Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп’ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп’ютерні науки»
зі спеціальності
122 – Комп’ютерні науки

тема роботи:

«Автоматизація етапів життєвого циклу процесу тестування програмного забезпечення»

Виконав студент гр. ЗКН-21 _____ Васильєв В. А.

Керівник _____ Бугра А. В.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2025

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *29* » *січня* 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи ЗКН-21 *Васильєву Віталію Андрійовичу*

1. Тема кваліфікаційної роботи: *«Автоматизація етапів життєвого циклу процесу тестування програмного забезпечення»*

затверджено наказом по університету № *255с* від *13.05.2025 р.*

2. Термін здачі кваліфікаційної роботи: *06.06.2025 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 78с., додатки, презентація у Microsoft PowerPoint (13 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Бугра А. В.

_____

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.03.25*
2	*Розділ 1*	*01.04.25*
3	*Розділ 2*	*01.05.25*
4	*Висновки*	*25.05.25*
5	*Оформлення кваліфікаційної роботи*	*28.05.25*
6	*Підготовка презентації та графічного матеріалу*	*20.05.25*
7	*Підготовка доповіді до захисту*	*05.06.25*

6. Дата видачі завдання: *28.01.2025р.*

Керівник _____ / Бугра А. В./

7. Запевнення: *Я, Васильєв Віталій Андрійович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Васильєв В. А./

АНОТАЦІЯ

Васильєв В. В. Автоматизація етапів життєвого циклу процесу тестування програмного забезпечення : кваліфікаційна робота бакалавра : 122 – Комп’ютерні науки. Кривий Ріг. Криворізький національний університет, 2025. 75 с.

Об'єктом дослідження є бізнес-процеси життєвого циклу програмного забезпечення на прикладі відділу контролю якості (відділу тестування) IT-компанії.

Метою роботи є аналіз та оптимізація бізнес-процесів тестування програмного забезпечення на прикладі відділу контролю якості IT-компанії.

У першому розділі було обґрунтовано теоретичні основи тестування ПЗ, визначено місце тестування у життєвому циклі розробки програмного забезпечення та підкреслено важливість інтеграції бізнес-процесів тестування у загальну стратегію IT-компанії для досягнення високої якості кінцевого продукту.

У другому розділі проведено аналіз та моделювання бізнес-процесів тестування програмного забезпечення. Для моделювання бізнес-процесів використано систему Bizagi, яка підтримує нотацію BPMN. Цей інструмент дозволив візуалізувати бізнес-процеси, а також підготувати їх для імітаційного моделювання та аналізу результатів. Розглянуто налаштування бізнес-процесів для імітаційного моделювання, проведено аналіз отриманих результатів, а також змодельовано оптимізовані бізнес-процеси, які враховують виявлені недоліки. Отримані результати імітаційного моделювання оптимізованих процесів підтвердили доцільність впровадження заходів щодо їх вдосконалення.

Ключові слова:

БІЗНЕС ПРОЦЕС, ЖИТТЄВИЙ ЦИКЛ ТЕСТУВАННЯ, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ОПТИМІЗАЦІЯ БІЗНЕС ПРОЦЕССІВ, ТЕСТУВАННЯ ПЗ

ЗМІСТ

ВСТУП

Більшість компаній, що мають у своїй структурі IT-відділ, задаються питанням щодо прискорення процесу розробки без втрати його якості. Жоден цикл розробки програмного забезпечення не обходиться без етапу тестування. Тестування програмного забезпечення – процес аналізу програмного засобу та супровідної документації з метою виявлення дефектів і підвищення якості продукту.

З розвитком індустрії розробки програмного забезпечення та ускладненням технологічного процесу, появою комплексних методів розв'язання задач, роль тестування зросла. Замість однієї з фінальних стадій створення проєкту тестування почало застосовуватись протягом усього циклу розробки. Надалі були створені гнучкі методики тестування, різноманітні оптимізації та було поглиблено інтеграцію з процесом розробки.

На даний момент в індустрії спостерігається величезний інтерес до тестування як такого, так і до методів покращення якості програм загалом, адже кінцевою метою будь-якого процесу тестування є забезпечення якості. Тестування програмного забезпечення дозволяє визначити, чи виконує програма те, що від неї очікують. А основним завданням тестування програмного забезпечення є зниження вартості розробки шляхом раннього виявлення дефектів.

Раннє виявлення дефектів і своєчасний зворотний зв'язок дозволяють створювати високоякісне й надійне програмне забезпечення. Крім того, організація ефективного й безперервного тестування пришвидшує постачання цього програмного забезпечення та зменшує витрати компанії завдяки тому, що у розробників з'являється можливість вносити зміни в код із мінімальними ризиками, здатними порушити працездатність програми.

Саме тому основною метою випускної кваліфікаційної роботи був аналіз і оптимізація бізнес-процесів тестування програмного забезпечення на прикладі відділу контролю якості IT-компанії.

Для досягнення поставленої мети були сформульовані наступні завдання:

– Огляд бізнес-процесів тестування програмного забезпечення;

– Аналіз бізнес-процесів тестування програмного забезпечення;

– Моніторинг роботи учасників відділу тестування;

– Моделювання бізнес-процесів тестування програмного забезпечення;

– Проведення імітаційного моделювання бізнес-процесів тестування програмного забезпечення;

– Аналіз результатів імітаційного моделювання бізнес-процесів тестування програмного забезпечення;

– Розробка плану заходів щодо оптимізації бізнес-процесів тестування програмного забезпечення;

– Впровадження заходів щодо оптимізації бізнес-процесів тестування програмного забезпечення;

– Розрахунок ефективності від впровадження заходів щодо оптимізації бізнес-процесів тестування програмного забезпечення.

Отже, об’єктом дослідження є бізнес-процеси життєвого циклу програмного забезпечення на прикладі відділу контролю якості (відділу тестування).

Практична значущість дослідження полягає у розробці та впровадженні заходів щодо оптимізації бізнес-процесів тестування програмного забезпечення, які дозволять скоротити час і вартість процесу тестування, не втрачаючи якості програмного продукту, що випускається.

РОЗДІЛ 1
ОПИС ТА АНАЛІЗ ТЕОРЕТИЧНИХ АСПЕКТІВ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Опис основних проблем питання

Згідно з даними аналітичного звіту щодо ринку тестування програмного забезпечення за 2022–2024 [1] роки, 80% респондентів вважають, що головною метою відділу тестування є підвищення якості IT-продуктів, а 69% орієнтовані на підвищення задоволеності користувачів. При цьому мета щодо скорочення часу виведення продуктів на ринок займає третє місце і цікавить 62% респондентів, що на 26% більше, ніж у 2020 році.

Велика кількість компаній прагне скоротити терміни випуску продукту, при цьому не втративши якість і збільшивши задоволеність користувачів. За результатами опитування можна сказати, що така потреба в компаніях щороку лише зростає. Саме оптимізація процесів тестування може дозволити вирішити проблему швидкого випуску продукту без втрати його якості.

У статті [2], присвяченій скороченню часу тестування програмного забезпечення, розглядається метод підвищення ефективності тестування ПЗ, що дозволяє скоротити час тестування без втрати якості. Для цього методу проводились опитування у 17 команд, які займаються тестуванням, з метою з’ясувати, як у цих командах побудований процес тестування, а саме який вид тестування використовується і скільки людей бере участь у тестуванні.

У результаті дослідження автором статті були розроблені рекомендації, які дозволяють підвищити ефективність кожного розглянутого виду тестування.

У даній роботі буде запропонований і описаний алгоритм з оптимізації процесу тестування з урахуванням досвіду, методів оцінки ефективності та розрахунку економіки витрат у процесі тестування, описаних у статті.

1.2 Життєвий цикл тестування програмного забезпечення

Тестування є невід'ємною частиною сучасної розробки програмного забезпечення. Хоча існує дуже багато підходів до розробки програмного забезпечення, тестування необхідне скрізь. Тестування має свій власний життєвий цикл, який називають життєвим циклом тестування програмного забезпечення або STLC. Життєвий цикл тестування програмного забезпечення допомагає здійснювати процес тестування належним і ретельним чином.

Тестування програмного забезпечення, як і розробка програмного забезпечення, включає в себе кілька кроків, що йдуть у певній послідовності. Це і називається життєвим циклом тестування програмного забезпечення. Він визначає початок, кінець і проміжні моменти повного процесу тестування ПЗ.

Кожен етап STLC має певні результати, цілі та завдання. Хоча різні тестувальники можуть по-різному підходити до етапів, таких як об'єднання одного чи кількох в один, повторення тощо — основна ідея залишається незмінною. Нижче наведено основні фази життєвого циклу тестування програмного забезпечення, які використовуються незалежно від обраного підходу [3]:

- аналіз вимог;
- планування тестування;
- розробка тестів;
- виконання тестування;
- підготовка документації.

Кожен із цих етапів так чи інакше є обов'язковим для життєвого циклу тестування, незалежно від того, наскільки він формалізований і задокументований. Далі кожен етап буде розглянуто більш детально.

1.2.1 Аналіз вимог

Аналіз вимог являє собою діяльність з дослідження нових змін, аналізу документації до них, виконання статичного тестування вимог за необхідності.

Найчастіше цей етап починається після завершення погодження технічних вимог і специфікацій аналітиком команди. Вимоги можуть бути описані у явному формалізованому вигляді, а можуть бути представлені у вигляді user story. На цьому етапі тестувальник має змогу вплинути на реалізацію майбутніх змін з урахуванням його бачення системи та процесу загалом з точки зору тестування.

На цьому етапі тестувальник отримує розуміння змін, які плануються до розробки та впровадження. Результатом є фактично сформований чек-лист тестових вимог, які належить перевірити тестувальнику. Це може бути як формалізований документ, так і просто знання в голові конкретного фахівця з тестування.

1.2.2 Планування тестування

На етапі планування тестувальник визначається з цілями тестування, необхідними типами тестування, вимогами до тестового оточення (тестового середовища), визначає критерії початку та завершення кожного виду тестування, оцінює попередні строки тестування на основі отриманих раніше тестових вимог, ідентифікує потенційні ризики та спільно з іншими учасниками команди опрацьовує варіанти їх мінімізації, а також будь-які інші специфічні вимоги до тестування, які можуть виникнути в результаті виконання тестів.

Результатом цього етапу може бути формалізований тест-план, який описує всі вищезгадані моменти. Цей документ найчастіше додатково погоджується тестувальником з усіма зацікавленими учасниками проєкту. Буває, що такий документ не формалізується, а всі аспекти обговорюються всередині команди на зустрічах з планування та фіксуються в протоколах цих зустрічей. Це значно скорочує процес планування тестування, оскільки є можливість одразу обговорити відкриті питання з усіма учасниками команди та знайти відповідне рішення. Результатом у такому випадку буде лише оцінка активностей з тестування.

Після завершення етапу планування тестувальник переходить до одного з ключових етапів процесу тестування — це розробка тестів.

1.2.3 Розробка тестів

Після опрацьованого тест-плану тестувальник приступає до розробки тест-кейсів і їх наповнення. Цей етап може займати досить тривалий час, особливо якщо йдеться про великий проєкт. Тестувальник насамперед складає чек-лист перевірок, формує структуру тест-кейсів і після цього переходить до їх наповнення кроками, різними умовами, а також визначенням тестових даних для їх успішного виконання. Крім того, на цьому етапі можуть визначатися тест-кейси, які підлягатимуть подальшій автоматизації. Існує підхід, коли тестувальник створює для себе чек-лист перевірок з тестовими даними та умовами, але не переходить до написання повноцінних тест-кейсів. Результатом цього етапу є сформоване детальне розуміння того, що необхідно буде тестувати. Це можуть бути повноцінні тест-кейси, чек-листи або бачення того, що потрібно буде перевіряти. Окрім цього, необхідно виконати підготовку до тестування.

Підготовка до тестування включає такі активності:

– Деплой версії застосунку з необхідними для тестування змінами;

– Створення тестового оточення з необхідною конфігурацією;

– Підготовка тестових даних для тестування;

– Доопрацювання скриптів автоматизації тестування.

Як результат тестувальник отримує працездатне тестове середовище з тестовими даними та встановленим застосунком для тестування.

1.2.4 Виконання тестування

Цей етап передбачає послідовне виконання кількох типів тестування до моменту отримання позитивного висновку та успішного проходження тестів. Спочатку виконується інтеграційне або навіть одразу системне тестування — залежно від специфіки тестування в конкретній компанії. Після завершення

системного тестування проводиться регресійне тестування. Найчастіше завдяки автоматизації тестування, тестувальник виконує лише інтеграційне й системне тестування, після чого повністю або частково запускаються автоматизовані регресійні тести. Приймальне тестування виконується або на стороні замовника, або на спеціально відведеній для цього зустрічі, яку часто називають «Демо». Під час цієї зустрічі тестувальник демонструє реалізацію функціоналу системи, і замовник приймає ці зміни або відправляє на повторне доопрацювання. Результатами цього етапу можна вважати успішно пройдені тест-кейси або перевірки за чек-листом.

1.2.5 Підготовка документації

Останнім завершальним етапом тестування перед впровадженням нових змін на продуктивні середовища є звітність про результати тестування. Цей етап передбачає створення формальних звітів за результатами тестування змін, які можуть включати в себе загальні результати тестування, відкриті проблеми й дефекти, а також рекомендації для супроводу щодо встановлення змін на продуктивне середовище.

На цьому у більшості випадків процес тестування завершується, і зміни передаються для встановлення на продуктивне середовище кінцевим користувачам або замовникам.

1.3 Види тестування програмного забезпечення

Види тестування прийнято класифікувати за різними ознаками [4]:

– за доступністю коду;

– за об’єктом тестування;

– за позитивністю сценаріїв;

– за цілями тестування;

– за ступенем автоматизації;

– за виконанням коду;

– за рівнем тестування;

– пов’язані зі змінами;

– за виконавцями тестування;

– тощо.

У різних джерелах класифікація може відрізнятися за ознаками, але загальна суть видів тестування залишається такою ж. У цій роботі буде розглянуто види за рівнем тестування, за виконанням коду та пов’язані зі змінами.

Тестування за виконанням коду поділяють на: статичне і динамічне [5].

Статичне тестування – тип тестування, який передбачає, що програмний код під час тестування не буде виконуватись. Статичне тестування починається на ранніх етапах життєвого циклу ПЗ і є, відповідно, частиною процесу аналізу вимог. Більшість статичних технік можуть бути використані для «тестування» будь-яких форм документації, включаючи вичитку коду, інспекцію проєктної документації, функціональної специфікації та вимог.

Динамічне тестування – тип тестування, який передбачає запуск програмного коду. Таким чином, аналізується поведінка програми під час її роботи. Для виконання динамічного тестування необхідно, щоб тестований програмний код був написаний, скомпільований і запущений. Крім того, динамічне тестування може включати різні підвиди, наприклад, рівні тестування.

Тестування на різних рівнях проводиться протягом усього життєвого циклу розробки та супроводу програмного забезпечення. Рівень тестування визначає те, над чим виконуються тести: над окремим модулем, групою модулів або системою в цілому. Виділяють чотири основні рівні тестування: компонентне або модульне, інтеграційне, системне та приймальне [5].

Компонентне або модульне тестування перевіряє функціональність і шукає дефекти в частинах застосунку, які доступні і можуть бути протестовані окремо, наприклад, модулі програм, об’єкти, класи, функції тощо. Зазвичай компонентне тестування проводиться шляхом виклику коду, який необхідно

перевірити, із використанням середовищ розробки. Усі знайдені дефекти, як правило, виправляються в коді без формального їх опису.

Інтеграційне тестування призначене для перевірки зв'язку між компонентами, а також взаємодії з різними частинами системи або зв'язку між різними системами. Інтеграційне тестування, у свою чергу, поділяється на рівні:

– компонентний інтеграційний рівень перевіряє взаємодію між компонентами системи після проведення компонентного тестування;

– системний інтеграційний рівень перевіряє взаємодію між різними системами після проведення системного тестування.

Основним завданням системного тестування є перевірка як функціональних, так і нефункціональних вимог всієї системи загалом. Під час системного тестування виявляються неправильне використання ресурсів системи, несумісність з оточенням, непередбачені сценарії використання, відсутня або неправильна функціональність, незручність використання тощо. Для мінімізації ризиків, пов'язаних з особливостями поведінки системи у тій чи іншій середовищі, під час тестування рекомендується використовувати тестове оточення, максимально наближене до того, на яке буде встановлено продукт після випуску.

Виділяють два підходи до системного тестування:

– на основі вимог, коли для кожної вимоги пишуть тесткейси, що перевіряють виконання цієї вимоги;

– на основі випадків використання, коли на основі уявлень про способи використання продукту створюють тесткейси для перевірки кожного сценарію.

Приймальне тестування – це формальний процес тестування, який перевіряє відповідність системи вимогам і проводиться з метою:

– визначення задоволення системою приймальних критеріїв;

– ухвалення рішення замовником або іншою уповноваженою особою про прийняття додатку.

Приймальне тестування виконується на основі набору типових тестових сценаріїв, розроблених на основі вимог до цього додатку. Фаза приймального тестування триває доти, доки замовник не ухвалить рішення про випуск додатку або відправлення його на доопрацювання.

