості проєкту T^* на ранніх етапах розвитку, з урахуванням складності акторів, K_{ACF} та складності use case, K_{UCCF}, продуктивності розробники, PF та рівня ризику проекту, G. При цьому, задано:

$X = \{x_1, x_2, ..., x_n\}$;

$J = \{j_1, j_2, ..., j_m\}$.

В якості критерію оцінки визначено загальну трудомісткості проекту при дотриманні заданих обмежень:

– технічних (наприклад, вибір технології, платформи);

– організаційних (наприклад, стандарти якості, досвід команди);

– зміни вимог користувачів (наприклад, зміна функціональності).

Використання методу use case в якості основи для математичної моделі дозволить досягти більш точної оцінки та прогнозів тривалості проекту порівняно з традиційними методами.

2 ОГЛЯД ЛІТЕРАТУРИ

Дослідники зробили великий внесок у сферу формування та оцінки вимог в Agile-проєктах, але ступінь, до якого їхні пропозиції були прийняті на практиці, залишається недостатньо розкритим. Сучасні дослідження показують, що 71% програмних проєктів зазнають невдачі через погане управління та неправильну оцінку вимог на етапі планування. Доповідь Standish CHAOS [1], яка досліджувала 9236 IT-проєктів, виявила, що трьома основними причинами невдач проєктів були: відсутність введення даних користувачами, неповні вимоги або зміни вимог. Крім того, дослідження свідчать, що [2]: деяких вимог не вистачає в кінці ітерації, а деякі вимоги були неоднозначними та непослідовними, що з часом потребувало додаткової роботи над вимогами; деякі вимоги втрачаються або недостатньо структуровані чи мають незрозуміле походження, що призводить до втрати їх розуміння.

Методологія Agile була призначена головним чином для того, щоб допомогти розробникам створити проєкт, який може швидко адаптуватися до перетворення запитів і підходить для проєктів з високим ступенем невизначеності. Agile – продукт будується інкрементально, тобто замість початкового жорсткого планування створення і випуску всього продукту, одразу невеликі цінні інкременти продукту плануються та випускаються поступово [13].

Розглянемо публікації та основні нормативні документи, що визначають основні процеси, пов’язані з формуванням та визначенням вимог користувачів. Стандарт IEEE Std 610.12-1990 містить глосарій термінів, пов’язаних із програмною інженерією, відповідно до якого вимога – це «умова або здатність, необхідна користувачеві для вирішення проблеми або досягнення об’єктивних цілей» [4]. Поряд з цим, вітчизняні автори публікацій [5, 6] доповнюють це визначення можливостями, якими повинна володіти система, щоб виконати контракт або задовільнити стандартам, специфікаціям або іншим формальним документам [5]. Інший підхід більш узагальнений. Це набір потреб потенційних користувачів щодо властивостей, якостей та функцій програмного продукту, який потрібно розробити або модифікувати [6].

Стандарт ISO/IEC 25062 [7] прямо не згадує вимоги, але неявно забезпечує основу для визначення вимог щодо зручності їх використання. Стандарт ISO 9241-210:2019 спрямований на визначення процесів для визначення вимог до взаємодії користувача з інформаційно-комунікаційними технологіями. Він надає загальні рекомендації щодо того, як слід збирати, аналізувати та документувати вимоги користувачів під час розробки систем, що включають в себе інтерфейси користувача [8]. Стандарт ISO/IEC/IEEE 29148 містить різні аспекти збору, документування, аналізу, перевірки та управління вимогами, включаючи вимоги користувачів, надає загальні вказівки для їх управління та обробки [9].

Авторами в наукових виданнях публікуються різні підходи опису процесів і методів, які можна використовувати для збору інформації про користувачів та їхні завдання [10, 11, 12]. Деякі автори чітко розмежовують потреби користувача та вимоги користувача [11]. Вимоги користувачів зазвичай включаються як частина вимог зацікавлених сторін документ специфікації, як описано в ISO/IEC/IEEE 29148. В авторській статті [2] наводяться практичні результати досліджень щодо структурування вимог за різними ознаками в процесі розробки проєктів. В Agile-проєктах зачасту формування вимог здійснюється двома основними способами – з використанням природної мови, тобто мови, якою користувачі або стейкхолдери проєкту спілкуються (наприклад user story та use case) та з використанням концептуальних моделей (model based – Use Case Diagrams, Class Diagrams, Activity Diagrams, State Diagrams). Найбільш широке поширення отримав перший підхід, як найбільш гнучкий, а другий носить допоміжний характер. Поряд з цим використання графічних елементів є широко поширеним в практиці формування вимог. Основною метою використання графічних елементів є надання додаткової інформації у вигляді анотацій або допомогти візуалізувати вимоги. Вони є більш наочною формою бачення вимог. Вибір методу залежить від типу проєкту, специфіки бізнесу та вподобань команди. Найчастіше використовують комбінацію різних методів для найкращого розуміння та документування вимог стейкхолдерів [2].

З розвитком продуктів та інформаційних технологій вимоги користувачів стають дедалі складнішими, і в результаті генеруються більш сучасні методи отримання вимог, за допомогою яких вимоги стають вимірними, тестованими, пов’язаними

DOI 10.15588/1607-3274-2025-1-17

з бізнес-потребами, і описаними з рівнем деталізації достатнім для конструювання системи.

Отже, одна з проблем в управлінні Agile-проєктами – це відсутність загальноприйнятих термінів та підходів, якими користуються фахівці для опису вимог користувача та їх впливу на оцінку трудомісткості проєктів. У різних джерелах проблему пропонується вирішувати, виходячи з різних умов, що склалися в менеджменті проєкту та мало уваги приділяється ролі управління вимогами в цих процесах. З огляду на розбіжність робимо висновок про важливість вибору та обґрунтування підходів та методів до формування та аналізу вимог при розробці IT-проєктів на основі Agile.

3 МАТЕРІАЛИ І МЕТОДИ

В цьому розділі статті автором наводиться опис запропонованої моделі оцінки трудомісткості Agile проєкту на основі методу use case. Варто відзначити, що формування вимог – це постійний процес, який починається на етапі пресейлу (попередньої оцінки) і триває протягом усього життєвого циклу проекту. На кожному етапі вимоги уточнюються, деталізуються і можуть змінюватися, використовуються різні інструменти та методи. Тому, автором проводиться дослідження та наводяться пропозиції щодо вирішення проблеми формування та оцінки вимог в управління IT-проєктом методом use case на етапі попередньої оцінки проєкту за візією замовника. На цьому етапі головною метою є отримати первинне уявлення про проєкт, визначити його масштаб, складність та потенційну вартість за різними складовими з можливістю гнучкої адаптації проєкту до змінних умов замовника. Зазвичай це високоривневі, неформалізовані вимоги, які замовник висловлює у загальних рисах. Вони слугують для створення першої оцінки проекту та визначення подальших дій. Запропонована автором модель дозволяє вирішити поставлене завдання. В нашому дослідженні така модель буде мати математичну формалізовану складову, що забезпечить досягнення поставленої задачі, тобто створити модель оцінки тривалості Agile-проекту в годинах на етапі попередньої оцінки з метою підвищити її точність та забезпечити більш гнучке управління проектом у відповідності до змінних вимог користувача та впливу організаційно-технічних ризиків.