Здебільшого тестувальники беруть участь в інтеграційному, системному та приймальному тестуваннях. Кожен рівень тестування використовується при видах тестування, пов’язаних зі змінами.

Після внесення необхідних змін, таких як виправлення дефектів, програмне забезпечення повинно бути протестоване для підтвердження того факту, що проблема справді була вирішена. Нижче перелічені види тестування, які необхідно проводити після встановлення програмного забезпечення, для підтвердження працездатності додатку або правильності виконаного виправлення дефекту [5].

Димове тестування походить із інженерного середовища: «Під час введення в експлуатацію нового обладнання вважалося, що тестування пройшло успішно, якщо з установки не пішов дим.»

У сфері програмного забезпечення димове тестування розглядається як короткий цикл тестів, що виконується для підтвердження того, що після збірки коду (нового або виправленого) встановлюваний додаток запускається і виконує основні функції.

Висновок про працездатність основних функцій робиться на основі результатів поверхневого тестування найважливіших модулів додатку щодо можливості виконання необхідних завдань та наявності критичних і блокуючих дефектів. У разі відсутності таких дефектів димове тестування оголошується пройденим і додаток передається для проведення повного циклу тестування, інакше — димове тестування оголошується проваленим і додаток повертається на доопрацювання.

Регресійне тестування — це вид тестування, спрямований на перевірку змін, внесених у додаток або навколишнє середовище, наприклад, виправлення дефекту, злиття коду, міграція на іншу операційну систему або базу даних, для

підтвердження того факту, що існуюча раніше функціональність працює так само, як і раніше.

Як правило, для регресійного тестування використовуються тесткейси, написані на ранніх етапах розробки та тестування. Це дає гарантію того, що зміни у новій версії додатку не пошкодили вже існуючу функціональність. Сам термін регресійного тестування залежно від контексту використання може мати різний зміст:

– регресія багів — спроба довести, що виправлена помилка насправді не виправлена;

– регресія старих багів — спроба довести, що недавня зміна коду або даних зламала виправлення старих помилок, тобто старі баги знову стали відтворюватися;

– регресія побічного ефекту — спроба довести, що недавня зміна коду або даних зламала інші частини розроблюваного додатку.

1.4 Види тестової документації

Тестова документація — це тип документації, яка описує процес, цілі та результати тестування програмного забезпечення [6]. Вона також може містити інформацію про середовище, налаштування та конфігурацію, необхідні для проведення тестування. Тестова документація використовується для доведення деталей плану тестування або стратегії до відома зацікавлених сторін, розробників і тестувальників.

Деякі компанії стверджують, що тестова документація — це необов’язковий процес, який лише збільшує витрати на розробку і більше працює на імідж компанії-розробника програмного забезпечення, ніж на забезпечення реальної цінності. Але правда в тому, що тестова документація є невід’ємною частиною процесу тестування, яка повинна передувати, супроводжувати і завершувати будь-який бізнес-процес тестування.

Основними цілями тестової документації є:

– управління майбутніми роботами з тестування та демонстрація зацікавленим сторонам результатів тестування, щоб вони мали можливість приймати обґрунтовані рішення щодо робіт із розробки програмного забезпечення;

– запобігання пропускам і проблемам у процесах тестування або дублюванню при проведенні тестування, щоб переконатися, що виконані всі необхідні тести.

Тестова документація дозволяє зробити тестування прозорим, оскільки документація з тестування показує всі аспекти тестування — від постановки цілей до методів тестування та результатів. Це дає замовнику чітке уявлення про те, яка робота була виконана і які результати отримані. Це також корисно для інших тестувальників, яким передається завдання, і для розробників, яким доводиться вирішувати виявлені дефекти або проблеми.

Також документація покращує комунікацію в команді, тому що кількість помилок при спілкуванні значно зменшується, якщо в команди є єдине джерело інформації про функціональність продукту або навіть про процес роботи над проєктом. Бувають ситуації, коли всі розуміють ситуацію по-різному, і в такому випадку є на що послатися і що обговорити. Також немає необхідності робити зайві дзвінки, щоб отримати потрібну інформацію, бо вона постійно оновлюється, і кожен знає, де шукати її актуальну версію. І, звичайно, це дуже допомагає, коли в команду приходять нові люди або коли команда тестування змінюється загалом, що без якісної документації може бути неможливо.

За належного документування тестування ризик виникнення несподіваних проблем значно знижується. Добре складена тестова документація відображає реальний досвід розробки, на основі якого можна зробити багато цінних речей і використовувати найкращі практики в інших проєктах. Документацію можна використовувати для навчання нових членів команди, пошуку найкращих практик і обміну ними, проведення аргументованих дискусій і багато чого іншого.

Хороша тестова документація є відмінною демонстрацією того, як проводиться фундаментальне тестування, які використовуються підходи та як тестуються різні частини програмного продукту.

Існує багато тестових документів, і вони можуть відповідати найрізноманітнішим стандартам. Найпоширенішим типом тестової документації є план тестування. У плані тестування описується підхід, який буде застосований для тестування конкретної програмної системи. Він містить інформацію про те, що буде тестуватися, як це буде тестуватися і хто відповідатиме за проведення тестів.

Інші типи тестової документації включають тест-кейси, тестові сценарії та звіти про дефекти. Тестові приклади описують конкретні кроки, які необхідно виконати для тестування конкретної функції. Тестові сценарії — це більш детальні інструкції, в яких точно описується, як слід проводити тестування. Звіти про дефекти документують будь-які помилки, виявлені під час тестування.

Типи тестової документації залежать від конкретної компанії, продукту та замовника. Далі будуть докладніше розглянуті найбільш поширені типи тестової документації.

Внутрішні тестові документи необхідні для виконання робочих завдань і для використання членами команди розробки та тестування. Вони містять цілі, методи та результати тестування на технічному рівні.

Стратегія тестування — це документ, у якому викладаються основні аспекти тестування, зокрема, на яких рівнях проєкту воно буде виконуватися. Під час тестування розробники, проектувальники та менеджери можуть у будь-який час звертатися до цього документа, щоб переконатися, що тестування й надалі проводиться в межах початкової стратегії та визначеної області.

План тестування — це документ, у якому детально описані всі основні аспекти проєкту тестування, такі як обсяг, підхід, ресурси, розклад, учасники тестування та їх діяльність. Оскільки це найпростіший документ, він

використовується найчастіше, розповсюджується серед усієї команди та доводиться до відома клієнтів.

Чек-лист — це документ, який містить короткий опис функцій, які повинен перевірити тестувальник. Виглядає чек-лист як список функцій із зазначенням результату перевірки. Чек-листи можуть використовуватися замість тест-кейсів, оскільки їх легше підготувати. Але якщо потрібно більш конкретний опис процедури, без тест-кейсів не обійтися.

Тест-кейс — це документ, у якому міститься докладний опис кроків і дій, які тестувальник повинен виконати для тестування якоїсь однієї частини функціоналу, а також критерії проходження тестів. Компанії можуть використовувати різні формати тест-кейсів, але інформація в них завжди дуже детальна і конкретна [7].

Тестові дані - це документ, який описує дані, необхідні для тестування, щоб наблизити продуктивність додатку до реального використання. Наприклад, це можуть бути набори згенерованих користувачів, документи, медіафайли, метадані та все інше, з чим додаток має працювати у реальному світі.

Зовнішні тестові документи необхідні для наочного представлення кінцевих результатів, оскільки використання різних типів візуалізацій полегшує їх розуміння. Вони часто надаються замовникам і можуть бути надані користувачам у тій чи іншій формі.

Звіти про помилки - це документ, який описує конкретну помилку в додатку у всіх її можливих аспектах. Це дуже важливий і часто публікуваний звіт, який безпосередньо допомагає покращити продукт у конкретних аспектах.

Звіт по тестуванню - це документ, у якому узагальнюються результати циклу тестування. Він може містити вартість виявлення дефектів, ефективність набору тестів, результативність тестування, співвідношення зусиль на доопрацювання та перевірку тощо.

Звіт про приймання користувачем - це документ, що містить результати тестування перед відправкою замовнику на відповідність вимогам. Це гарантує, що погляди розробників і замовників залишаються ідентичними під час розробки і тестування, а технічна реалізація повністю їм відповідає.

Для того, щоб документація працювала на користь команді та проекту і приносила користь, необхідно дотримуватися таких правил.

Підтримка документації в актуальному стані.

Поширеною помилкою є припинення оновлення документації, навіть якщо спочатку ви виконували всю документацію дуже добре. І це здається незначним, але лише до того моменту, поки документація не знадобиться проекту чи команді.

Зберігати всю документацію в одному місці. Документація не повинна бути розкидана по різних місцях у спробі зберегти її максимально близько до відповідної кодової бази. Це може здатися доброю ідеєю, але на практиці це може збити з пантелику членів команди і створити плутанину.

Зберігати документацію в надійному місці. Не розміщувати у відкритому доступі жодної документації, яка може негативно вплинути на бізнес-цілі. У такому випадку може знадобитися зашифроване сховище і можливість надавати доступ певним особам.

Використовувати стандартні шаблони для документації. Найкраще застосовувати універсальні рішення для документації, такі як Word і Excel. Це дозволить будь-якій зацікавленій особі легко відкривати і читати їх без встановлення додаткового програмного забезпечення чи інших ускладнень.

Робити документацію докладною. Документація має бути добре структурованою, змістовною, докладною та зрозумілою. Всі учасники процесу, які користуються документацією, повинні розуміти її однозначно.

Набір тестової документації може змінюватися від проекту до проекту. Але існують незмінні тестові документи, які якісне ведення яких дає компанії, її співробітникам і клієнтам безліч переваг.

1.5 Аналіз та моделювання бізнес-процесів тестування програмного забезпечення

1.5.1 Діяльність IT компанії

Компанія займається розробкою програмного забезпечення на замовлення, а також має власну продуктовою лінійку. Основна діяльність компанії:

– розробка ПЗ "під ключ" (концепція, архітектура, аналітика, розробка, тестування);

– автоматизація та діджитал-трансформація підприємств і процесів (3D, CAD, GIS);

– розробляємо та імплементуємо власні продукти з автоматизації;

– IT-консалтинг.

Продуктова лінійка компанії – це геоінформаційні рішення GeoSolution, які призначені для збору, зберігання, обробки, представлення та моделювання просторових даних і пов’язаної з ними атрибутивної інформації. Ця лінійка містить такі продукти [8]:

Структура компанії наведена на рисунку 1.1. У своїй структурі компанія містить наступні відділи:

– відділ управління проєктами включає керівників IT-проєктів, які займаються ініціалізацією проєкту, управлінням бюджетом та організацією команди;

– відділ контролю якості включає фахівців з контролю якості або тестувальників, які відповідають за якість розроблюваного ПЗ;

– група обробки даних включає фахівців, які відповідають за налаштування геоданих у продуктовій лінійці;

– відділ персоналу включає HR-спеціалістів, які займаються підбором персоналу та організацією заходів компанії;

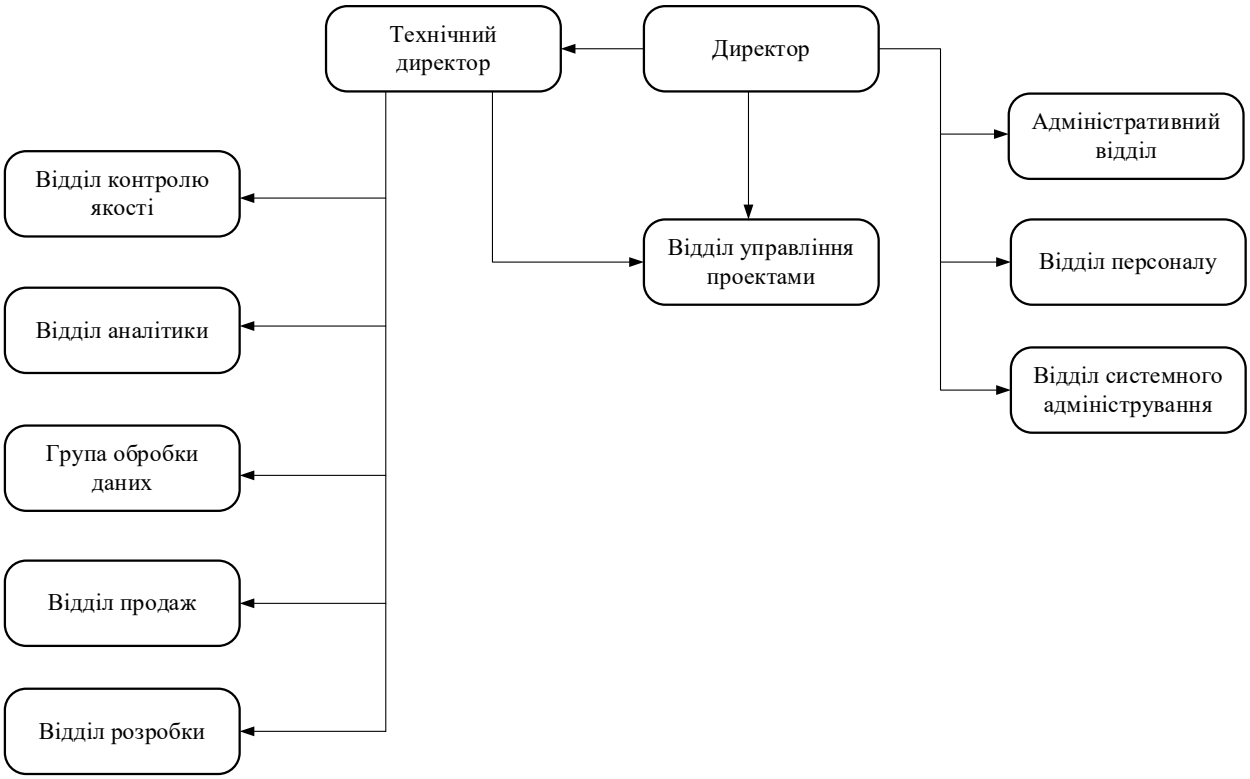

Рисунок 1.1 – Структура IT компанії , що розглядається

– відділ системного адміністрування включає DevOps-фахівців і системних адміністраторів, які займаються налаштуванням периферії та оточення для функціонування розроблюваного ПЗ;

– відділ розробки включає групу backend та frontend розробників;

– відділ аналітики включає фахівців, які відповідають за документацію до розроблюваного ПЗ та з’ясування вимог у замовників;

– відділ продажу включає фахівців, які займаються просуванням і продажем рішень з продуктової лінійки;

– адміністративний відділ включає бухгалтера, економіста, помічника директора та директора з загальних питань.

1.5.2 Огляд бізнес-процесів тестування програмного забезпечення

Будь-яка IT-компанія має у своїй структурі відділ тестування, який здебільшого займається тестуванням випускаємого ПЗ і відповідає за якість цього продукту. Організація цього відділу може відрізнятися залежно від

специфіки компанії, наприклад, відділ може поділятися на підрозділи, які окремо відповідають за безпеку, за автоматизацію, за навантажувальне тестування, ручне тестування тощо.

Прийнята компанія у своїй структурі має один відділ, що виконує основні функції тестування ПЗ, іменований як відділ контролю якості. Для того, щоб зануритися у роботи ВКЯ було вивчено наявні регламенти даного відділу. Частина регламентів відділу знаходиться у стадії розробки, але основна частина готова до використання і дозволяє вибудувати процес роботи відділу.

Вся документація відділу і компанії зберігається у так званій базі знань, яка розташована в Confluence. Confluence – це робочий простір, вікі-система для використання організацією з метою створення єдиної бази знань і обміну цими знаннями з іншими учасниками. Використання Confluence дозволяє створювати і редагувати сторінки будь-яким учасникам проєкту, проводити дискусії, коментувати робочі документи та керувати проєктами. Усі регламенти, що зберігаються в цій системі, написані учасниками ВКЯ і їхнім керівником і не мають копій в офіційних документах. Вони спрямовані на те, щоб нові співробітники швидше занурювалися в процеси відділу, а поточні співробітники зверталися до цих регламентів при виникненні спірних або конфліктних ситуацій на проєктах.

Основною метою вивчення цих регламентів було виділення процесів роботи ВКЯ. Для цього знадобилися такі регламенти:

– «Робота з відділом аналітики в рамках тестування фіче-сторінок» – описує бізнес-процеси тестування аналітичної документації залежно від типу проєкту;

– «Єдиний процес роботи на проєкті» – описує основні етапи роботи спеціаліста ВКЯ на проєкті;

– «Складання тест-плану на проєкті» – опис шаблону плану тестування та його значимості;

– «Визначення критеріїв готовності проєкту до релізу» – визначення і опис документа релізу та його шаблону;

– «Робота з ЖЦ задач на проєкті» – опис основних процесів задач, якими користується спеціаліст ВКЯ;

– «Робота з керівництвом користувача на проєкті» – опис значимості документа і етапів його складання;

– «Актуалізація тестової документації при re-testi помилок» – опис робіт, після яких необхідна актуалізація тестової документації;

– «Оцінка роботи спеціаліста ВКЯ на проєкті» – опис мінімальної, ймовірної та максимальної оцінки на роботи ВКЯ;

– «Проведення аудиту проєкту як відправної точки для занурення в проєкт» – опис значимості аудиту та порядку його проведення;

– «Попередня підготовка до ретроспективи» – опис значимості ретроспектив і шаблону для заповнення відповідної сторінки.