Розглянувши існуючі методи формування та опису вимог будемо поєднувати графічні, аналітичні та математичні інструменти для побудови моделі оцінки трудомісткості проекту. На основі діаграми use case, яка використана в даному дослідженні для візуалізації бізнес-вимог проєкту мобільного додатку відеоредактора, опишемо етапи побудови моделі.

На першому етапі автором пропонується застосувати коригуючі коефіцієнти до заданої продуктивності розробки для *i*-го use case, що враховують складність кожного актора та відповідного варіанта використання (use case), що відображені на діаграмі (рис.3) з метою визначення рівня складності системи. Для цього розподілимо акторів та use case на прості, середні і складні за критеріями, представленими в табл.2 та табл.3.

Таблиця 2 – Критерії розподілу акторів на прості, середні і складні

Класифікація актора	Характеристика актора
Simple	Зовнішня система, яка взаємодіє із системою, яка використовує чітко визначений API
Average	Зовнішня система, яка взаємодіє із системою, що використовує стандарт протоколів зв'язку (напр.TCP/IP, HTTP, база даних)
Complex	Людина-актор, що використовує GUI інтерфейс програми

З вище представлених характеристик акторів, нами зроблено такі висновки щодо ідентифікації акторів на прості чи складні:

– якщо актор взаємодіє з декількома варіантами використання без додаткових умов, то він, швидше за все, є простим;

– якщо взаємодія між актором і системою є складною, в нашому прикладі це додаткові дії, які повинен виконати актор, щоб забезпечити реалізації варіанту використання, то актор, швидше за все, є складним;

– важливість актора для системи. якщо актор є важливим для системи, то він, швидше за все, є складним.

Наступним етапом побудови моделі є розрахунок *ACF*, який визначається як сума добутків кількості акторів за категорією складності та його ваги. Отримане середнє значення з цього числа на одного актора і буде K_{ACF}.

Далі аналізуємо систему за критерієм кількості простих, середніх та складних варіантів використання (табл. 3).

Таблиця 3 – Класифікація use case на прості, середні та складні

Класифікація use case	Межі use case
Simple	межі 1–3
Average	межі 4–7
Complex	8 і більше

На основі отриманих даних розраховуємо коефіцієнт середньої складності use case за даними *UCCF* та відповідний K_{UCCF}, які визначаються аналогічно *ACF* та K_{ACF} відповідно.

Після аналізу складності системи в моделі автором пропонується використати показник впливу організаційно-технічних ризиків, *G*. Отже, наступним етапом побудови моделі є опис його розрахунку.

На цьому етапі формується набір субфакторів та їх відносна важливість в розробці проєкту. Окремо формується набір технічних та організаційних субфакторів. Команда експертів присвоює значення та вагу кожному з субфакторів, виходячи зі складності проекту, зокрема: значення 0 – якщо субфактор не має

DOI 10.15588/1607-3274-2025-1-17

значення, 3 – якщо субфактор має середній вплив та 5 – якщо субфактор моє дуже великий вплив на проект. На основі сформованого набору із 13 технічних субфакторів та їх ваги визначається K_{TCF}. Це дозволяє оцінити складність розробки та впровадження програмного забезпечення. Чим вище значення K_{TCF}, тим складнішим вважається проект. Формула для розрахунку:

$$k_{TCF} = 0{,}6 + \frac{TF}{100}, \quad (2)$$

де TF – це індекс, який враховує конкретні технічні аспекти проекту. Його значення, обчислюється на основі інших параметрів, таких як кількість функцій, складність алгоритмів, використання нових технологій тощо.

0,6 – це константа, яка додає базовий рівень складності до проекту. Значення константи 0,6 відображає мінімальний рівень складності, який присутній у будь-якому проекті, а також дозволяє більш точно порівнювати різні проекти за їхньою складністю.

Оцінка організаційних ризиків має аналогічний підхід до розрахунку, як і для TCF. Показник організаційної складності проєкту K_{OCF} враховує різноманітні фактори, які можуть вплинути на загальну ефективність проекту, наприклад, досвід команди, технологічна складність проекту, розмір проекту тощо. Для цього враховано 8 субфакторів, кожному з яких визначено вагу у загальному їх значенні для розробки проекту. Формула для розрахунку матиме вигляд:

$$k_{OCF} = 1{,}4 + (-0{,}03 \cdot OF), \quad (3)$$

де 1,4 – це константа, яка додає базовий рівень складності до проєкту. Навіть якщо всі інші фактори, що впливають на проект, є ідеальними, завжди існує певний рівень складності, пов'язаний з організацією роботи, комунікацією, управлінням проектом тощо.

–0,03 – це коефіцієнт, який показує, як сильно фактор OF впливає на загальну складність. Мінус перед числом означає, що зі збільшенням OF значення K_{OCF} зменшується. Іншими словами, чим вище значення OF, тим менш складним вважається середовище проекту.

Визначивши фактори впливу на загальний рівень ризику проекту та згрупувавши їх дві категорії отримаємо загальний показник впливу організаційно-технічних ризиків G, що розраховується за формулою:

$$G = K_{TCF} \cdot K_{OCF}, \quad (4)$$

Розрахувавши коефіцієнти, що пропонується використовувати для коригування продуктивності розробки системи для i-го use case, переходимо до наступного етапу розробки моделі – визначення трудомісткості проєкту, T^*, що завершує процес побудови загальної моделі. В нашому випадку це формалізований шаблон математичної моделі, яка дозволяє розрахувати загальну тривалість проекту у годинах, T^* (формула 5):

$$T^* = \sum (c_i \cdot PF_i \cdot k_{ACF}) + (c_j \cdot PF_i \cdot k_{UCCF}) + (c_j \cdot PF_i \cdot G), \quad (5)$$

де c_i – кількість акторів для i-го use case за даними рисунку 3; PF_i – продуктивність розробки для i-го use case, що задано на рівні 40 годин на розробку, як базовий рівень для даної команди на основі пjпереднього досвіду роботи.

Запропоновані коефіцієнти та математичний шаблон розрахунку трудомісткості проєкту на основі методу use case, може використовуватись до різних проектів з метою побудови різних сценаріїв розвитку проекту при змінних вимогах користувача до функціоналу та існуючих ризиків.

Отже, наведені автором етапи демонструють детальний підхід до оцінки трудомісткості проекту на основі методу Use Case на етапі попередньої оцінки проєкту для керівників та бізнес-аналітиків IT-проєкту. Модель враховує зміну вимог користувачів, що виражена через коефіцієнти складності актора та use case, а також як технічні аспекти проекту (складність функціоналу, технології), так і організаційні фактори (досвід команди, стабільність вимог), які закладені у відповідному коефіцієнті.