В результаті вивчення регламентів були виділені два типи бізнес-процесів, які присутні у відділі:

– процес роботи спеціаліста на проєкті;

– внутрішні процеси відділу.

Процес роботи спеціаліста на проекті регламентований і виконується кожним співробітником відділу, коли той заходить на проект. Частина процесів може відрізнятися залежно від типу проекту. У компанії в основному виділяють наступні типи проектів:

– продуктовий проект – проект, який входить у продуктовий лінійку компанії, такі проекти є проектами компанії, що продаються іншим компаніям;

– комерційний проект – проект замовної розробки, який надходить від певного замовника.

Незалежно від типу проекту виділяються етапи процесу роботи спеціалістів ВКЯ на проекті:

– підготовчий етап;

– аналіз вимог;

– розробка тестів;

– тестування;

– підготовка документації.

Про кожен етап процесу буде описано детальніше далі. Внутрішні процеси відділу пов’язані з роботами, які виконуються спеціалістами всередині відділу і не відносяться до проектів компанії. Ці роботи дозволяють оцінити роботу кожного спеціаліста ВКЯ і відділу в цілому.

До таких робіт відносять:

– проведення ретроспектив;

– збір метрик;

– заходи, спрямовані на досягнення індивідуального плану розвитку кожного співробітника.

1.5.3 Контроль за роботою учасників відділу тестування

Як згадувалося вище, частина регламентів відділу знаходиться на стадії розробки, тому для повного розуміння процесів і роботи відділу було вирішено провести інтерв’ювання спеціалістів ВКЯ. Метою даного інтерв’ю є отримання інформації про процеси відділу безпосередньо у його учасників. Інтерв’ю проводилося у форматі особистої бесіди окремо з кожним спеціалістом. Це було зроблено з метою, щоб учасники інтерв’ю не орієнтувалися на відповіді один одного. Процес інтерв’ювання дозволяє дізнатися думки спеціалістів про процес, з яким вони працюють, його недоліки і переваги, а також можливі побажання щодо зміни і/або покращення розглянутого процесу.

Підготовка до інтерв’ю складалася зі складання списку запитань, які були розділені на блоки відповідно до основних етапів процесів, що використовуються у роботі спеціалістів ВКЯ.

У таблиці А.1 (Додаток А) наведено список запитань, підготовлений для проведення інтерв’ю.

В інтерв’ю взяли участь 6 осіб, серед яких: керівник ВКЯ, 2 старших спеціалісти ВКЯ та 3 молодших спеціалісти ВКЯ.

В результаті проведеного інтерв’ю були враховані особливості процесів кожного етапу, які не були описані у регламентах або не були зрозумілі після їх прочитання. Виявлено їхні переваги та недоліки, отримано середню оцінку часу, який витрачається на виконання робіт на етапах тестування. Деякі учасники інтерв’ю також висловили свою думку щодо покращення та оптимізації існуючих процесів, яка буде врахована при розробці відповідних заходів.

Висновки до розділу:

У першому розділі було досліджено та проаналізовано ключові теоретичні аспекти тестування програмного забезпечення. Спочатку було визначено проблему забезпечення якості програмного продукту та роль тестування у виявленні та запобіганні дефектів ще на ранніх етапах розробки.

Було розглянуто життєвий цикл тестування програмного забезпечення, який охоплює етапи аналізу вимог, планування тестування, розробки тестів, виконання тестування та підготовки необхідної документації. Детально розглянуті ці етапи показали важливість їх взаємозв’язку та системності підходу до забезпечення якості програмного продукту.

Окрему увагу приділено видам тестування програмного забезпечення, таким як модульне, інтеграційне, системне та приймальне тестування. Також було проаналізовано види тестової документації, яка супроводжує процес тестування, забезпечує його прозорість і контрольованість. Здійснено аналіз та моделювання бізнес-процесів тестування ПЗ, де розглянуто діяльність IT-компаній, особливості організації роботи відділу тестування та контроль якості виконання завдань. Цей аналіз дозволив виявити критичні точки процесів тестування, що впливають на ефективність роботи команди та кінцеву якість продукту.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЯ БІЗНЕС-ПРОЦЕСІВ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Після вивчення регламенту та проведення аналізу відповідей із інтерв'ю, проведеного зі спеціалістами відділу, було розпочато роботу з моделювання отриманих бізнес-процесів.

В якості інструменту для моделювання бізнес-процесів була обрана система для моделювання процесів у нотації BPMN – Bizagi. Bizagi – це BPM-система, спрямована на моделювання, виконання, автоматизацію та аналіз бізнес-процесів. Система Bizagi включає 3 модулі для повноцінного налаштування процесів [9]:

– modeler – повнофункціональне середовище моделювання процесів у нотації BPMN;

– studio – середовище розробки бізнес-процесів;

– engine – середовище виконання процесів, яке доступне користувачам у будь-якому браузері з будь-якого пристрою.

Для моделювання розглянутих у цій роботі процесів достатньо функцій модуля Modeler. Цей модуль призначений для моделювання послідовності дій та подій. Крім того, Bizagi Modeler підтримує імітаційне моделювання, експорт моделі та результатів моделювання у різні формати файлів. Модуль необхідний тим, кому потрібно проаналізувати та оптимізувати бізнес-процес.

2.1 Моделювання бізнес-процесів тестування програмного забезпечення

Як згадувалося вище, роботи спеціалістів відділу на проектах мають бізнес-процес, який поділяється на наступні етапи:

– підготовчий етап;

– аналіз вимог;

– розробка тестів;

– тестування;

– підготовка документації.

2.1.1 Бізнес-процес підготовчого етапу

Підготовчий етап описує процес входження спеціаліста у проект, тобто необхідний для того, щоб спеціаліст занурився в проект і оцінив його стан. Початковою подією цього етапу є готовність проекту до виконання робіт та сформована команда проекту. Процес підготовчого етапу не залежить від типу проекту, тобто всі дії процесу виконуються на будь-якому проекті компанії. Модель бізнес-процесу підготовчого етапу наведена на рисунку 2.1.

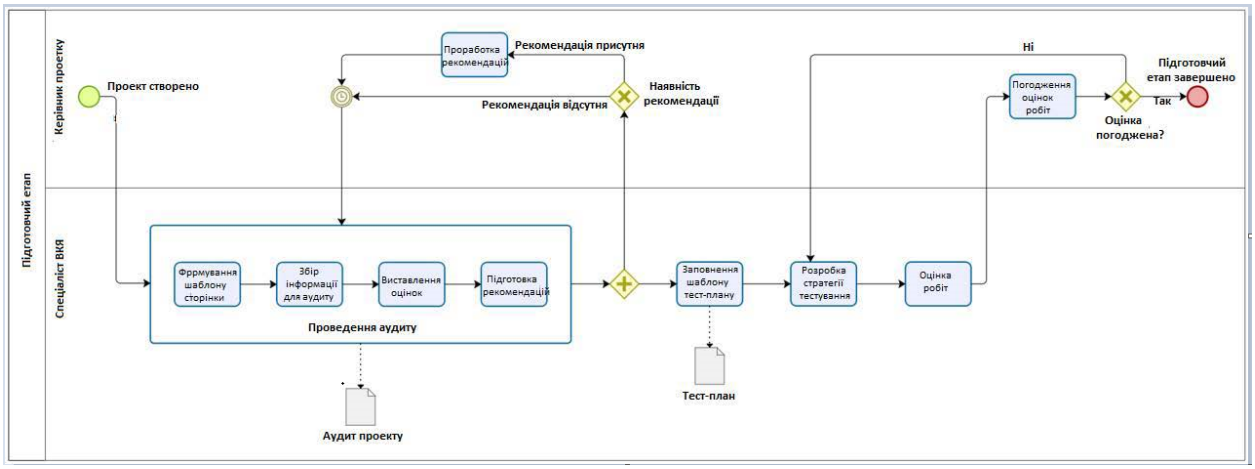

Рисунок 2.1 – Бізнес-процес підготовчого етапу

Як тільки проект готовий до робіт і до нього підключають спеціаліста відділу контролю якості, йому необхідно оцінити стан проекту, тобто провести його аудит. Аудит проекту в компанії проводиться з метою показати загальну картину стану проекту команді та іншим зацікавленим особам. Аудит проекту - це підпроцес, який включає такі дії:

– формування шаблону сторінки аудиту - аудит ведеться в Confluence, тому шаблон аудиту зберігається там же;

– збір інформації для аудиту - аудит складається з кількох блоків, для кожного блоку необхідно отримати потрібну інформацію і заповнити відповідний блок;

– виставлення оцінок - оцінки виставляються у вигляді світлофора: червоний, жовтий і зелений. Для того, щоб виставити необхідну оцінку, треба проаналізувати зібрану для аудиту інформацію;

– підготовка рекомендацій - за результатами аналізу зібраної інформації та виставлених оцінок спеціалісту ОКК потрібно скласти список рекомендацій для керівника проекту з метою усунення виявлених недоліків проекту.

Як тільки аудит проведено і складено список рекомендацій, їх направляють керівнику проекту, який займається опрацюванням цих рекомендацій. Можлива також ситуація, коли рекомендацій немає, і тоді керівнику проекту не потрібно їх опрацьовувати. Як видно з моделі бізнес-процесу, проведення аудиту - циклічний процес, який спрацьовує за таймером, оскільки аудит проводиться регулярно - раз на місяць, щоб відслідковувати стан проекту не лише при вході спеціаліста в проект, але й протягом усього ведення проекту.

Після проведення аудиту і паралельно з опрацюванням рекомендацій керівником проекту, спеціаліст ОКК займається заповненням шаблону плану тестування, який також зберігається в Confluence. Цінність цього документа в тому, що він показує основні роботи спеціаліста на проекті, які будуть проведені. У майбутньому, звертаючись до цього документа, можна відновити історію робіт спеціаліста та оцінити його роботу. Для коректного і повного заповнення цього документа спеціалісту ОКК необхідно розробити стратегію тестування, тобто план необхідних робіт. Перелік робіт індивідуальний залежно від проекту і може відрізнятися, але в основному містить список

робіт, які співпадають із наступними етапами процесу, які будуть розглянуті далі.

У компанії ведеться облік трудовитрат співробітників, тому оцінка робіт є важливою складовою не лише для звіту перед керівництвом, а й для планування подальших робіт. Фактичні та плановані оцінки робіт дозволяють керівникам оцінити вартість проекту і допомагають у плануванні майбутніх робіт і проектів. Оцінкою робіт з тестування займається спеціаліст ОКК, виходячи зі свого досвіду або пам'ятки-підказки, що є в регламенті відділу, де прописані мінімальні, ймовірні та максимальні оцінки основних робіт відділу.

Оцінка робіт обов'язково має бути погоджена з керівником проекту. За результатами погодження оцінка може бути не погоджена, тоді спеціалісту відділу необхідно переглянути складену стратегію тестування і переглянути оцінку робіт. Якщо ж оцінка погоджена з боку керівника проекту, то підготовчий етап завершується і починаються роботи наступного етапу

2.1.2 Бізнес-процес етапу аналізу вимог

Основна мета етапу аналізу вимог – це вивчити аналітичну документацію, яка може бути представлена у вигляді вимог або фіча-сторінок, виявити проблемні місця в цій документації та скласти план перевірок. Аналітичну документацію готує відділ аналітики, і як тільки документація готова – починається процес її тестування.

Варто відзначити, що процес аналізу вимог залежить від типу проекту: продуктовий або комерційний. На рисунку 2.2 представлено змодельований бізнес-процес для комерційного проекту. У цьому процесі беруть участь лише спеціаліст відділу аналітики та спеціаліст ВКЯ.

Як тільки аналітична документація готова до тестування, вона передається спеціалісту ВКЯ, який проводить тестування аналітики.

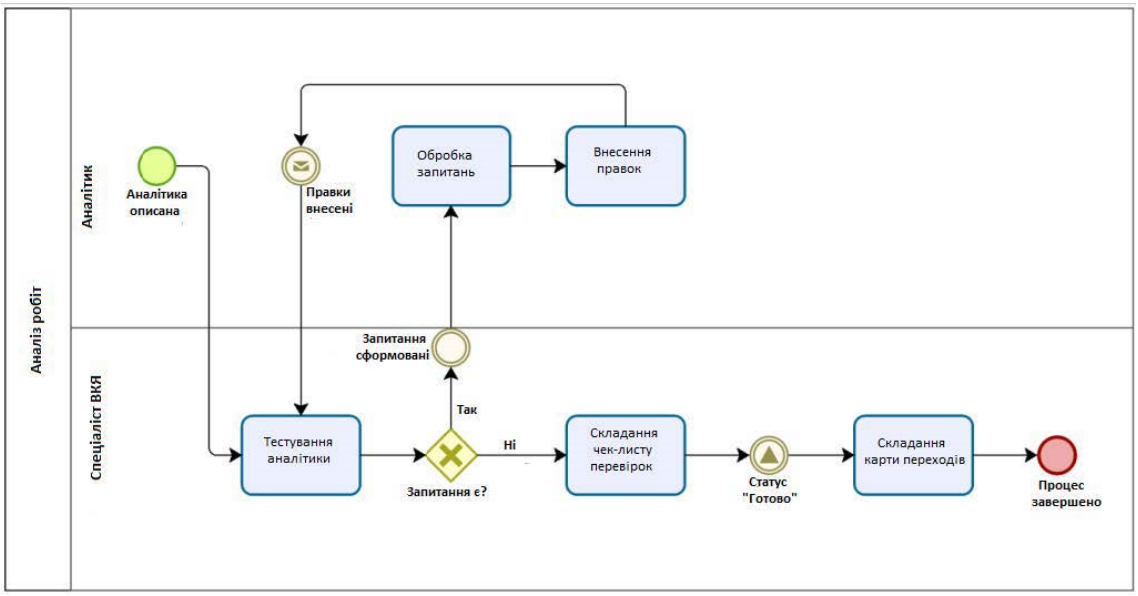

Рисунок 2.2 – Бізнес-процес етапу аналізу вимог для комерційного проекту

Тестування аналітики передбачає уважне прочитання описаних вимог та виявлення невідповідностей або суперечностей у них. Якщо такі недоліки в аналітичній документації виявлені, під час тестування формується пул питань або додаються коментарі до документа. Оскільки аналітична документація ведеться в Confluence, там підтримується можливість коментування частин тексту, що дозволяє спеціалісту відділу аналітики відслідковувати всі зміни на сторінці.

Після формування питань сторінка документації передається відповідному аналітику для опрацювання. Яким чином аналітик буде обробляти ці питання - це окремий процес відділу аналітики, на який спеціаліст ВКЯ не має впливу. Аналітик може проводити дзвінки або зустрічі із замовником, уточнювати питання з розробниками. Після обробки всіх питань аналітик вносить зміни до документації і повідомляє про це спеціаліста ВКЯ. Процес тестування аналітики повторюється, і таких ітерацій може бути декілька.

Лише після вирішення всіх питань спеціаліст ВКЯ переходить до формування чек-листа перевірок. Чек-лист - це нумерований перелік основних

перевірок, які необхідно виконати під час тестування системи. Після написання чек-листа аналітична документація вважається готовою, і спеціаліст ВКЯ ставить статус тестування «Готово».

Після перевірки аналітичної документації спеціаліст ВКЯ складає карту переходів проекту за фічами, описаними в документації. Карта переходів будується за допомогою MindMap і дає змогу наочно побачити весь проект та його основні функції. Іноді карта переходів замінює складання тест-кейсів для тестування, про що буде розказано в наступному бізнес-процесі.

Далі буде розглянуто процес етапу аналізу вимог для продуктового проекту, модель якого наведена на рисунку 2.3. Цей процес відрізняється тим, що в ньому бере участь ще одне зацікавлене лице - Product Owner. Product Owner - це спеціаліст, який володіє продуктом та відповідає за його розвиток.

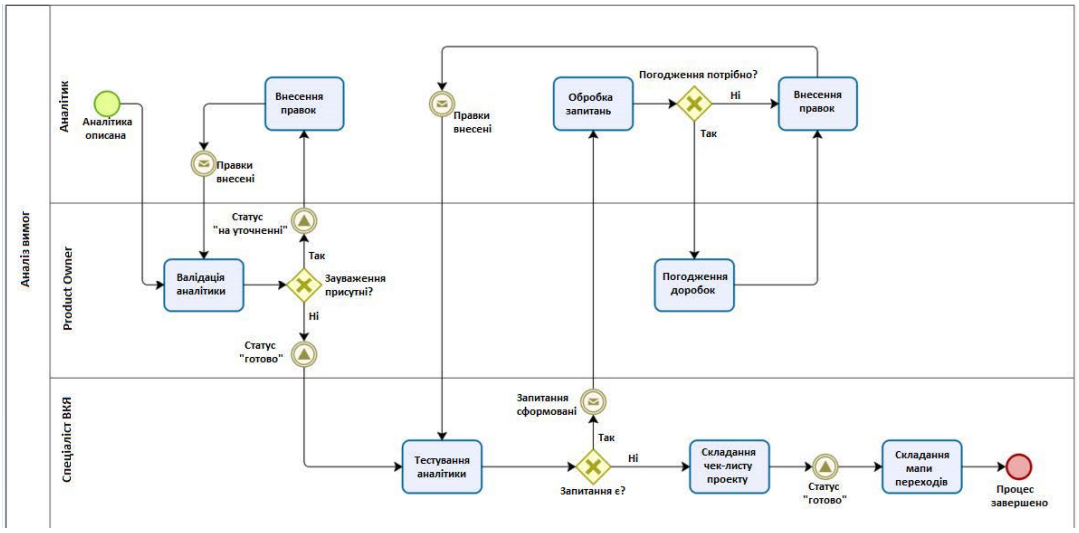

Рисунок 2.3 – Бізнес-процес етапу аналізу вимог для продуктового проекту

Після того, як аналітична документація готова до тестування, вона спочатку передається Product Owner, який проводить валідацію написаної аналітики. Під валідацією розуміється прочитання документації та написання зауважень у разі виявлення невідповідностей необхідному функціоналу системи. Якщо Product Owner знаходить зауваження, він залишає їх у вигляді

коментарів на сторінці аналітики та повертає аналітику на доопрацювання. Сигналом про наявність зауважень від Product Owner є встановлення відповідного статусу «На уточненні».