Запропонована математична модель дозволяє об'єднати всі отримані дані та розрахувати загальну трудомісткість проекту. При зміні вхідних параметрів, оцінюваний показник трудомісткості автоматично змінює своє значення. Такий підхід дозволяє швидко адаптуватися до різних типів проектів, а за умови правильного визначення вхідних даних, модель дозволяє отримати досить точні оцінки на ранньому етапі планування проєкту.

4 ЕКСПЕРИМЕНТИ

Для вирішення поставленої задачі, в даній частині роботи, застосуємо набір вихідних даних для побудови запропонованої математичної моделі та опишемо послідовність етапів робіт для отримання результатів від її застосування в умовах Agile проєкту створення мобільного додатку-відеоредактора, що базується на концепції методу use case.

В даному дослідженні для прикладу обрано IT-продукт, що забезпечує покращення конкурентних позицій замовника, який бажає створити мобільний додаток-відеоредактор для молодих талантів. Аутсорсингова компанія отримала Vision проєкту від замовника в якості бізнес-вимог до продукту. Vision проєкту – це велика ідея, яку замовник реалізує через продукт. У ньому містяться основні тези про продукт:

DOI 10.15588/1607-3274-2025-1-17

яким він буде, яку мету переслідує бізнес, хто користувач. У Vision від замовника продукту Filmy є інформація про ринок, користувачів, монетизацію та ідейні цілі Filmy. Також у документі перераховані фічі, які необхідно реалізувати. На основі цього документу буде тривати підготовка до написання вимог різних рівнів. Відповідно до Vision проєкту проджект з бізнес-аналітиком готують опис користувацьких вимог для подальшої роботи в Confluence. Перша версія продукту дозволить користувачам створювати відео, відео дуети та ділитися ними онлайн. Далі планується розширювати функціональність за рахунок великої бібліотеки відео та аудіо ефектів. Однак, оскільки даний проект має за мету подальшу комерціалізацію і у замовника відсутні чіткі цілі подальшого розвитку функціоналу, що робить задачу достатньо ризикованою для реалізації та унеможливлює чітке планування витрат на проект, в тому числі трудових і фінансових. В цих умовах робота над проектом лягає в площину концепції управління проектами Agile у поєднанні із сучасними інструментами моделювання вимог користування та їх оцінки на основі методу use case.

Для отримання вихідних даних, їх зберігання та використання використовувався пакет програмного забезпечення від компанії Atlassian, зокрема, це Jira і Confluence. На рисунку 1 відображено основні продукти Atlassian, які застосовуються в процесі розробки проєкту програмного забезпечення.

Рисунок 1 – Інструменти Atlassian в процесі розробки IT-проєкту

Jira – інструмент управління проектами, який дозволяє стежити за завданнями, багами та планувати релізи програмного забезпечення. В Atlassian Marketplace є багато додатків для інтеграцій, які вирішують супутні задачі по управлінню проєкту. В нашому дослідженні використовувалися плагіни для командної роботи щодо оцінки тривалості задач по проєкту за традиційним підходом: Agile Poker for Jira – Planning & Estimation (рис. 2) та Cost Monitoring for Jira (рис. 3).

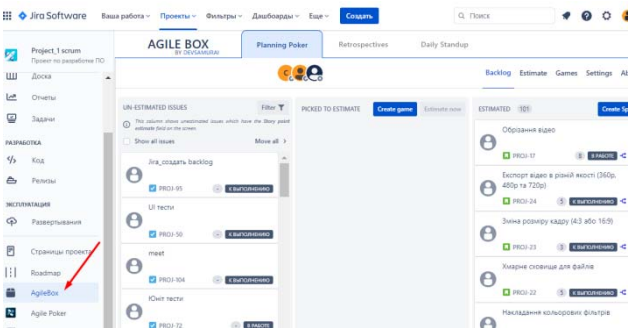

Рисунок 2 – Плагін Agile Poker for Jira – Planning & Estimation

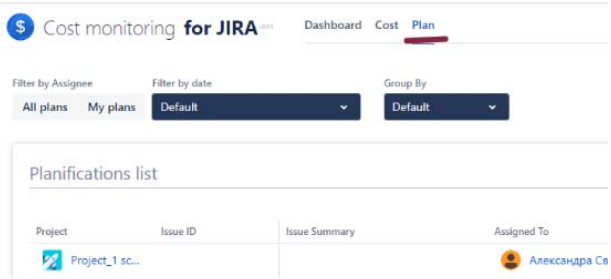

Рисунок 3 – Плагін Cost Monitoring for Jira

Для збереження даних та спільної роботи над проєктом використовувався інструмент Confluence. Тут накопичені командою знання поєднані із можливостями для спільної роботи.

Для реалізації моделі та обробки даних використовувався пакет аналітичних функції Excel. Для моделювання та візуалізації вимог стейкхолдерів з точки зору користувачів та їх взаємодії з системою використовувася метод Use Case Diagram у графічному інструменті draw.io.

Use Case Diagram для мобільного додатку відеоредактора Filmy відображена на рис. 3.

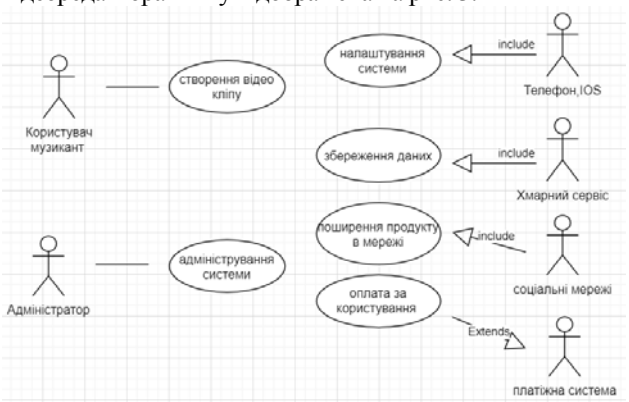

Рисунок 3 – Use Case Diagram для мобільного додатку відео редактора Filmy

В нашому дослідженні візуалізація допомагає оцінити складність системи, з точки зору взаємодії користувачів із системою. Класифікація акторів додатку на рівні складності наведена в таблиці 4.

Таблиця 4 – Класифікація акторів на прості, середні та складні

Складність актора	Актори	К-ть акторів	Вага	Складність актора
Simple	X3, X4	2	1	2
Average	X2, X5	2	2	4
Complex	X1, X6	2	3	6

На основі отриманих даних розраховуємо K_{ACF}, який становить в нашому випадку 2,0.

В нашій моделі також передбачено аналіз системи за рівнем складності. В таблиці 4 наведено характеристику системи за критерієм кількості простих, середніх та складних варіантів використання (табл. 5).

DOI 10.15588/1607-3274-2025-1-17

Таблиця 5 – Розподіл use case на прості, середні та складні

Класифікація use case	Межі use case	Актори	К-ть use case	Вага	Складність use case
Simple	межі 1–3	X2, X3, X4, X6	4	0,15	0,6
Average	межі 4–7	X5	4	0,35	1,4
Complex	8 і більше	X1	8	0,5	4

Отже, в нашому прикладі, складний актор – музикант, який взаємодіє з більш ніж 7-ма варіантами використання. Телефон та адміністратора відносимо до середніх акторів, які взаємодіють з 4-ма варіантами використання, що можна побачити на рисунку 2 (Use Case Diagram). Маркетолог, банк, хмарний сервіс і соціальні мережі визначено як прості актори, оскільки вони взаємодіють з одним варіантом використання. Варто відмітити, що представлена класифікація демонструє зв'язки системи на вищому рівні з її зовнішніми атрибутами. Кожен елемент системи має свій набір складових, який деталізує та описує вимоги для потреб наступного рівня. Наприклад, банк як простий актор бере участь у двох варіантах використання: «Оплатити користування додатком» та «Прийняти платіж за користування додатком». В свою чергу варіант використання «Оплатити користування додатком» передбачає такі дії банку:

– прийняття запиту на оплату від додатку;

– перевірка запиту на оплату на наявність помилок;

– обробка запиту на оплату;

– надання додатку інформації про успіх або невдачу оплати.

Варіант використання «Прийняти платіж за користування додатком» для банку. як актора, передбачає такі дії банку:

– прийняття платежу від користувача;

– перевірка платежу на наявність помилок;

– зарахування платежу на рахунок додатку.

Представлена класифікація акторів та use case на прості, середні та складні є досить умовною, оскільки кількість варіантів використання та акторів може змінюватися в залежності від конкретних вимог до додатку, з якими взаємодіє користувач. Наявність простих, середніх та складних зв'язків в мобільному додатку Filmy дає нам підстави зробити висновок, що аналізована система відноситься до середнього рівня складності, оскільки:

– у додатку є шість акторів, з яких чотири є складними. Це означає, що додаток повинен підтримувати взаємодію з широким спектром зовнішніх систем;

– у додатку є 16 варіантів використання, з яких 8 є складними. Це означає, що додаток повинен підтримувати широкий спектр функцій;

– додаток повинен підтримувати такі технології, як API, протоколи зв'язку та GUI. Це вимагає від додатку наявності значної функціональності та складності.

На основі отриманих даних в таблиці 4 розраховуємо K_{UCCF}, що складе 2,08.

Для розрахунку показника G з нашого математичного шаблону моделі, побудуємо відповідні таблиці (таблиця 6 та 7), в яких наведено дані щодо переліку факторів ризику, експертної оцінки впливу організаційно-технічних ризиків на їх ваги у проекті.

Таблиця 6 – Показники технічної складності проекту

Фактор	Опис	Вага	Оцінка	*TF*
T1	Розподілена система	2.0	5	10
T2	Вимоги до швидкодії та часу відгуку	1.0	5	5
T3	Зручність використання кінцевим користувачем	1.0	3	3
T4	Внутрішня обчислювальна складність	1.0	2	2
T5	Повторне використання коду	1.0	3	3
T6	Легкість інсталяції	0.5	1	0.5
T7	Легкість використання	0.5	5	2.5
T8	Портативність на інші платформи	2.0	2	4
T9	Обслуговування системи	1.0	2	2
T10	Паралельна обробка	1.0	3	3
T11	Функції безпеки	1.0	5	5
T12	Доступ для третіх сторін	1.0	1	1
T13	Навчання кінцевих користувачів	1.0	1	1
				Загальний TF: 42

На основі даних таблиці розраховуємо *TF*, що складає для нашого проєкту 42, а далі виконуючи умову формули 2, отримуємо K_{TCF} на рівні 1.02.

Для визначення відповідного коефіцієнті на основі факторів організаційної складності використаємо дані таблиці 8.

Таблиця 8 – Показники організаційної складності проекту

Фактор	Опис	Вага	Оцінка	*OF*
E1	Знайомство з використовуваним процесом розробки	1.5	3	4.5
E2	Досвід у розробці подібних додатків	0.5	3	1.5
E3	Досвід команди в об'єктно-орієнтованій розробці	1.0	2	2.0
E4	Компетентність провідного аналітика	0.5	5	2.5
E5	Мотивація команди	1.0	2	2.0
E6	Стабільність вимог	2.0	1	2.0
E7	Залучення співробітників на неповну зайнятість	–1.0	0	0.0
E8	Складність мови програмування	–1.0	4	–4.0
				Загальний OF: 10.5

DOI 10.15588/1607-3274-2025-1-17

В нашому випадку *OF* складає 10,5, а далі виконуючи умову формули 3, отримуємо K_{OCF} на рівні 1,085.

Підставляючи всі розрахункові величини у нашу формулу 5 отримаємо значення *T** на рівні 2011,22 год.

Результати моделювання показали, що запропонована модель дозволяє оцінити трудомісткість проекту з більшою точністю порівняно з традиційним підходом.

Цей приклад є лише базовим шаблоном, який може бути адаптований під інший тип проєкту з можливістю зміни вихідних даних в процесі оціни трудомісткості проєкту. Оптимальний вибір підходу залежить від конкретної ситуації. Для невеликих проектів з простими вимогами може бути достатньо традиційного підходу. Для великих і складних проектів більш доцільно використовувати запропоновану математичну модель. Можна комбінувати обидва підходи.

Застосування цих методів і підходів допоможе керівникам та командам розробки IT-проєктів формувати вимоги, які відповідають потребам користувачів і можуть бути успішно реалізовані під час формування трудомісткості проектів та складання бюджетів вищого рівня.

5 РЕЗУЛЬТАТИ

Для демонстрації ефективності запропонованої моделі було проведено дослідження на прикладі розробки мобільного додатку відеоредактора Filmy. Маючи набір вихідних змінних по проєкту та критерій оцінки, в статті автором було проведено порівняння двох варіантів оцінки трудомісткості робіт. Перший – традиційний, шляхом прямого підрахунку витрат годин на розробку від команди проекту та застосування до них коригуючих коефіцієнтів, що сформовані експертним шляхом на основі попереднього досвіду, *T* (формула 1). Другий – на основі математичної моделі, яка виражена формулою 5. Відобразимо та проаналізуємо отримані результати в обох варіантах (табл. 9) та проведемо обгрунтування доцільності застосування запропонованої модулі в системі управління IT-проєктами на етапі.

В таблиці 10 наведені основні нормативи розподілу годин трудомісткості між етапами розробки IT-проєкту, які використовувались для визначення загального значення показника *Ti*(*f*).