Аналітик вносить правки згідно з отриманими зауваженнями і, повідомивши Product Owner, відправляє аналітику на повторну валідацію. Таких ітерацій може бути кілька. Коли Product Owner приймає аналітику, він встановлює статус «Готово».

Статус «Готово» від Product Owner є сигналом для спеціаліста ВКЯ, що аналітичну документацію можна брати в роботу, тобто починати тестування. Процес тестування аналітики спеціалістом ВКЯ нічим не відрізняється від процесу для комерційного проекту: формується пул питань, які передаються спеціалістам відділу аналітики.

Спеціалісти відділу аналітики опрацьовують отримані питання. Сам процес опрацювання питань є внутрішнім процесом відділу аналітики, на який спеціаліст ВКЯ не впливає. Якщо внесені зміни зачіпають функціональну частину, аналітик погоджує їх з Product Owner, оскільки саме він відповідає за продукт. Після узгодження або відхилення змін аналітик вносить корективи до документації і передає її знову на тестування спеціалісту ВКЯ.

Як і в комерційному проєкті, кількість таких ітерацій може бути кілька. Лише після вирішення всіх питань спеціаліст ВКЯ приступає до складання чек-листа перевірок.

Подальші дії спеціаліста ВКЯ ідентичні процесу в комерційному проєкті: він встановлює статус «Готово», що означає, що документація повністю протестована, складений мінімальний чек-лист майбутніх перевірок, а також створена карта переходів проєкту, яка відображає основну функціональність системи.

Варто зазначити, що всі роботи, проведені на цьому етапі, погоджуються з керівником проєкту на підготовчому етапі та відображаються в плані тестування, який було складено раніше.

2.1.3 Бізнес-процес етапу розробки тестів

На основі чек-листів, які були складені на етапі аналізу вимог, розробляється тестова документація така як тест-кейси. Тест-кейси є основною документацією фахівця, оскільки вони грають одну з ключових ролей при тестуванні продукту. Тому цей етап є важливою частиною роботи спеціаліста на проекті. Модель бізнес-процесу розробки тестів представлена на рисунку 2.4.

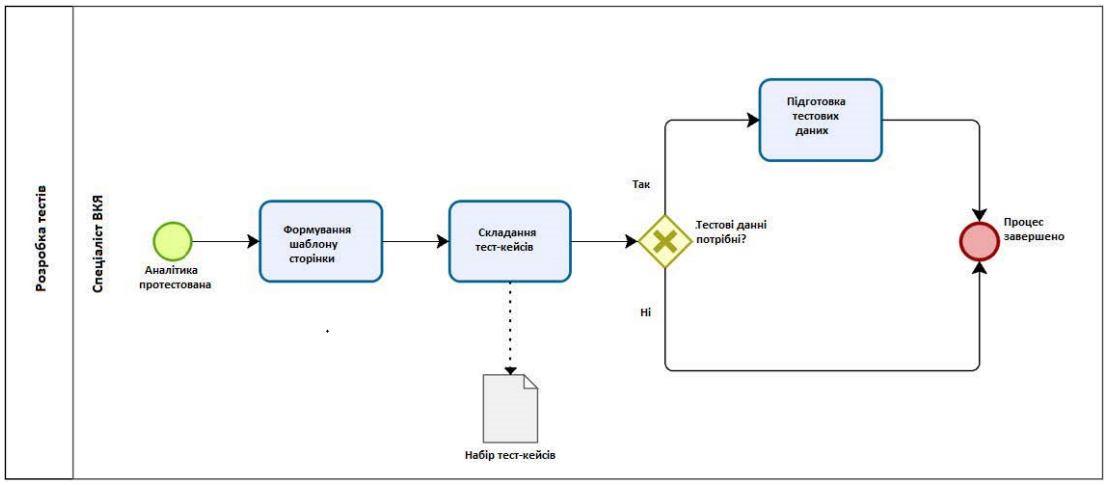

Рисунок 2.4 – Бізнес-процес етапу розробки тестів

Тест-кейси складаються та зберігаються в Confluence у вигляді таблиці, тому для початку необхідно сформувати та заповнити шаблон для збереження тест-кейсів. Після формування шаблону починається етап складання та розробки тест-кейсів. Хоч цей процес і описаний як одна дія, він досить тривалий і вимагає повної концентрації від спеціаліста ВКЯ.

Цей вид документації є дуже важливим, оскільки він призначений не лише для спеціаліста, який безпосередньо складає тест-кейси, але й для нових працівників, які занурюються у проєкт, або для допомоги під час проведення тестування. Тому тест-кейси повинні бути повними і описувати достатню кількість перевірок.

Для проведення тестування також можуть знадобитися різні тестові дані. Необхідність і структура тестових даних значною мірою залежить від

специфіки проєкту. Наприклад, якщо це проєкт розробки геоінформаційної системи, то необхідно підготувати дані, які відображатимуться на карті проєкту - це можуть бути об’єкти будівель та інших споруд. Або ж у якості тестових даних можуть використовуватись різноманітні вкладення у вигляді файлів різних форматів: зображення, документи тощо. Всі ці дані повинні відповідати розробленому тест-кейсу.

Процес розробки тестів вважається завершеним, якщо написані всі тест-кейси та підготовлені всі тестові дані для цих тест-кейсів.

2.1.4 Бізнес-процес етапу тестування

Етап тестування передбачає безпосереднє тестування продукту та виявлення помилок. Тестування на проекті може проходити кілька разів, що залежить переважно від тривалості проекту. В якому вигляді проходитиме етап тестування залежить від виду застосовуваного на проекті тестування. Варто згадати, що кожен із видів може зустрітися як на комерційному проекті, так і на продуктовому. На рисунку 2.5 представлений процес етапу тестування.

Процес етапу тестування починається одразу після того, як буде готова тестова документація та стенд із розроблюваною системою буде готовий до тестування. Далі, залежно від бізнес-цінності тестування, обирається вид тестування. Вид тестування може бути різним, але найчастіше у компанії використовують регресійне тестування, тестування за критичним шляхом, димове або приймальне тестування.

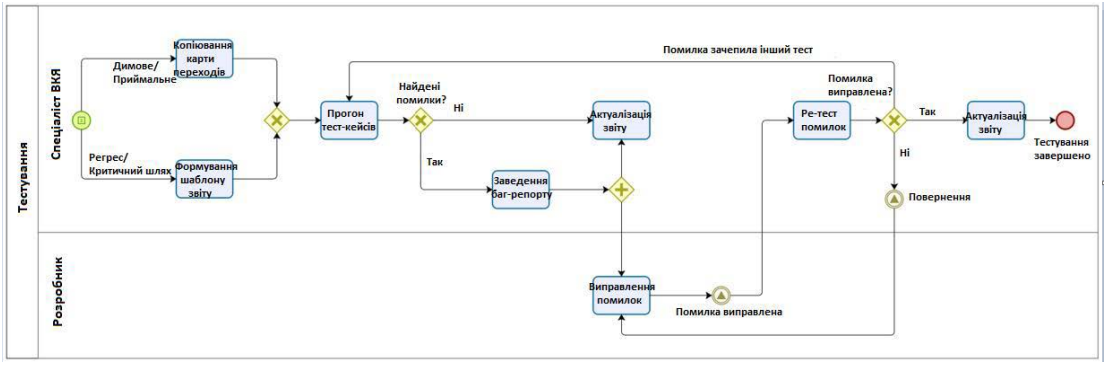

Рисунок 2.5 – Бізнес-процес етапу тестування

Регресійне та тестування за критичним шляхом передбачають великий обсяг перевірок і охоплюють всю функціональність системи. Димове та приймальне тестування необхідні для перевірки основних функцій системи, якими користуватиметься кінцевий користувач.

Якщо для системи необхідне регресійне тестування або тестування за критичним шляхом, необхідно сформувати шаблон для ведення звіту про тестування. Такий документ ведеться в Confluence і заповнюється вручну спеціалістами ВКЯ. Він містить список тест-кейсів, які необхідно пройти під час тестування, та результати їх виконання. На етапі формування шаблону звіту потрібно вказати вид тестування, обрати необхідні тест-кейси та перенести їх назви на сторінку звіту.

Якщо ж для системи потрібне димове або приймальне тестування, створювати звіт про тестування необов’язково, достатньо просто скопіювати карту переходів MindMap, сформовану на етапі аналізу вимог. Як уже було зазначено, ця карта містить основний функціонал системи, а димове та приймальне тестування передбачають перевірку базового функціоналу, тому карта може замінити звіт про проходження тестування.

Після підготовки сторінки та карти переходів для ведення звітів про тестування починається процес безпосереднього тестування або прогону тест-кейсів. Під час тестування виявляються помилки системи, які називають багами. Якщо знайдено помилки, потрібно створити так звані баг-репорти. Баг-репорт - це опис виникнення помилки, який передається розробнику для її усунення. Баг-репорти створюються в системі Jira і мають статуси, що показують, на якому етапі знаходиться помилка: у роботі у розробників, в тестуванні тощо.

Як тільки баг-репорт створено або тест-кейс пройдено без помилок, необхідно актуалізувати звіт про тестування або карту. Актуалізація означає проставлення статусу проходження тесту (наприклад, «пройдено» або «провалено») і, у разі провалу, додавання посилання на відповідний баг-репорт.

Паралельно розробник приступає до виправлення баг-репорту. Процес виправлення помилки - це внутрішній процес відділу розробки і кожного розробника, який не впливає на описуваний бізнес-процес. Коли розробник виправляє помилку, він змінює її статус на «В тестуванні» і передає її спеціалісту ВКЯ для проведення ре-тестування або перевірки функціональності з метою підтвердити, що помилка дійсно усунена.

Часто буває так, що під час ре-тестування помилка залишається не виправленою, тоді баг-репорт повертається розробнику на доопрацювання. Також виправлена помилка може зачепити іншу функціональність системи, у цьому випадку необхідно повернутися до процесу прогону тест-кейса, вибрати тест-кейс, який міг бути порушений, повторно його виконати і, якщо потрібно, створити новий баг-репорт та оновити звіт про тестування або карту.

Після усунення помилки також потрібно актуалізувати звіт про тестування або карту, щоб вони завжди відображали поточний стан продукту. Це необхідно як керівнику проєкту для планування майбутніх робіт, так і замовнику.

2.1.5 Бізнес-процес етапу підготовки документації

Етап підготовки документації є останнім етапом процесу тестування і зазвичай виконується після того, як будуть проведені всі роботи на попередніх етапах.

Етап підготовки документації може відрізнятися залежно від замовника та його потреб, може відрізнятися з точки зору набору необхідних для передачі документів. У модель були додані основні документи, які найчастіше вимагають замовники.

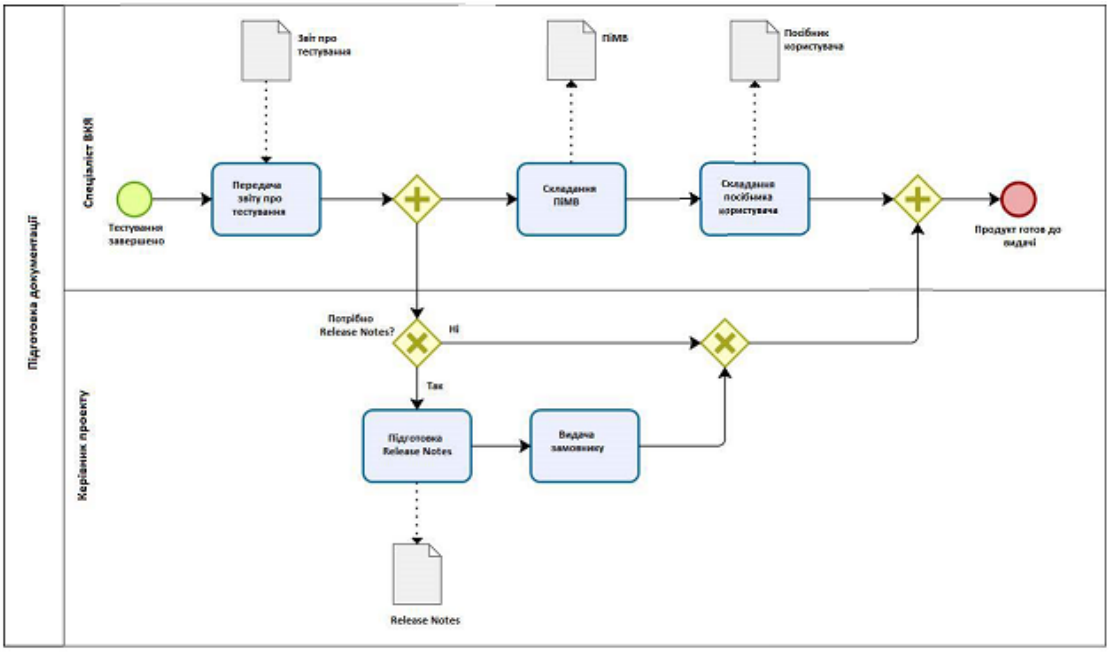

Рисунок 2.6 – Бізнес-процес етапу підготовки документації

Після проходження тест-кейсів на етапі тестування складається звіт про тестування. Як тільки тестування завершується, цей звіт передається керівнику проєкту для оцінки готовності проєкту до релізу. Після цього розпочинається робота над документацією, яку виконують різні фахівці, тому цей процес відбувається паралельно.

Керівник проєкту, за потреби, складає Release Notes. Зазвичай така необхідність виникає на довготривалих комерційних проєктах або продуктових проєктах. Release Notes — це документ, який містить інформацію про виконані завдання або виправлені помилки, що демонструє замовнику, які роботи були проведені і які оновлення будуть розгорнуті на робочих стендах замовника.

Після того, як цей документ готовий, він передається замовнику.

Спеціаліст ВКЯ займається складанням ПіМВ - програми та методики випробувань. Цей документ показує, які перевірки були проведені під час тестування системи, зазвичай він містить набір основних тест-кейсів, необхідних для проведення тестування. Окрім ПіМВ, спеціаліст ВКЯ також

відповідає за написання Керівництва користувача. Цей документ містить опис системи та інструкцію щодо її використання.

Як тільки всі документи будуть оформлені, вони передаються замовнику на приймання. Замовник перевіряє ці документи і може повернути їх на доопрацювання, сформувавши відповідні зауваження. Процес вважається завершеним, коли всі необхідні документи прийняті замовником.

2.2 Налаштування бізнес-процесів тестування програмного забезпечення для імітаційного моделювання

Перш ніж розпочинати аналіз змодельованих бізнес-процесів, необхідно провести імітаційне моделювання. Метою імітаційного моделювання є відтворення досліджуваної системи, її імітація в часі. Імітаційне моделювання дозволяє проводити експерименти з системою для отримання інформації про неї, що допоможе під час подальшого аналізу та оптимізації.

Для проведення імітаційного моделювання в системі Bizagi Modeler необхідно визначити ресурси, які беруть участь у даному процесі. Як ресурси були обрані всі учасники розглянутих процесів. Визначення ресурсів у Bizagi показано на рисунку 2.7.

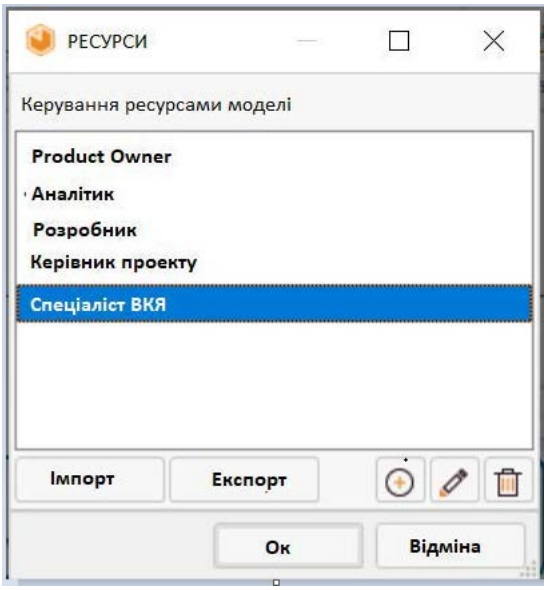

Рисунок 2.7 – Ресурси бізнес-процесів

Далі були налаштовані параметри для кожного елемента в усіх розглянутих моделях.