Наведені в таблиці 9 та 10 дані є достатньо зрозумілими для аналізу та визначення трудомісткості проєкту. Для розрахунку нормативи визначені експертним шляхом на основі досвіду роботи команд над іншими проектами. Однак така простота криє за собою ряд недоліків, зокрема відволікання розробників від безпосередньої їх роботи для проведення оцінки трудомісткості фіч, суб'єктивізм в оцінці, що випливає з набутого досвіду фахівців та неможливості врахувати всі залежності в проекті та передбачити вплив такого фактору як зміна вимог користувача. Усунути зазначені недоліки допоможе запропонована в цій статті математична модель, яка є більш складною і сучасною, оскільки базується на методах use case та враховує зміни вимог користувачів та інші ризики розробки продукту. Використовуючи формулу (5) отримаємо такі результати (табл. 11).

Таблиця 9 – Трудомісткість робіт, *Ti*(*f*) за стадіями розробки мобільного додатку відеоредактора Filmy (традиційний підхід)

Фіча, *F*	*Ti*(*f*), год.	UI	Dev	QA	PM, BA	CR
Обрізка відео	273,6	32	120	45,6	22,8	53,2
Додавання відео із галереї	172,8	16	80	28,8	14,4	33,6
Додавання музики до відео	273,6	32	120	45,6	22,8	53,2
Додавання нового проекту	100,8	16	40	16,8	8,4	19,6
Додавання тексту	151,2	24	60	25,2	12,6	29,4
Зміна розміру кадру (4:3 або 16:9)	172,8	16	80	28,8	14,4	33,6
Експорт в різній якості	122,4	8	60	20,4	10,2	23,8
Публікація відео в соціальні мережі	136,8	16	60	22,8	11,4	26,6
Адміністрування системи	453,6	32	220	75,6	37,8	88,2
Хмарне сховище для файлів	151,2	24	60	25,2	12,6	29,4
Платіжна система	28,8		16	4,8	2,4	5,6
Налаштування телефону	356,4	48	150	59,4	29,7	69,3
Всього, *T*	2394	264	1066	399	199,5	465,5

Таблиця 10 – Нормативу розподілу трудомісткості за етапами розробки проєкту

Етап	% від часу на розробку
Стабілізація системи	0,25
Тестування системи	0,3
Час менеджера	0,15
Комунікація в команді	0,1
Поставка готового коду	пів дня – день
Ризики	0,2

На основі математичних розрахунків отримали результати, які відображені в таблиці 11. Варто відзначити, що всі зазначені коефіцієнти вже були попередньо розраховані та усувають існуючі недоліки в традиційному підході оцінки трудомісткості проєкту. Зокрема, K_{ACI} та K_{UCCF} – це коефіцієнти для опису функціональності системи, які враховують фактори зміни кількості акторів та use case та є змінними для *T**, *G* – комплексний показник, що враховує вплив організаційних та технічних факторів ризику на проект. В основу розрахунку покладено базовий показник продуктивності розробки на один

DOI 10.15588/1607-3274-2025-1-17

use case, що в середньому становить 40 годин, значення якого враховує досвід попередніх проектів. Вибір такої величини для розробки є досить умовним і залежить від багатьох факторів, зокрема досвіду роботи та структури команди. Він слугує скоріше відправною точкою для подальших розрахунків, ніж точним і універсальним показником. В нашому випадку ця величина є також змінною, яка може варіюватися від проєкту до проєкту.

Таблиця 11 – Показники результативності використання моделі оцінки трудомісткості проєкту на основі методу Use Case, T^*

Актори, X	к-ть акторів, c_i	К-ть use case, c_j	Середня продуктивність на 1 use case, PF_i	PF з урахуванням K_{ACI}	PF з урахуванням K_{UCCF}	PF з урахуванням, G	Загальний час на проєкт, T^*
X1	1	7	40	80,00	583,3	309,9	1013,2
X2	1	1	40	80,00	83,33	44,27	247,6
X3	1	1	40	80,00	83,33	44,27	247,6
X4	1	1	40	80,00	83,33	44,27	247,6
X5	1	1	40	80,00	83,33	44,27	247,6
X6	1	1	40	80,00	83,33	44,27	247,6
		12		480,00	1000,0	531,22	2011,2

На рисунку 4 видно частку, яку займає кожна з складових трудомісткості проєкту у варіанті запропонованої моделі (верхня частина рисунку), нижня частина рисунку показує структуру відповідно до базового варіанту визначення трудомісткості, на основі експертної оцінки.

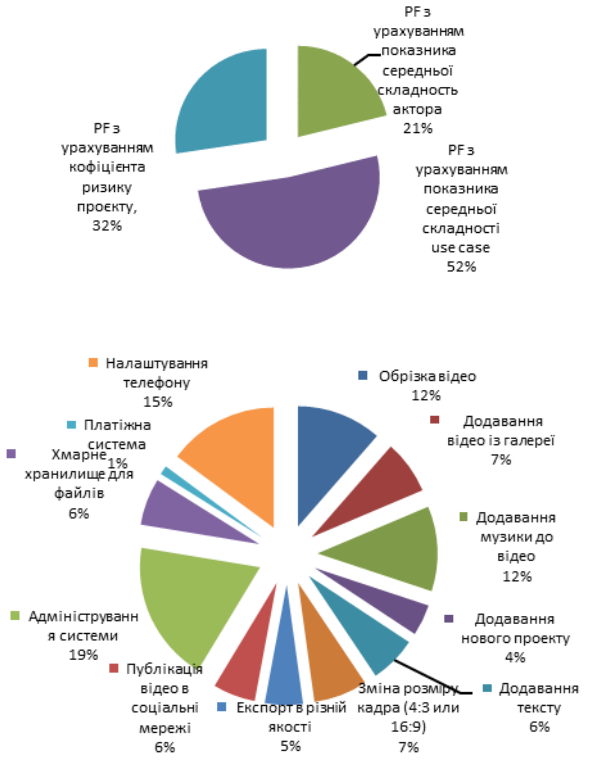

Рисунок 4 – Структура трудомісткості до та після зміни моделі визначення трудомісткості проєкту

Отримані результати показують, що традиційний підхід до оцінки трудомісткості, заснований на прямому підрахунку годин і коригуючих коефіцієнтах, дає значення 2394 години, T. На нижній діаграмі рисунку 4 видно структуру цих витрат і наочно демонструє велику кількість оцінювальних робіт, які необхідно провести команді експертів для визначення кількості трудових годин для проєкту. В той же час, більш сучасний і складний підхід, що використовує математичну модель на основі use case, оцінює трудомісткість у 2011 годин, T^* та має більш просту структуру оцінювальних робіт, не дивлячись на кількість факторів, що в ній закладені для розрахунку. Це дозволяє отримати на 16% меншу оцінку трудомісткості. Така різниця може бути особливо значною для великих проектів та буде корисною для побудови бюджету проєкту за різних сценаріїв при встановлених обмеженнях. Однак більш вагомим буде внесок запропонованої моделі через її точність та гнучкість в розрахунках, яка дозволяє врахувати різні фактори впливу на проект і прогнозувати трудомісткість в залежності від зміни вхідних параметрів.