Для дій моделі задаються налаштування часу обробки та ресурс, який виконує цю роботу. Час обробки дій береться з регламенту «Оцінка роботи спеціаліста ОКК на проекті» або з досвіду роботи співробітників на проекті.

Кожному ресурсу необхідно встановити витрати за годину його роботи. Витрати за годину роботи беруться середні за 2024 рік. На рисунку 2.8 показано налаштування витрат за годину роботи кожного ресурсу.

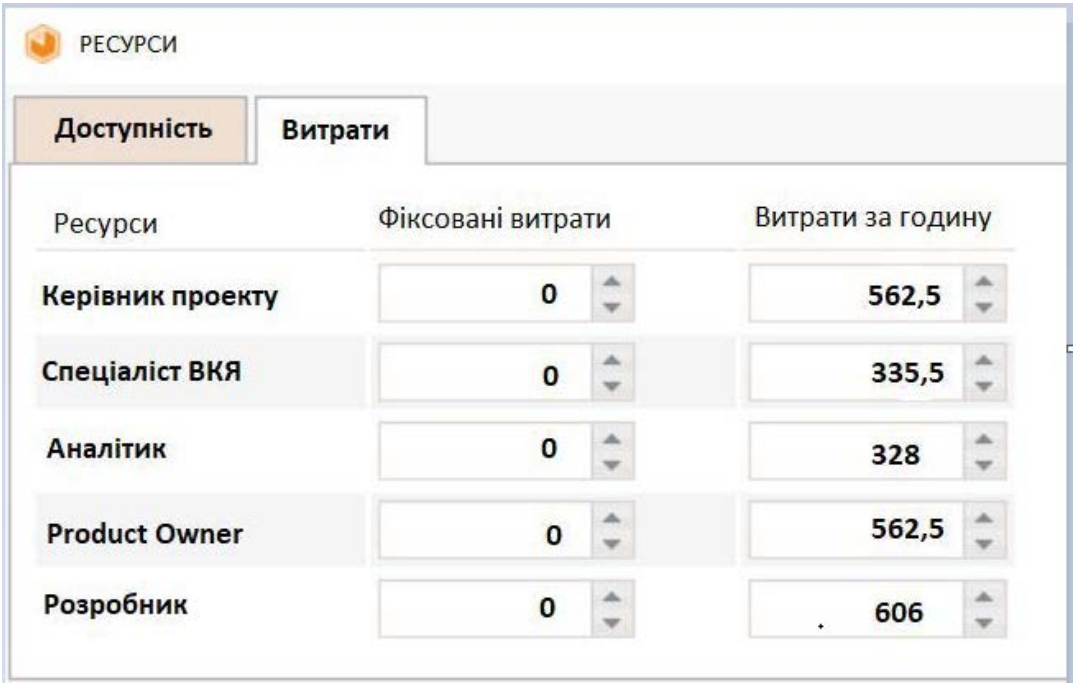

Рисунок 2.8 – Витрати роботу ресурсів

Шлюзи моделі налаштовуються як відсоток ймовірності настання наступної події. Відсоток ймовірності настання подій береться на основі досвіду співробітників компанії, які беруть участь у розглянутих процесах.

У Додатку Б наведені усі налаштування імітаційних моделей.

2.3 Аналіз результатів моделювання бізнес-процесів тестування програмного забезпечення

Після проведення імітаційного моделювання були отримані результати у вигляді таблиці затрат на виконання кожного етапу процесу та часу, який витрачається на кожну дію етапу. Далі розглянемо кожен етап детальніше.

На моделях за результатами імітаційного моделювання буде відображена легенда, де значення червоного квадрата означає кількість виконаних дій, а значення синього квадрата - це загальний час обробки дії.

Результат імітаційного моделювання підготовчого етапу представлений на рисунку 2.9.

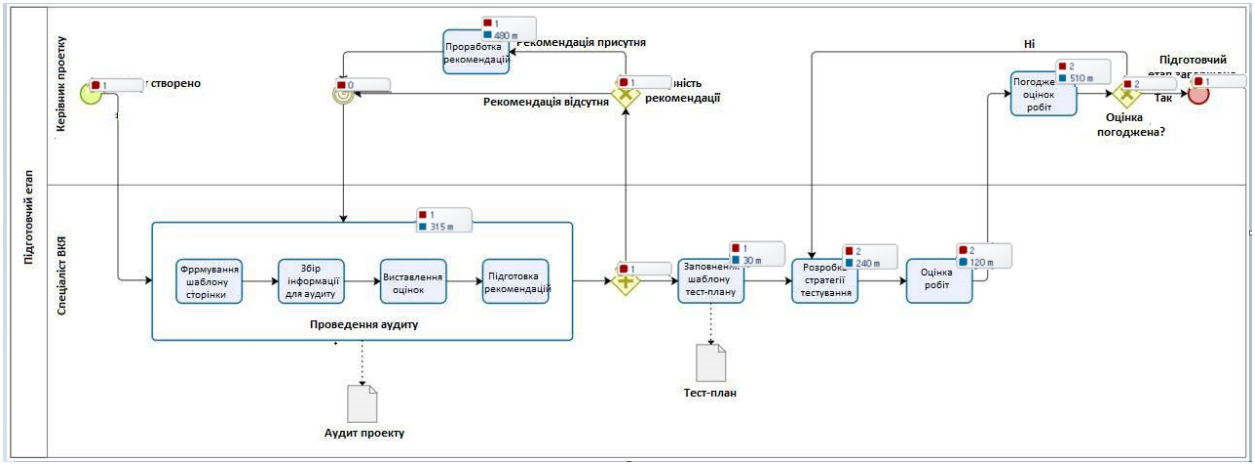

Рисунок 2.9 – Результати імітаційного моделювання підготовчого етапу

За результатами імітаційного моделювання можна відзначити, що велика кількість часу витрачається спеціалістами ВКЯ на проведення аудиту по проекту. Враховуючи, що аудит проводиться щомісяця, а не один раз, є сенс автоматизувати процес збору інформації для аудиту. Це дозволить скоротити час роботи спеціаліста і прискорити проходження всього підготовчого етапу.

У даному процесі спостерігається цикл: якщо оцінка не погоджується керівником, це призводить до додаткового проходження дій процесу і, як наслідок, збільшення часу виконання процесу та витрат на нього. Щоб уникнути таких ситуацій, можна змінити процес, додавши ще одну дію –

озвучення меж бюджету на виконання робіт перед складанням тест-плану та стратегії тестування. У такому разі спеціаліст ВКЯ буде орієнтуватися на задані цифри і намагатиметься не виходити за їх межі, а також будуватиме стратегію тестування з урахуванням рамок бюджету.

Заходи з оптимізації підготовчого етапу:

– автоматизація процесу збору інформації для аудиту;

– додавання дії з інформування спеціалістів ВКЯ про бюджет проекту до складання тест-плану та розробки стратегії тестування.

Результат імітаційного моделювання етапу аналізу вимог для комерційного проекту представлений на рисунку 2.10.

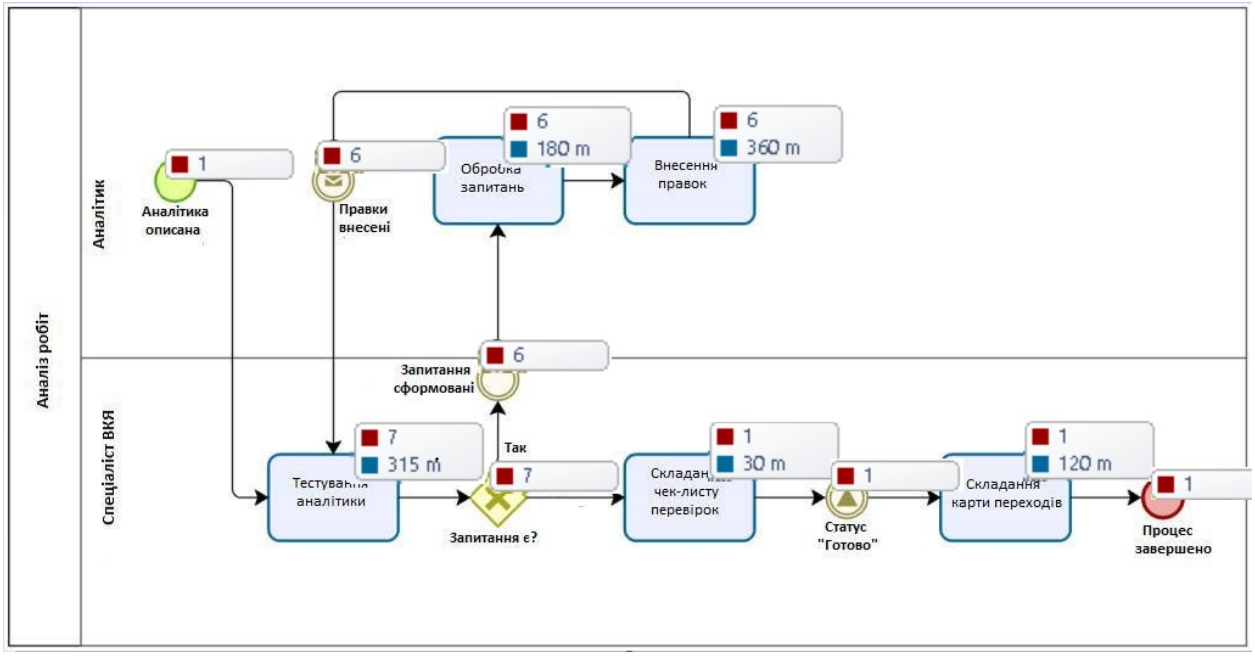

Рисунок 2.10 – Результати імітаційного моделювання етапу аналізу вимог комерційного проекту

За результатами імітаційного моделювання помітна циклічність повернень аналітики на доопрацювання — це ситуація, коли під час тестування виникають питання або зауваження до написаної документації. На моделі видно, що кількість повернень в середньому становить 6, що є досить вагомою цифрою і впливає на тривалість бізнес-процесу розглядуваного етапу. Ця ситуація не підлягає оптимізації та автоматизації через те, що

документацію пишуть аналітики, і під час написання можуть бути фактори, які впливають на кількість питань і повернень — наприклад, аналітик не врахував частину функціональності системи або помилився при написанні сценаріїв використання. Звичайно, залежно від виду аналітики кількість таких повернень може бути як більшою, так і меншою, або повернень взагалі може не бути. У цій моделі розглянуто найчастіший у роботі приклад. Дії зі складання чек-листа та карти переходів також не підлягають оптимізації.

На завершення аналізу бізнес-процесу етапу аналізу вимог на комерційному проекті можна сказати, що порядок дій, який виконується під час роботи на розглянутому етапі, не підлягає оптимізації і не потребує її.

Результат імітаційного моделювання етапу аналізу вимог для продуктового проекту представлений на рисунку 2.11.

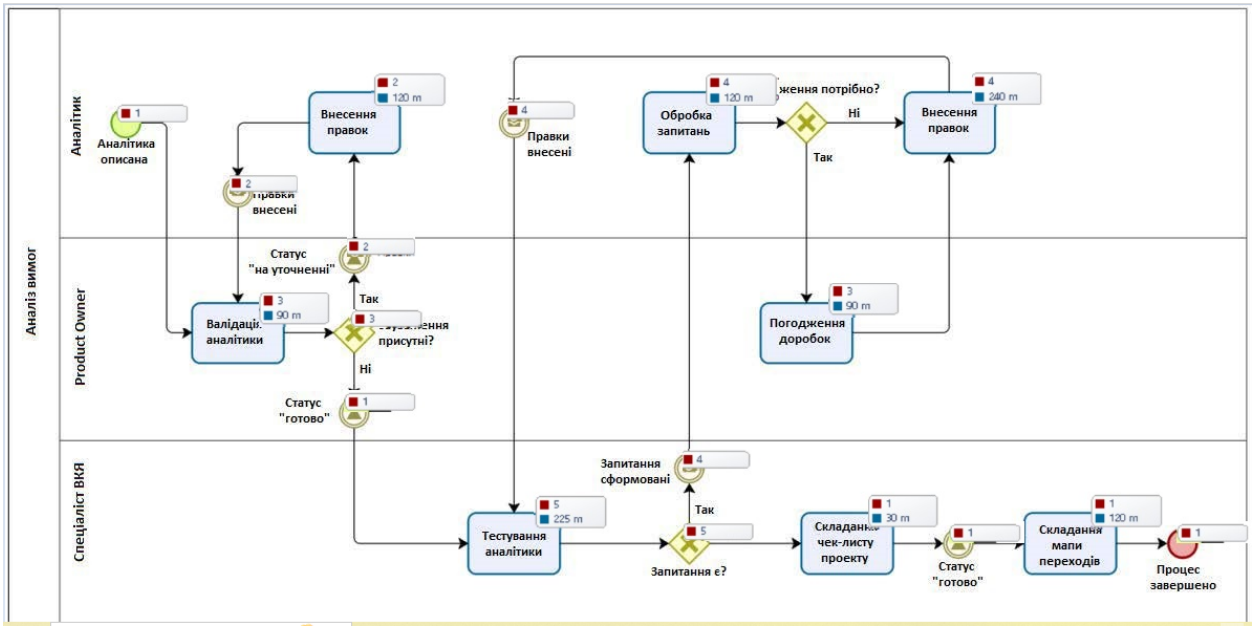

Рисунок 2.11 – Результати імітаційного моделювання етапу аналізу вимог продуктового проекту

За результатами імітаційного моделювання можна помітити, що валідація аналітики Product Owner відбувається на початкових етапах, тобто одразу після її фактичного написання. Це призводить до повернень аналітики на доопрацювання через зауваження, залишені Product Owner. На імітаційній моделі спостерігається два таких повернення. Після валідації аналітика

переходить на тестування, і, незважаючи на те, що аналітика вже проходила валідацію, питання з боку тестування або ВКЯ залишаються, а отже зберігаються й повернення, аналогічно комерційним проектам. Найімовірніше, таких повернень буде менше, але вони не виключаються повністю і середня їх кількість становить чотири. Незважаючи на зменшення кількості повернень, вони стають дорожчими через те, що будь-яка внесена зміна потребує погодження з Product Owner. За результатами імітаційної моделі видно, що таких погоджень було 3 з 4 повернень, що є вагомою цифрою.

З метою зменшення циклічності повернень аналітики та збереження сенсу існування цього етапу було запропоновано змінити порядок дій процесу, а саме перенести валідацію Product Owner після тестування аналітики спеціалістом ВКЯ. Це рішення дозволить Product Owner бачити та валідовувати повністю готову і протестовану аналітику, оскільки часто трапляється, що аналітики під час написання документації упускають частину функціоналу, яка виявляється на етапі тестування спеціалістом ВКЯ. Окрім цього, таке рішення дозволить скоротити кількість повернень від Product Owner і позбавить погодження доопрацювань. А дії зі складання чек-листів і карти переходів можна виконувати паралельно з валідацією Product Owner. У разі зміни функціоналу після валідації під час повторного тестування можна актуалізувати чек-листи і карту, що не займе багато часу.

Заходи з оптимізації етапу аналізу вимог продуктового проекту:

– перенесення дії валідації аналітики після тестування та зміна процесу з урахуванням цього перенесення;

– складання чек-листів і карти переходів паралельно з валідацією Product Owner.

Результат імітаційного моделювання етапу розробки тестів представлений на рисунку 2.12.

Зберігання та розробка тест-кейсів спеціалістами ОКК здійснюється в Confluence. Сторінка в Confluence являє собою формат текстового документа,

де у вигляді таблиці зберігаються розроблені спеціалістом тест-кейси. Перед початком розробки тест-кейсів необхідно сформувати шаблон сторінки в Confluence. Варто зазначити, що для кожної сторінки аналітичної документації створюється своя сторінка цього шаблону, тому залежно від обсягу проекту кількість таких сторінок може варіюватися.