В цілому можна зробити висновок, що обидва підходи можуть використовувати у практиці оцінки трудомісткості, для більш простих проектів – традиційний, для складних або з високим рівнем ризику – другий, на основі запропонованої моделі оцінки трудомісткості проєкту.

6 ОБГОВОРЕННЯ

Більшість наукових досліджень у сфері формування та оцінки вимог проєктів за методологією Agile проєкту присвячені збору та аналізу вимог на стадіях user requirements та system requirements, однак мало досліджень приділено фазі presale, тобто періоду, перед офіційним запуском процесу виконання проєкту. Основі методи, які наводяться та характеризуються – це user story та use case. однак в літературних джерелах не простежується тенденція до формалізації цих підходів у математичний формат для їх поєднання в єдину модель оцінки трудомісткості проєкту на етапі попередньої оцінки проєкту. Більшість досліджень зосереджені на стадії system requirements проєкту, тобто на стадії детального планування вимог до продукту для потреб команди проєкту, де для формування та оцінки вимог використовується метод User Story, який дозволяє зробити короткий опис фічі у форматі відповіді на питання: «Хто, що і навіщо робить?». Це пояснення фічі з точки зору кінцевого користувача, наприклад, за таким форматом опису (є й інші формати): AS <a type of user>, I WANT <some goal> SO THAT <some reason>.

Вихідними змінними тут виступає набір сформованих вимог (some goal) від користувача (a type of user) на етапі user requirements, що задовольняє мету користувача (some reason). Даний підхід до опису вимог користувача також враховує критерії

DOI 10.15588/1607-3274-2025-1-17

прийняття для окремої User story – це умови, які повинні бути виконані, щоб вважати історію користувача завершеною. Вони допомагають визначити, чи відповідає історія користувача потребам користувача і чи відповідає вона очікуванням команди розробки. Отже, фіча може складатися з однієї або декількох User Story, залежно від її масштабів. Це полегшує роботу команди та відповідний контроль над виконанням завдань проєкту. Критерії прийняття можна уточнювати як для кожної User story, так і цілої фічі. Вибір залежить від масштабу, складності та деталізації завдання. Така формалізація та структура User story є достатньо простою і зрозумілою для всіх стейкхолдерів на стадіях user requirements та system requirements проєкту.

Однак, проведений аналіз літературних джерел не виявив досліджень, які б застосовували такий формат опису вимог на рівні presale. Це пояснюється високою трудомісткістю та ресурсоємністю такої оцінки для компанії на даному етапі роботи із замовником. Тому очевидним стає застосування інших підходів для вирішення поставленої задачі. В нашому дослідженні вибір було зроблено на методі use case, який формалізовано в математичний шаблон для оцінки трудомісткості проєкту на етапі пресейлу.

Запропонований формалізм відкриває широкі перспективи для розробки більш точних і об'єктивних методів оцінки та математичних моделей, які дозволяють швидко, точно та ефективно визначати сценарії трудомісткості витрат проєкту, особливо з урахуванням ризиків розробки та зміни умов користувача.

Отримані результати дослідження демонструють потенціал запропонованої математичної моделі для більш точної оцінки трудомісткості розробки програмного забезпечення, зокрема мобільних додатків. Порівняння з традиційним методом, заснованим на експертних оцінках, виявило ряд переваг запропонованого підходу, зокрема:

– математична модель усуває суб'єктивність, властиву експертним оцінкам, за рахунок використання кількісних показників та алгоритмів розрахунку;

– такий підхід до визначення вимог та трудомісткості проєкту на етапі попередньої оцінки дозволяє враховувати широкий спектр факторів, що впливають на трудомісткість проекту, таких як зміна вимог, технологічні ризики та організаційна складність розробки функціоналу продукту;

– результати, отримані за допомогою запропонованої моделі на основі методу use case, демонструють більш високу точність порівняно з традиційним підходом;

– запропонована модель може бути легко застосована до інших проектів, забезпечуючи стабільність результатів.

Незважаючи на переваги, запропонована модель має певні обмеження. Наприклад, точність прогнозування залежить від якості вхідних даних та правильності калібрування моделі. Для більш точного прогнозування необхідно проводити додаткові дослідження та збирати більші обсяги даних про реальні проекти.

Перспективними напрямками подальших досліджень є:

– верифікація моделі на більшій вибірці проектів за типами та рівнями складності;

– перевірка стійкості моделі до різних типів проектів та масштабів, залежно від ризиків.

За проаналізованими факторами ризику, що наводяться в даній статті, проведено експертну оцінку їх впливу на трудомісткість проєкту. На рисунку 56 зображені відповідна динаміка потенційних втрат годин трудомісткості.

Згідно з наведеним графіком, втрати часу через різні фактори ризику суттєво впливають на загальну трудомісткість проєкту. Це означає, що непередбачені ситуації та проблеми, які виникають під час реалізації проекту, можуть призводити до значного збільшення часу, необхідного для його завершення. Оскільки в нашому дослідженні є сталий показник G на рівні 11067, в подальшому пропонується підходити до його розрахунку диференційовано в залежності від ймовірності настання в кожному окремому випадку і типу проєкту.

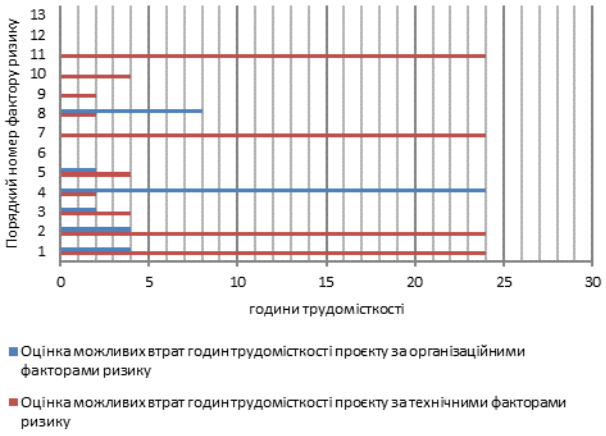

Рисунок 5 – Показники можливих втрат годин трудомісткості за факторами ризиків проєкту

Для більш глибокого розуміння впливу різних факторів ризику на трудомісткість проекту необхідно проводити подальші дослідження та створювати кількісні моделі, які дозволять прогнозувати їх рівень та ймовірність настання. Це дозволить більш точно підходити до планування проектів, гнучко реагувати на запити користувачів та витрачати менше ресурсів на їх оцінку.

ВИСНОВКИ

Вирішується актуальна проблема формування та оцінки вимог користувача в управлінні IT-проєктами на основі методу Use Case в методології Agile.

DOI 10.15588/1607-3274-2025-1-17

Наукова новизна отриманих результатів полягає у наступному:

1) вперше запропоновано комплексний підхід до проблеми формування та оцінки вимог до Agile-проєкту на етапі планування трудомісткості проєкту, що базується на врахуванні факторів зовнішнього та внутрішнього впливу та поєднанні графічних, аналітичних та математичних інструментів, зокрема, діаграми use case та коригуючих коефіцієнтів, що враховують складність системи та вплив організаційних та технічних ризиків.

2) дістав подальший розвиток метод Use Case шляхом деталізації за визначеними критеріями акторів та варіантів використання (use case) на прості, середні та складні, яка виражена через коефіцієнти складності актора та use case з метою їх застосування під час розрахунку одного з показників ефективності проєкту – трудомісткості.

3) вперше запропоновано математичну модель формування вимог Agile проєкту на основі методу use case, результат якої виражено показником загальної трудомісткості, що дозволяє робити точні розрахунки тривалості проєкту та будувати різні сценарії бюджету за рахунок використання кількісних показників та алгоритмів розрахунку, які враховують такі фактори як: зміна вимог замовника, технологічні ризики та організаційну складність розробки функціоналу продукту.

Практична значимість дослідження полягає в тому, що авторський підхід до вирішення проблеми формування вимог в управління IT-проєктом, що базується на використанні математичної моделі, дозволяє отримати швидкі і більш точні результати оцінки трудомісткості порівняно з традиційним підходом, а також легко адаптуватись до інших типів проєктів, забезпечуючи стабільність результатів. Проведений розрахунок підтвердив доцільність її застосування на прикладі мобільного додатку на 16% економії трудовитрат, а отже здатність досягати більшої ефективності.

Перспективними напрямками подальших досліджень є перевірка стійкості моделі до різних типів проєктів та масштабів, залежно від впливу ризиків.

ПОДЯКА

Робота підтримана державним університетом «Житомирська політехніка». Госпдоговірна тема: «Рекомендації щодо удосконалення веб-сторінок сайту deps.ua» (реєстраційний номер: 0123U102487).

ЛІТЕРАТУРА

1. The Impact of Incomplete or Changing Requirements on IT Project Success. [Electronic resource]. – Access mode: https://www.boardroommetrics.com/blog/the-impact-of-incomplete-or-changing-requirements-on-it-project-success-20131222.htm
2. The state-of-practice in requirements specification: an extended interview study at 12 companies / [X. Franch, C. Palomares, C. Quer et al.] // Requirements Engineering. – 2023. – No. 28 (3). – P. 377-409. DOI:10.1007/s00766-023-00399-7
3. Solutions for software requirement risks using artificial intelligence techniques / [V.K.R. Reddy, U. Rahamathunnisa, P. Subhashini et al.] // Handbook of Research on Data Science and Cybersecurity Innovations in Industry 4.0 Technologies. – 2023. – P. 45–64. DOI: 10.4018/978-1-6684-8145-5
4. IEEE Standard Glossary for Software Engineering Terminology, ANSI/IEEE Std 610.12-1990. [Electronic resource]. – Access mode: https://ieeexplore.ieee.org/document/159342
5. Козак О. Л. Опорний конспект лекцій з курсу «Аналіз вимог до програмного забезпечення» для студентів напрямку підготовки «Програмна інженерія» / О. Л. Козак. – Тернопіль, 2011. – 56 с. [Електронний ресурс]. – Режим доступу: http://dspace.wunu.edu.ua/retrieve/14135/FCIT_kKN_sPZS_dAVPZ_%20LEC.pdf
6. Грицюк Ю. І. Особливості формулювання вимог до програмного забезпечення. [Електронний ресурс] / Ю. І. Грицюк, О. А. Нємова. – Режим доступу: file:///C:/Users/%D0%9E%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D1%80%D0%B0/Downloads/Osoblivosti_formuluvanna_vimog_do_programnogo_zabe.pdf
7. ISO/IEC 25062. Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Common Industry Format (CIF) for Usability Test Reports (2006). [Electronic resource]. – Access mode: [Electronic resource]. – Access mode: https://www.iso.org/standard/43046.html
8. ISO 9241-210:2019. Ergonomics of human-system interaction – Human-centred design for interactive systems (2019). [Electronic resource]. – Access mode: https://www.iso.org/standard/77520.html
9. ISO/IEC/IEEE 29148. Systems and software engineering – Life cycle processes – Requirements engineering (2011). [Electronic resource]. – Access mode: https://www.iso.org/standard/72089.html
10. Kotonya G. Requirements Engineering: Processes and Techniques (1st ed.). / G. Kotonya, I. Sommerville // Wiley Publishing. – 1998. – P. 294.
11. Geis T. Specifying Usability Requirements and Test Criteria for Interactive Systems: Consequences for New Releases of Software-related Standards Within the ISO 9241 Series / T. Geis, W. Dzida, W. Redtenbacher // Federal Institute for Occupational Safety and Health. – 2004. – P.63.
12. Maguire M. User Requirements Analysis: A Review of Supporting Methods / M. Maguire, N. Bevan // In Proceedings of the IFIP 17th World Computer Congress – TC13 Stream on Usability: Gaining a Competitive Edge. – 2002. – P. 133–148. DOI:10.1007/978-0-387-35610-5_9
13. Hernande R. Requirements management in DevOps environments: a multivocal mapping study. / R. Hernandez, B. Moros, J. Nicolas // Requirements Engineering. – 2023.– 28 (3). – P. 317–346. Cited 1 time. DOI:10.1007/s00766-023-00396-w

DOI 10.15588/1607-3274-2025-1-17

14. Sathe C. A. An Empirical Study of Project Management Constraints in Agile Software Development: Multigroup Analysis between Scrum and Kanban / C. A. Sathe, C. Panse // Brazilian Journal of Operations and Production Management. – 2023. – Vol. 20, No. – P. 1–17. 3 special edition, https://doi.org/10.14488/BJOPM.1796.2023

15. Silvius G. Exploring the Project Owner's Behaviour of Addressing Sustainability in Project Assignment and Governance / G. Silvius, R. Ursem, J. Magano // Sustainability. – 2023. – No. 15(19). – P. 14294; https://doi.org/10.3390/su151914294

Стаття надійшла до редакції 05.12.2023
Після доробки 17.12.2024.

UDC 004.9

USE CASE METHOD IN IT PROJECT MANAGEMENT BASED ON AGILE METHODOLOGY

Svintsycka O. M. – PhD, Associate professor, Associate professor of the Educational Department of Computer Sciences, Zhytomyr Polytechnic State University, Zhytomyr, Ukraine.

Puleko I. V. – PhD, Associate professor, Associate professor of the Educational Department of Computer Engineering and Cyber Security, Zhytomyr Polytechnic State University, Zhytomyr, Ukraine.

Graf M. S. – PhD, Head of the Educational Department of Computer Sciences, Zhytomyr Polytechnic State University, Zhytomyr, Ukraine.