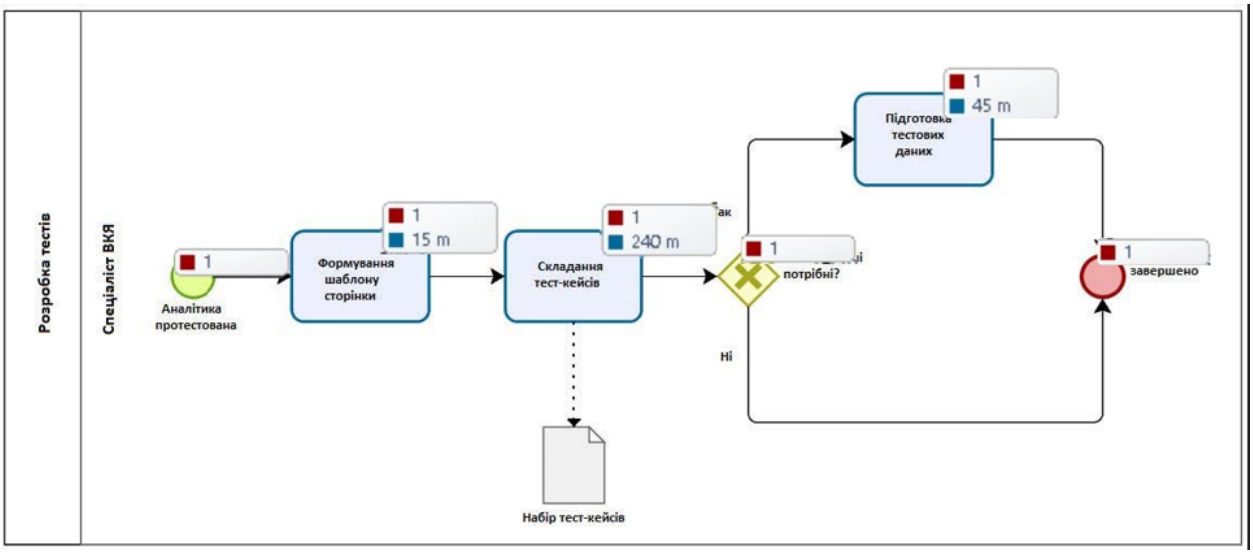

Рисунок 2.12 – Результати імітаційного моделювання етапу розробки тестів

Для імітаційного моделювання були виставлені налаштування для формування одного такого шаблону.

Існує велика кількість спеціалізованих систем для зберігання та складання тестової документації, у тому числі й тест-кейсів. Слід відзначити, що Confluence не є такою системою, тому необхідно розглянути інші варіанти для розробки та зберігання тест-кейсів. Розгляд таких варіантів дозволить автоматизувати складання звітів і структурувати зберігання тест-кейсів. Оскільки основною системою для ведення завдань у проектах є Jira, були розглянуті варіанти спеціалізованих плагінів для управління тестуванням у Jira.

Інструмент управління тестуванням для Jira - це додаток для управління тестуванням та забезпечення якості, який можна використовувати всередині Jira. Ці надбудови Jira зазвичай призначені для допомоги групам контролю

якості в реалізації функцій автоматизації тестування, обміну інформацією в режимі реального часу та можливості повторного використання тестових сценаріїв [10].

Для визначення того, який інструмент краще використовувати, були виділені такі критерії: користувацький інтерфейс, зручність використання, інтеграція, ціна.

Інструменти управління тестуванням можуть суттєво відрізнятися за ціною: деякі пропонують безкоштовні або недорогі варіанти, тоді як інші можуть мати високі початкові витрати або періодичні збори. Деякі системи управління тест-кейсами для Jira стягують плату залежно від кількості користувачів або проектів, інші ж можуть пропонувати необмежений доступ.

За результатами проведеного порівняльного аналізу було обрано інструмент управління тестуванням - Xray. Він спочатку інтегрований у Jira і використовує власні типи задач, що дозволяє централізовано виконувати роботи в одному місці. Xray зручний у використанні, допомагає структуризувати тест-кейси, а також надає можливість відслідковувати звіти про тестування в режимі реального часу та автоматизувати експорт інших документів.

Заходи з оптимізації етапу розробки тестів:

– впровадження Xray у процес управління тестуванням;

– складання регламентів і інструкцій щодо роботи з Xray.

Імітаційне моделювання етапу тестування проводилося 2 рази. У першому випадку розглядався процес, коли проводиться димове або приймальне тестування, тобто в якості звіту використовується карта переходів. А в другому випадку розглядався процес, коли проводиться регресійне тестування або тестування за критичним шляхом з повноцінним звітом та прогоном розроблених на попередньому етапі тест-кейсів.

Результати імітаційного моделювання етапу тестування для димового або приймального тестування наведені на рисунках 2.13 та 2.14.

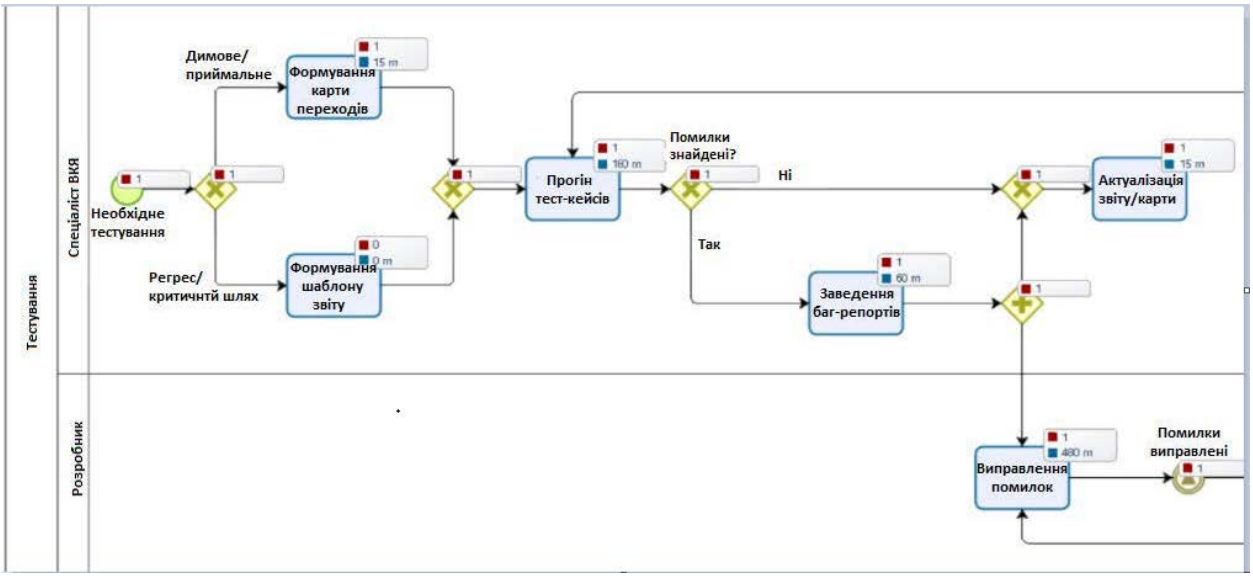

Рисунок 2.13 – Результати імітаційного моделювання етапу тестування

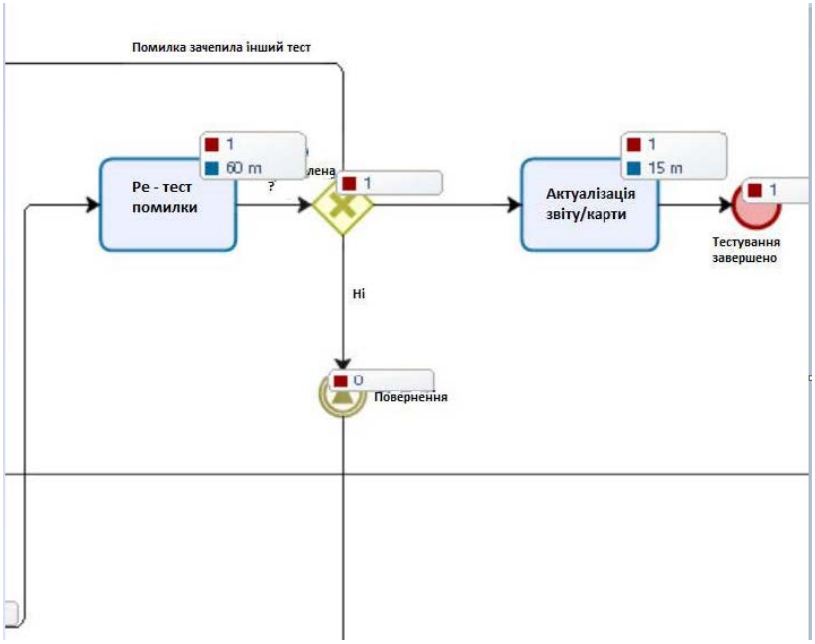

Рисунок 2.14 – Результати імітаційного моделювання етапу тестування

За результатами прогону можна назвати, що це процес підлягає оптимізації.

Результати імітаційного моделювання етапу тестування для регресійного тестування або тестування по критичному шляху представлені на рисунках 2.15 та 2.16.

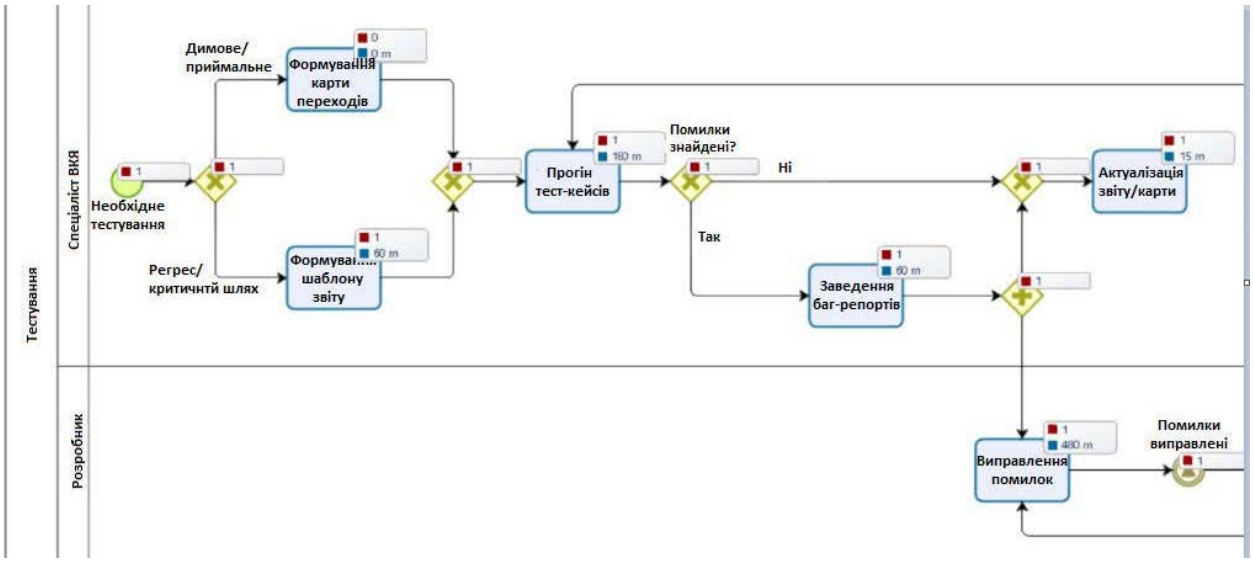

Рисунок 2.15 – Результати імітаційного моделювання етапу тестування

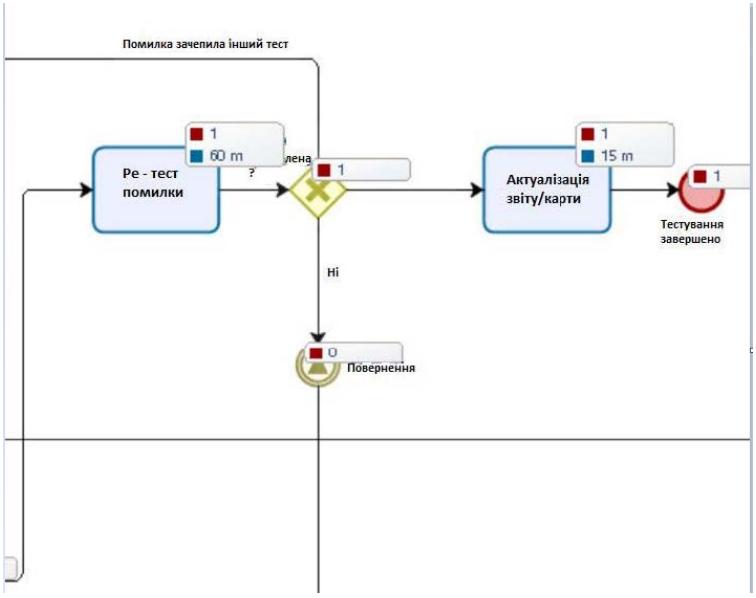

Рисунок 2.16 – Результати імітаційного моделювання етапу тестування

Для проведення регресійного тестування або тестування за критичним шляхом необхідно складати звіт про тестування, який зберігається у вигляді сторінки в Confluence і заповнюється вручну спеціалістом ОКК. Щоб підготувати шаблон для звіту, його потрібно заповнити всіма тест-кейсами, які були розроблені на етапі розробки тестів, що займає значну кількість часу — у цьому випадку 1 годину.

Також надалі такий звіт необхідно постійно актуалізувати. Завдяки впровадженню спеціалізованої системи управління тестуванням, звіти формуються автоматично, і немає потреби постійно їх оновлювати та слідкувати за їх актуальністю, адже Xray відображає актуальний стан звіту в режимі реального часу.

Дія «Заведення баг-репортів» вважається частим і важливим етапом тестування, оскільки саме на основі баг-репорту розробник знаходить та виправляє виявлену помилку. Важливо, щоб баг-репорти були повними і пояснювали суть помилки. Засоби Jira дозволяють створювати шаблони при створенні задач. Такий шаблон допоможе трохи скоротити час створення баг-репортів і гарантуватиме, що спеціаліст ВКЯ заповнить усі важливі частини баг-репорту.

Заходи з оптимізації етапу тестування:

– зміна процесу з урахуванням впровадження Xray;

– формування шаблону для баг-репортів.

Результат імітаційного моделювання етапу підготовки документації представлений на рисунку 2.17.

За результатами імітаційного моделювання етапу підготовки документації видно, що значна частина часу витрачається на складання такої документації, як ПіМВ та керівництво користувача. У Xray є функція Document Generator, яка дозволяє налаштовувати власні шаблони та створювати документи на основі даних у Jira.

ПіМВ будується на основі тест-кейсів, оскільки після впровадження Xray тест-кейси зберігаються в Jira, тому за допомогою Document Generator їх можна експортувати у документ, шаблон якого потрібно налаштувати.

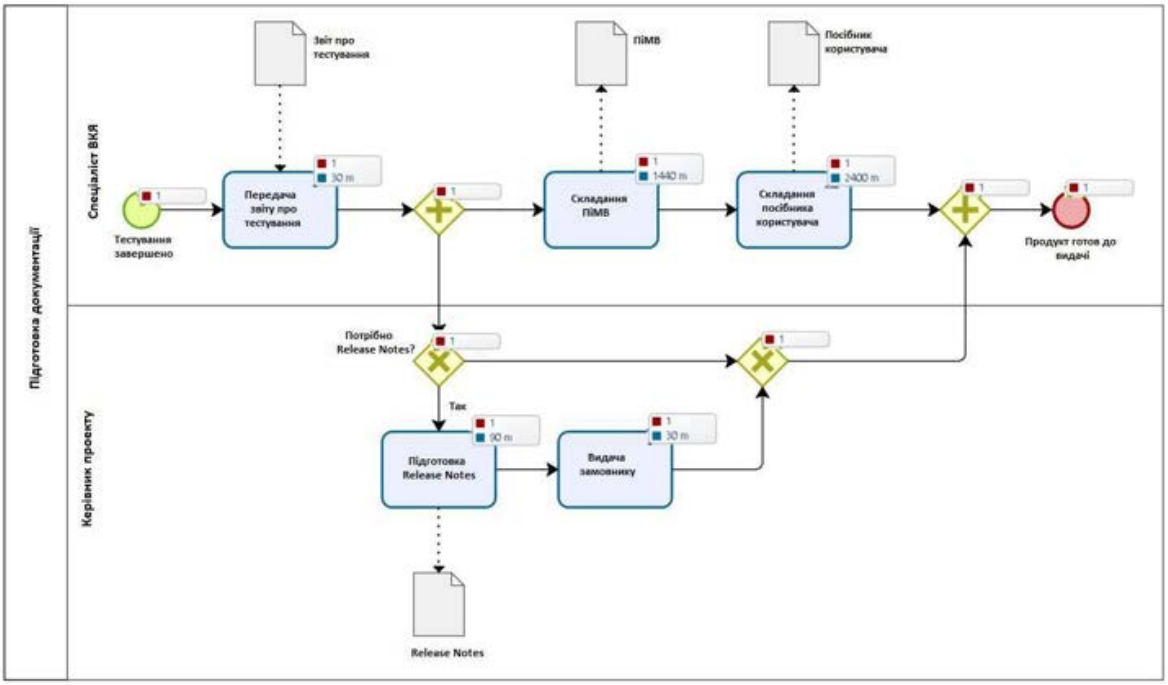

Рисунок 2.17 – Результат імітаційного моделювання етапу підготовки документації

Керівництво користувача — це документ, призначений для надання користувачам допомоги у використанні продукту. Процес створення такого документа не підлягає оптимізації.

Окрім ПіМВ, за допомогою Document Generator можна створити шаблон для експорту такого документа, як Release Notes. Release Notes формуються на основі внесених у систему змін — це можуть бути помилки або задачі, які зберігаються в Jira.

Заходи з оптимізації етапу підготовки документації:

– підготовка шаблону для Release Notes;

– підготовка шаблону для ПіМВ.

2.4 Моделювання оптимізованих бізнес-процесів тестування програмного забезпечення

За результатами проведеного аналізу імітаційного моделювання бізнес-процесів були складені заходи щодо їх оптимізації. Кожен етап роботи

спеціаліста ВКЯ на проєкті було змінено відповідно до цих заходів з метою подальшого імітаційного моделювання, яке дозволить оцінити доцільність впровадження розглянутих заходів.