Petrosian R. V. – Senior Lecturer, Computer Sciences department, Zhytomyr Polytechnic State University, Zhytomyr, Ukraine.

ABSTRACT

Context. The article considers the role and process of forming user requirements based on the Use Case method in assessing the complexity of an Agile project at the stage of preliminary assessment by the company's management. Since the mid-70s, it has been known that errors in requirements are the most numerous, expensive, and time-consuming to correct in projects. In this regard, the importance of requirements management in IT projects using modern technologies and methods for their formation and evaluation is increasing.

Objective. Formation and evaluation of user requirements in IT project management based on the Use Case method and their impact on one of the project performance indicators at the planning stage, particularly labor intensity.

Method. The article proposes a new author's approach to the formation and evaluation of user requirements in Agile projects, taking into account the impact of risks and system complexity assessment based on the Use Case method, and as a result of the study and proposals to achieve this goal, a mathematical model for estimating project complexity is proposed. The mathematical template of the model allows us to consider additional variables that may affect the project, such as the number of user levels, available functionality, and technical and organizational risks. It is flexible and can be adapted to the different needs of a particular project, which aligns with the principles of the Agile methodology. The number of components in the formula can be changed to take into account the importance of different variables or expanded to take into account additional variables that may affect the project.

Results. A mathematical model for estimating project complexity based on the use case method has been developed and tested using the example of a mobile application, which contains a set of initial data for product development and constraints on changing user requirements and organizational and technical risks. The proposed mathematical model allows you to quickly, accurately, and efficiently determine scenarios of project labor intensity of various types and levels of complexity and can serve as an effective tool for making management decisions. A mathematical model for estimating project complexity based on the use case method has been developed and tested using the example of a mobile application, which contains a set of initial data for product development and constraints on changing user requirements and organizational and technical risks. The proposed mathematical model allows you to quickly, accurately, and efficiently determine scenarios of project labor intensity of various types and levels of complexity and can serve as an effective tool for making management decisions.

Conclusions. The general findings obtained after analyzing the methods of forming and evaluating user requirements in Agile management are as follows. At the work planning stage, based on an expert assessment of each functional requirement, the primary project evaluation model has been replaced by a more modern and complex one based on the use case method and considering changes in user requirements and other product development risks. The new model uses graphical, analytical, and mathematical tools, including a use case diagram, adjustment factors considering the complexity of the actor and use case, and factors considering organizational and technical risks. As a result, we get a mathematical format for calculating the project's complexity. This approach allows us to adapt to different types of projects quickly. With the correct initial data definition, the model will enable us to obtain reasonably accurate estimates early in project planning. The practical results of the study demonstrate the potential of the proposed mathematical model, which can be logically continued by verifying the model on a larger sample and assessing its resilience to different types of projects and risks.

KEYWORDS: user requirements, IT product, Agile project, User Story, Use Case, mobile application.

DOI 10.15588/1607-3274-2025-1-17

REFERENCES

1. The Impact of Incomplete or Changing Requirements on IT Project Success. [Electronic resource]. Access mode: https://www.boardroommetrics.com/blog/the-impact-of-incomplete-or-changing-requirements-on-it-project-success-20131222.htm
2. Franch X., Palomares C., Quer C., Chatzipetrou P., Gorschek T. The state-of-practice in requirements specification: an extended interview study at 12 companies, *Requirements Engineering*, 2023, No. 28 (3), pp. 377–409. DOI:10.1007/s00766-023-00399-7
3. Reddy V. K. R., Rahamathunnisa U., Subhashini P., Aancy H. M., Meenakshi S., Boopathi S. Solutions for software requirement risks using artificial intelligence techniques, *Handbook of Research on Data Science and Cybersecurity Innovations in Industry 4.0 Technologies,* 2023, pp. 45–64. DOI: 10.4018/978-1-6684-8145-5
4. IEEE Standard Glossary for Software Engineering Terminology, ANSI/IEEE Std 610.12-1990. [Electronic resource]. Access mode: https://ieeexplore.ieee.org/document/159342
5. Kozak O. L. Opornyi konspekt lektsii z kursu «Analiz vymoh do prohramnoho zabezpechennia» dlia studentiv napriamku pidhotovky «Prohramna inzheneriia». Ternopil, 2011, 56 p. [Elektronnyi resurs]. Rezhym dostupu: http://dspace.wunu.edu.ua/retrieve/14135/FCIT_kKN_sPZS_dAVPZ_%20LEC.pdf
6. Hrytsiuk Yu.I., Niemova O.A. Osoblyvosti formuliuvannia vymoh do prohramnoho zabezpechennia. [Elektronnyi resurs]. Rezhym dostupu: file:///C:/Users/%D0%9E%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D1%80%D0%B0/Downloads/Osoblivosti_formuluvanna_vimog_do_programnogo_zabe.pdf
7. ISO/IEC 25062. Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Common Industry Format (CIF) for Usability Test Reports (2006). [Electronic resource]. Access mode: [Electronic resource]. Access mode: https://www.iso.org/standard/43046.html
8. ISO 9241-210:2019. Ergonomics of human-system interaction – Human-centred design for interactive systems (2019). [Electronic resource]. Access mode: https://www.iso.org/standard/77520.html
9. ISO/IEC/IEEE 29148. Systems and software engineering – Life cycle processes – Requirements engineering (2011). [Electronic resource]. Access mode: https://www.iso.org/standard/72089.html
10. Kotonya G., Sommerville I. Requirements Engineering: Processes and Techniques (1st ed.). Wiley Publishing, 1998, P. 294.
11. Geis T., Dzida W., Redtenbacher W. Specifying Usability Requirements and Test Criteria for Interactive Systems: Consequences for New Releases of Software-related Standards Within the ISO 9241 Series, *Federal Institute for Occupational Safety and Health,* 2004, P.63.
12. Maguire M., Bevan N. User Requirements Analysis: A Review of Supporting Methods, *In Proceedings of the IFIP 17th World Computer Congress – TC13 Stream on Usability: Gaining a Competitive Edge,* 2002, pp. 133–148. DOI:10.1007/978-0-387-35610-5_9
13. Hernande R., Moros B., Nicolas J. Requirements management in DevOps environments: a multivocal mapping study, *Requirements Engineering*, 2023, No. 28 (3), pp. 317–346. Cited 1 time. DOI:10.1007/s00766-023-00396-w
14. Sathe C. A., Panse C. An Empirical Study of Project Management Constraints in Agile Software Development: Multigroup Analysis between Scrum and Kanban, *Brazilian Journal of Operations and Production Management*, 2023, Vol. 20, No, pp. 1–17. 3 special edition, https://doi.org/10.14488/BJOPM.1796.2023.
15. Silvius G., Ursem R., Magano J. Exploring the Project Owner's Behaviour of Addressing Sustainability in Project Assignment and Governance, *Sustainability*, 2023, No. 15(19), P. 14294; https://doi.org/10.3390/su151914294

DOI 10.15588/1607-3274-2025-1-17