Оптимізований процес підготовчого етапу роботи спеціаліста ВКЯ на проекті представлений на рисунку 2.18.

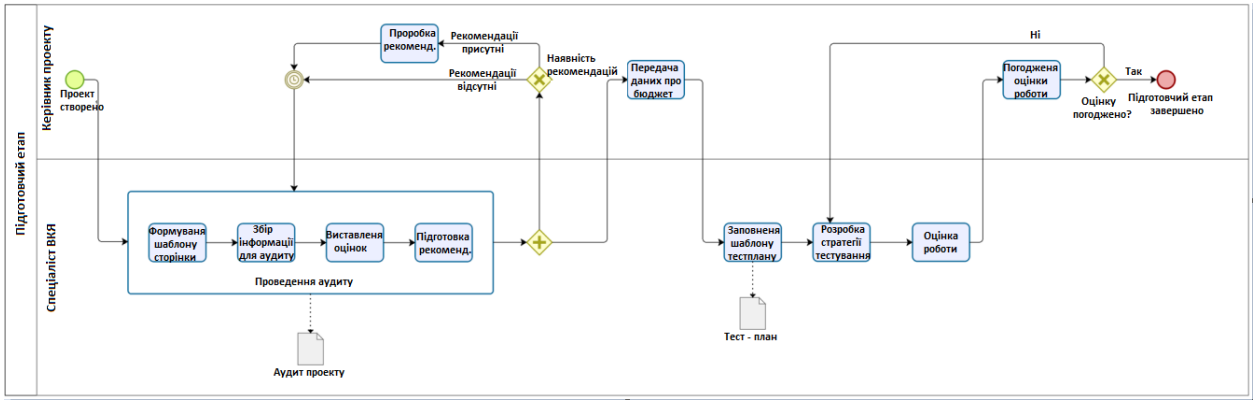

Рисунок 2.18 – Оптимізована модель підготовчого етапу

Відповідно до заходів, виявлених на етапі аналізу бізнес-процесів, були здійснені наступні зміни:

– з урахуванням автоматизації збору метрик для аудиту, скорочується час обробки дії «Збір інформації для аудиту» до 1,5 години. Оскільки частину метрик автоматизувати не вдається, подальше скорочення часу їх збору неможливе.

– додано дію «Передача даних про бюджет» у керівника проєкту, час обробки якої в середньому становить 30 хвилин.

– час обробки дії «Узгодження оцінки робіт» скоротився до 1 години завдяки додаванню дії «Передача даних про бюджет», оскільки нова дія передбачає, що спеціаліст ВКЯ буде дотримуватися встановлених рамок.

– ймовірність того, що оцінка не буде погоджена, зменшилася до 5%. У ці 5% включені всі можливі ситуації, які можуть негативно вплинути на погодження оцінки.

Оптимізований процес етапу аналізу вимог продуктового проєкту представлений на рисунку 2.19.

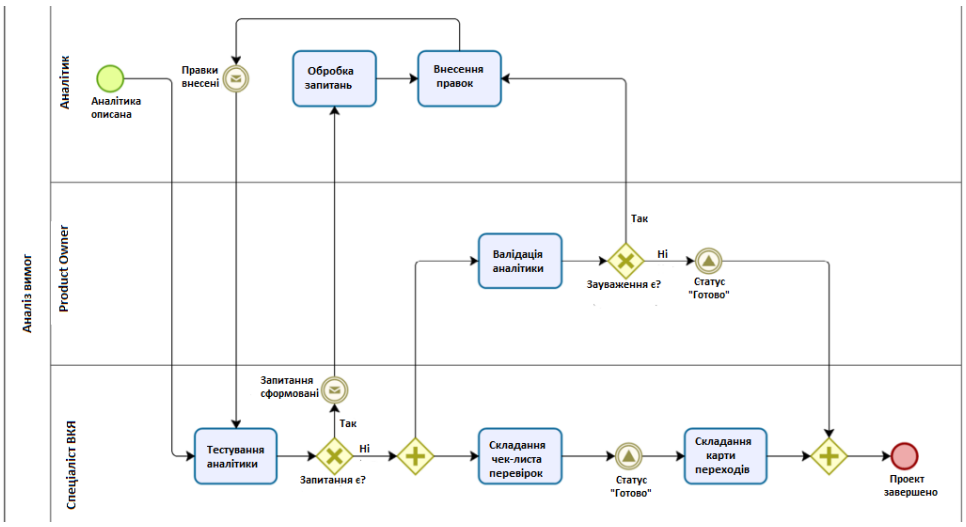

Рисунок 2.19 – Оптимізована модель етапу аналізу вимог продуктового проекту

Відповідно до заходів, виявлених на етапі аналізу бізнес-процесів, було повністю змінено процес аналізу вимог продуктового проєкту. Основні зміни пов’язані з тим, що валідація аналітики Product Owner відбувається після етапу тестування, що призвело до змін у самому процесі.

Після написання документації аналітиком спеціаліст ВКЯ приступає до тестування. В ході тестування можуть виникати питання та зауваження щодо функціональності або структури аналітичної документації. Якщо зауваження від спеціаліста ВКЯ є, аналітик опрацьовує отримані питання та вносить відповідні правки до документації, після чого аналітика повертається на тестування для перевірки внесених змін. Як тільки питання по аналітичній документації від спеціаліста ВКЯ завершуються, сторінка переходить до варіації Product Owner. Якщо від Product Owner надійшли зауваження, ймовірність чого низька, аналітика повертається на доопрацювання до аналітика і далі проходить цикл тестування. Як тільки аналітична

документація прийнята Product Owner, тобто від нього відсутні зауваження, встановлюється статус «Готово». Паралельно з валідацією спеціаліст ВКЯ може приступати до складання чек-листів та карти переходів. У разі повернення аналітики від Product Owner із зауваженнями спеціаліст ВКЯ після тестування за необхідності зможе актуалізувати вже складені чек-листи та карту переходів. Процес аналізу вимог продуктового проєкту вважається завершеним, коли аналітика пройшла валідацію Product Owner, а спеціаліст ВКЯ виконав необхідні завдання — склав чек-лист і карту переходів.

Для імітації оптимізованої моделі бізнес-процесу було внесено наступні зміни:

- Час обробки дії «Валідація аналітики» збільшився до 1 години, оскільки документація, що надходить у оптимізованому процесі, більш повна та об’ємна, на перевірку такої аналітики в середньому може йти 1 година.
- Ймовірність того, що після валідації аналітики будуть зауваження, знизилась до 15%. Ці 15% — ймовірність того, що Product Owner захоче додати нову або змінити існуючу функціональність системи.

Оптимізований процес етапу розробки тестів представлений на рисунку 2.20.

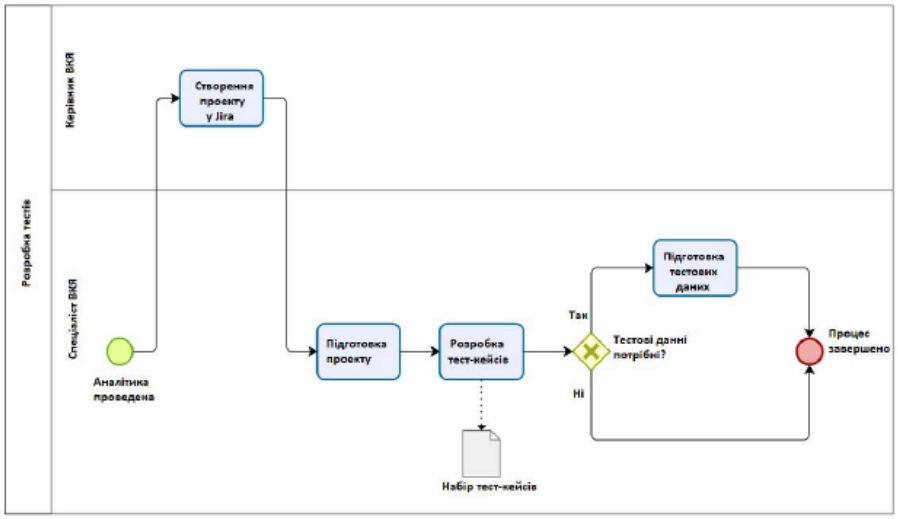

Рисунок 2.20 – Оптимізована модель етапу розробки тестів

Для оптимізації бізнес-процесу розробки тестів було прийнято рішення впровадити спеціалізований плагін Jira для управління тестуванням — Xray. Оскільки Xray є надбудовою для Jira, ведення всієї документації відбуватиметься у ній, для чого необхідно створювати нові проєкти. Для створення проєкту в Xray потрібне підключення керівника ВКЯ. Як тільки проєкт буде створено, спеціаліст ВКЯ повинен його підготувати відповідно до прийнятої структури, після чого приступає до розробки тест-кейсів. Також при необхідності спеціаліст має підготувати тестові дані.

Для імітації оптимізованої моделі бізнес-процесу були внесені наступні зміни:

– додано новий ресурс – керівник ВКЯ, витрати на годину роботи якого складають 812,5 грн;

– додано дію «Створення проєкту в Jira» у керівника ВКЯ. Час обробки дії – 15 хвилин;

– дія «Формування шаблону сторінки» замінена на дію «Підготовка проєкту»;

– час обробки дії «Підготовка проєкту» становить 15 хвилин.

Час розробки тест-кейсів та підготовки тестових даних не змінився. Оптимізований процес етапу тестування представлений на рисунку 2.21.

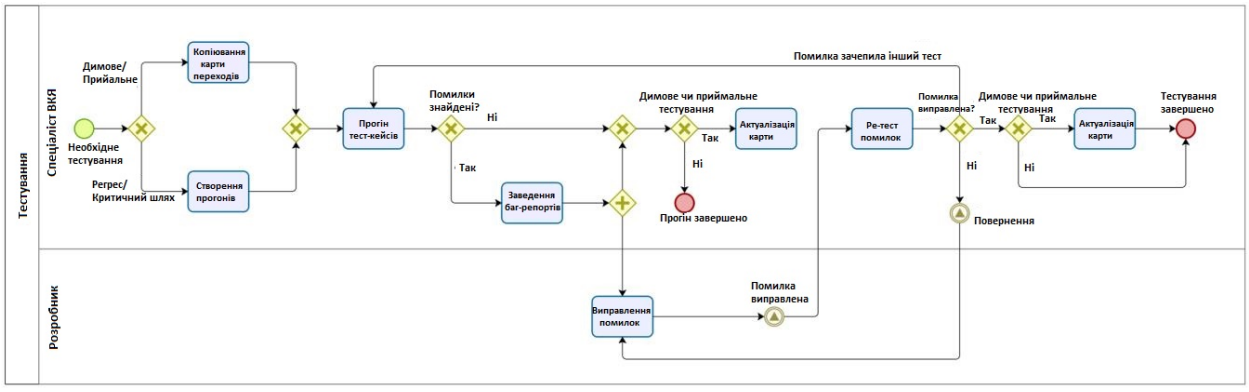

Рисунок 2.21 – Оптимізована модель етапу тестування

Впровадження спеціалізованої програми для управління тестуванням дозволило змінити процес для регресійного тестування або тестування за

критичним шляхом. Димове або приймальне тестування не залежать від використовуваної програми для управління тестуванням, тому змін у відповідній гілці процесу не відбулося.

Для імітації оптимізованої моделі бізнес-процесу були внесені наступні зміни:

– дія «Формування шаблону звіту» замінена на дію «Створення прогона». Час обробки дії «Створення прогона» становить 15 хвилин, оскільки в Xray прогін створюється досить швидко на основі розроблених тест-кейсів;

– час обробки дії «Заведення баг-репортів» скоротився до 45 хвилин за рахунок сформованого в Jira шаблону для заведення баг-репортів;

– для регресійного тестування або тестування за критичним шляхом відсутня дія актуалізації звіту після прогона та після ретесту, оскільки Xray відображає звіт про тестування автоматично в режимі реального часу.

Оптимізований процес етапу підготовки документації представлений на рисунку 2.22.

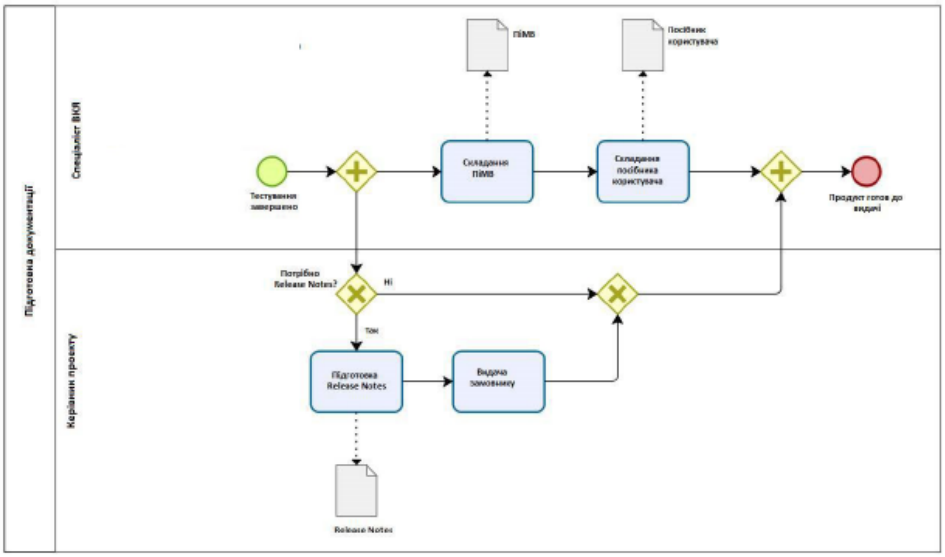

Рисунок 2.22 – Оптимізована модель етапу підготовки документації

Впровадження Xray дозволило позбутися дії передачі звіту про тестування, оскільки керівник сам може отримувати звіт у Jira про

проходження тестування, який зберігається там в актуальному стані в режимі реального часу.

Для імітації оптимізованої моделі бізнес-процесу були внесені такі зміни:

– час обробки дії «Підготовка Release Notes» скоротився до 30 хвилин завдяки шаблону, який вивантажується в один клік. Керівнику проєкту залишається лише, за потреби, відредагувати вивантажений документ;

– час обробки дії «Складання ПіМВ» скоротився до 8 годин завдяки шаблону, який також вивантажується в один клік. Спеціалісту ВКЯ залишається лише відкоригувати документ з урахуванням вимог замовника.

Оскільки документ ПіМВ є об’ємним і може сягати 40 сторінок, час обробки цієї дії все одно залишається значним.

2.5 Аналіз результатів імітаційного моделювання оптимізованих бізнес-процесів тестування програмного забезпечення

Аналогічно імітаційному моделюванню поточних бізнес-процесів були отримані результати імітаційного моделювання оптимізованих бізнес-процесів. Також на моделях за результатами імітаційного моделювання відображається легенда, де значення червоного квадрата означає кількість виконаних дій, а значення синього квадрата – загальний час обробки дії.

На рисунках 2.23 та 2.24 представлено результат імітаційного моделювання підготовчого етапу оптимізованої моделі.

В результаті імітаційного моделювання підготовчого етапу оптимізованої моделі скоротився час на проведення аудиту, а саме на збір інформації для нього.

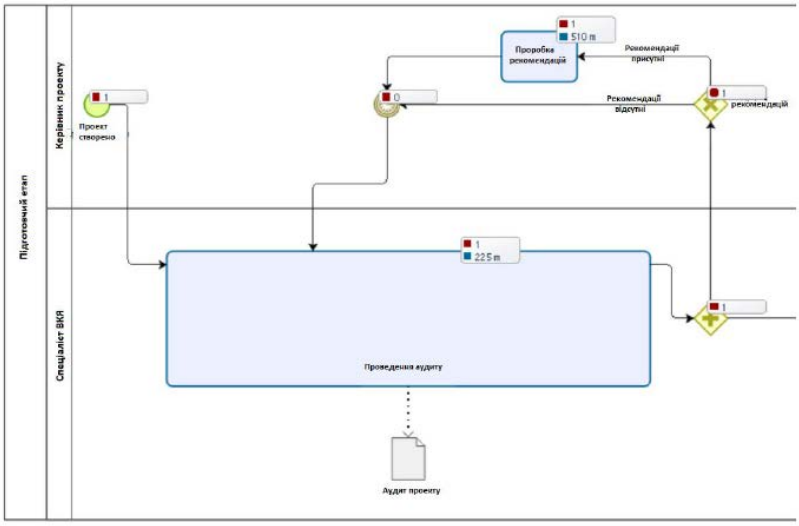

Рисунок 2.23 – Результати імітаційного моделювання підготовчого етапу оптимізованої моделі

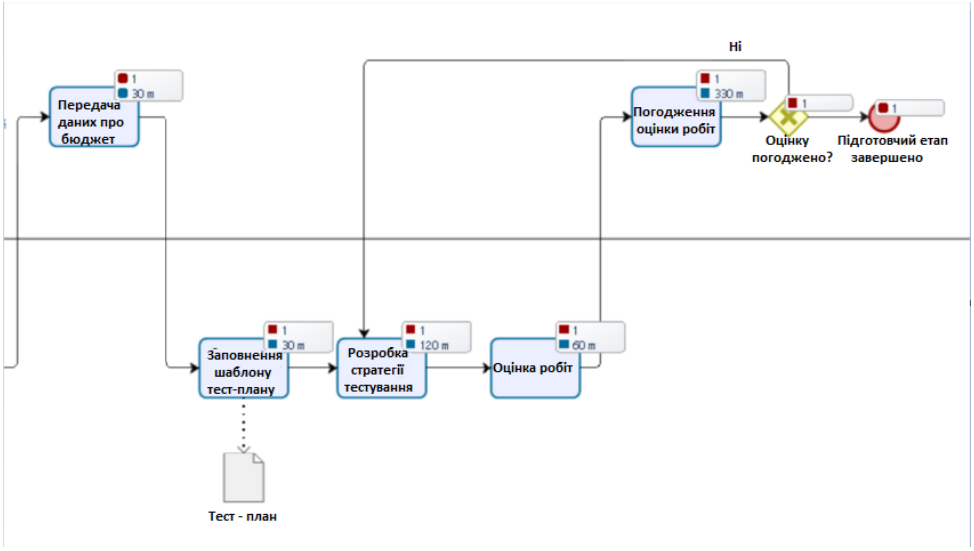

Рисунок 2.24 – Результати імітаційного моделювання підготовчого етапу оптимізованої моделі

Також завдяки додаванню процесу «Передача даних про бюджет» ймовірність виникнення циклу через незгоду з оцінкою робіт зменшилася, а отже, знизився час виконання таких дій, як «Розробка стратегії тестування», «Оцінка робіт» та «Узгодження оцінки робіт».

На рисунку 2.25 наведено результат імітаційного моделювання етапу аналізу вимог продуктового проекту оптимізованої моделі.

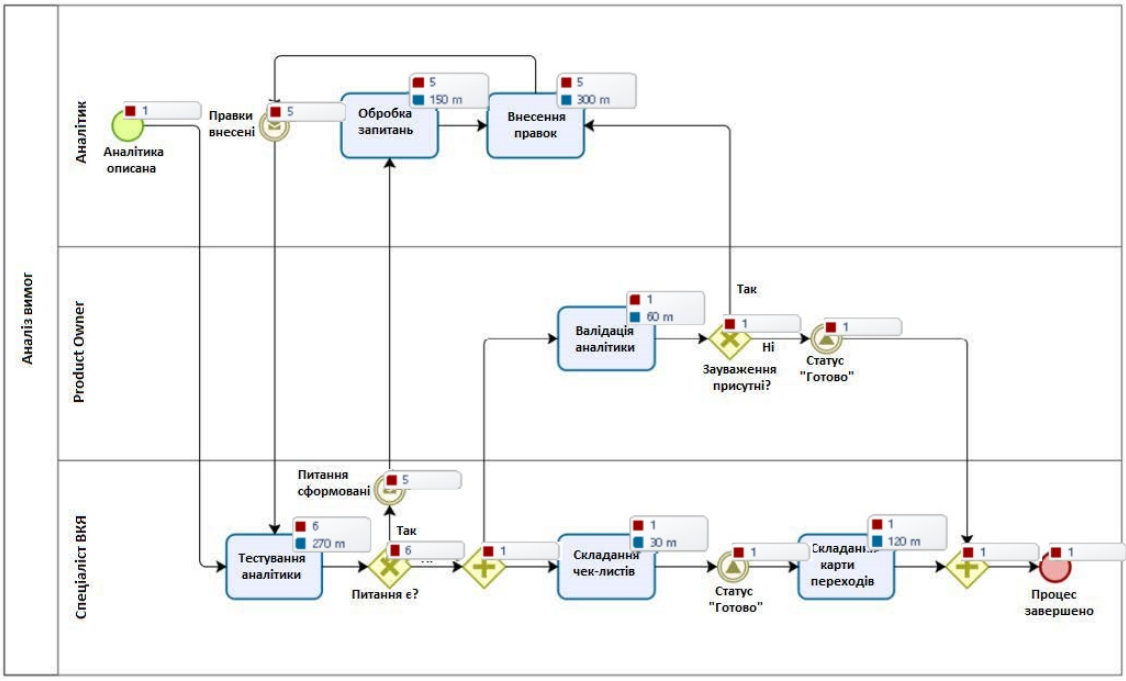

Рисунок 2.25 – Результати імітаційного моделювання етапу аналізу вимог продовольчого проекту оптимізованої моделі

В результаті імітаційного моделювання етапу аналізу вимог оптимізованої моделі скоротився час роботи Product Owner, оскільки він бере участь у валідації лише на кінцевому етапі. Також зменшилася циклічність повернення зауважень від Product Owner завдяки тому, що спеціаліст ВКЯ приступає до тестування аналітичної документації раніше і виявляє суперечності або відсутні сценарії функціонування системи на своєму етапі. Це також дозволяє Product Owner бачити результати аналітичної документації у повному обсязі. Загалом процес став коротшим за обсягом, часом та витратами ресурсів.

На рисунку 2.26 представлено результат імітаційного моделювання етапу розробки тестів оптимізованої моделі.

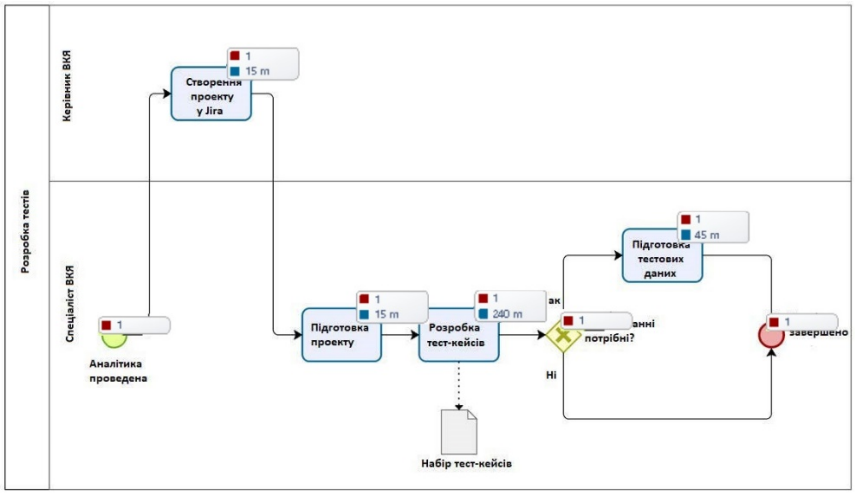

Рисунок 2.26 – Результати імітаційного моделювання етапу розробки тестів оптимізованої моделі

В результаті імітаційного моделювання етапу розробки тестів оптимізованої моделі час роботи процесу збільшився через додавання роботи керівника ВКЯ. Незважаючи на це, таке рішення дозволить заощадити час і ресурси на подальших етапах процесу.

На рисунках 2.27 і 2.29 наведено результати імітаційного моделювання етапу тестування оптимізованої моделі. Імітаційне моделювання проводилось лише для регресійного тестування або тестування за критичним шляхом, оскільки лише ця частина процесу зазнала змін.

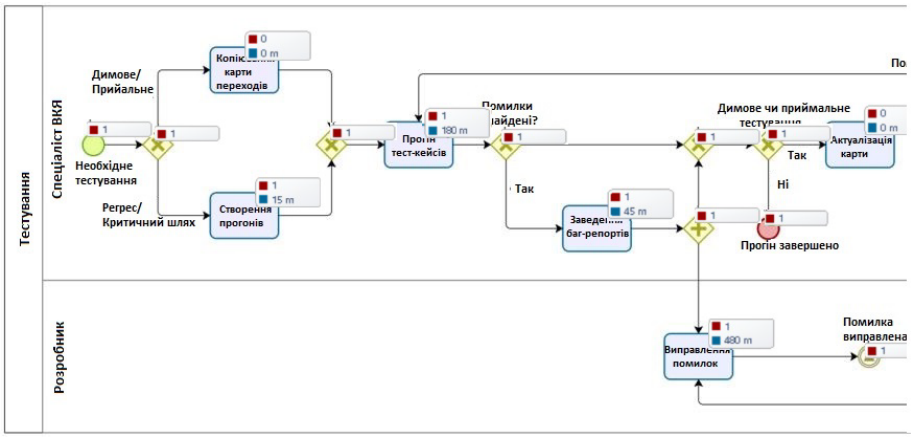

Рисунок 2.27 – Результати імітаційного моделювання етапу тестування оптимізованої моделі

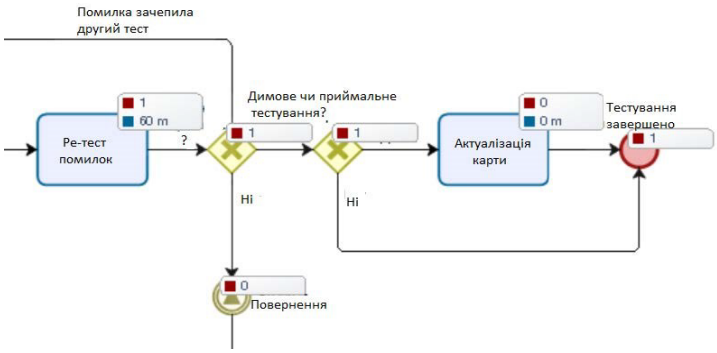

Рисунок 2.28 – Результати імітаційного моделювання етапу тестування оптимізованої моделі

В результаті імітаційного моделювання етапу тестування для регресійного тестування або тестування за критичним шляхом помітно скорочення часу на підготовку звіту про тестування, оскільки завдяки Xray цей звіт тепер формується автоматично в режимі реального часу. Також відпала потреба в його актуалізації після прогона або повторного тестування помилок, що зменшує час роботи спеціаліста ВКЯ і тривалість самого процесу загалом.

На рисунку 2.29 наведено результат імітаційного моделювання етапу підготовки документації оптимізованої моделі.

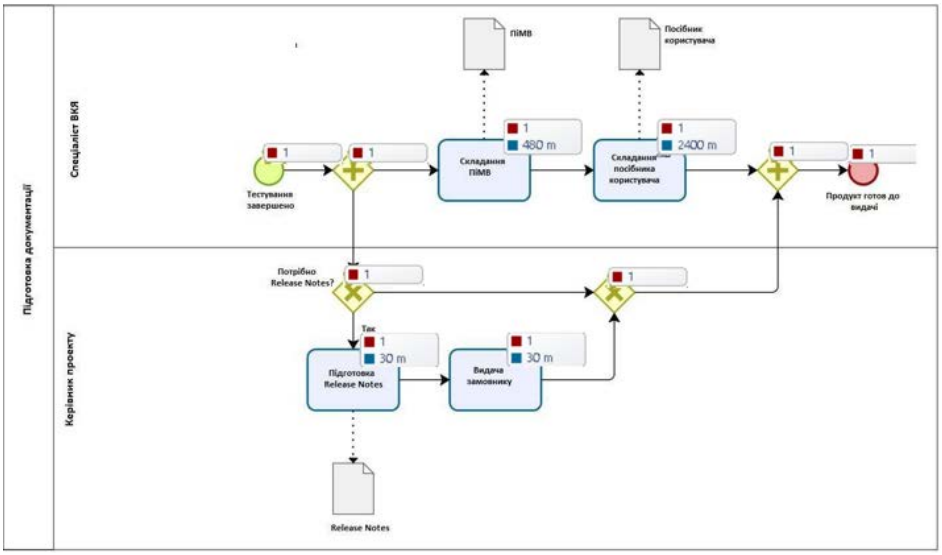

Рисунок 2.29 – Результати імітаційного моделювання етапу підготовки документації до оптимізованої моделі

В результаті імітаційного моделювання етапу підготовки документації оптимізованої моделі видно, що скоротився час участі керівника проекту у складанні Release Notes та скоротився час участі спеціаліста ВКЯ у складанні такої документації, як ПіМВ. Отже, і скоротився час виконання всього процесу.

2.6 Впровадження заходів щодо оптимізації бізнес-процесів тестування програмного забезпечення

З впровадженням Xray тест-кейси почали зберігатися безпосередньо в Jira. У Jira тест-кейси структуровані за тестовими наборами, які відповідають аналітичній документації, що забезпечує централізоване зберігання всіх даних в одному місці. Це усуває необхідність постійно створювати і підтримувати шаблони сторінок у Confluence для розробки та зберігання тест-кейсів.

Крім того, спеціалістом ВКЯ були підготовлені регламенти та інструкції з використання Xray, які розміщені у базі знань ВКЯ в Confluence. Це полегшує освоєння та стандартизацію роботи з системою серед працівників.

Раніше, коли тест-кейси зберігалися в Confluence, звіти про тестування також створювалися вручну спеціалістом ВКЯ безпосередньо в Confluence.

Впровадження Xray автоматизувало цей процес, що значно прискорило і спростило формування звітів про тестування.

Xray автоматично будує звіт про тестування після проходження прогону. По-перше, є статус-бар, який показує результати прогону під час тестування. Приклад такого статус-бару представлений на рисунку 2.30.

Рисунок 2.30 – Статус проходження прогону тестування

На рисунку 2.30 видно кілька статусів: зелений – тест пройдено, червоний – тест завершився помилкою, чорний – тест пропущено з якоїсь причини, жовтий – тест у процесі виконання, сірий – тест ще не був виконаний. Також показано загальну кількість тест-кейсів, що дає змогу наочно побачити статус проходження тестування та залишок тестів для виконання.

По-друге, існують окремі звіти про тестування, які генеруються автоматично. Це звіти по тест-плану, звіти по прогонах, звіти про проходження кожного тест-кейсу та звіти, пов’язані з вимогами.

Наступним заходом для оптимізації процесу стало створення шаблону для баг-репортів. Шаблон баг-репорту можна створити за допомогою сценарію автоматизації в Jira. Сценарій автоматизації містить: правило виконання дії, умову спрацювання дії та саму дію. Так, для створення шаблону помилки правило виконання дії – це коли створюється завдання, умова виконання – якщо завдання є багом (помилкою), а дія – це зміна опису помилки, тобто додавання обов’язкових полів для заповнення. Сценарій автоматизації наведено на рисунку 2.31, а результат – на рисунку 2.32.

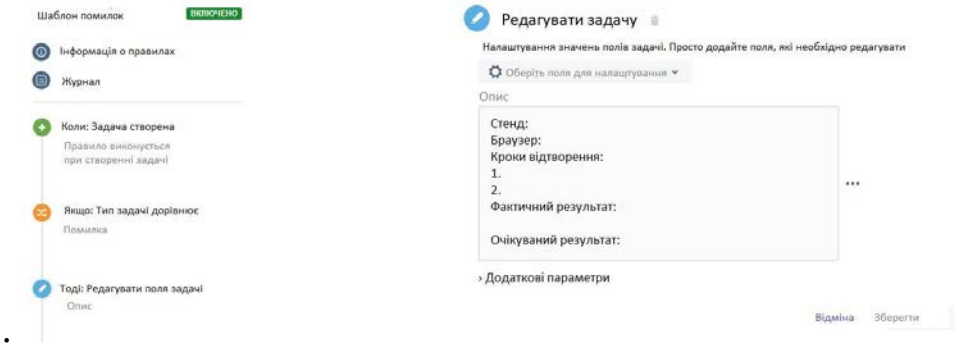

Рисунок 2.31 – Сценарій автоматизації для шаблону баг-репорту

Останні два заходи пов’язані зі створенням шаблону для експорту документів із Jira. Вивантаження документів із Jira здійснюється за допомогою Document Generator, куди необхідно завантажити відповідний шаблон. Перед тим, як завантажити шаблон, його потрібно розробити. Розробка шаблону виконується за допомогою скрипта.

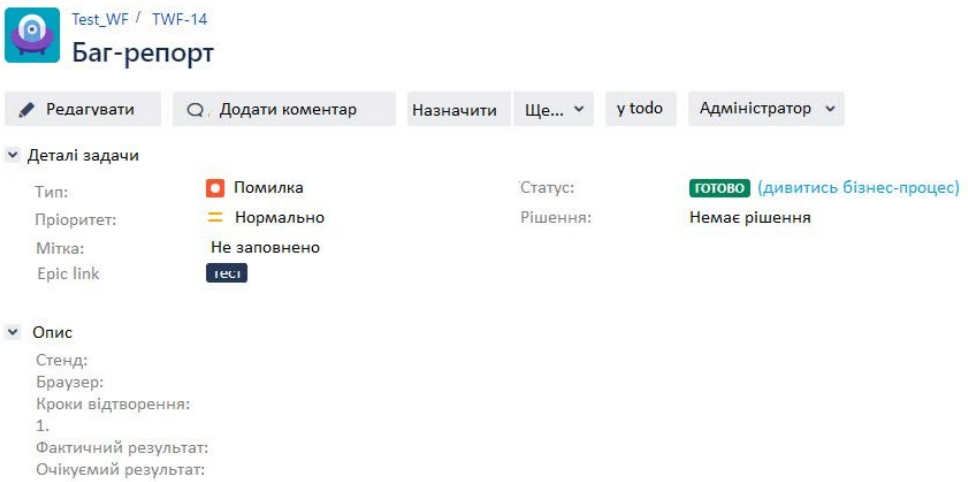

Рисунок 2.32 – Результат автоматизації створення шаблону баг-репорту

Варто зазначити, що для ПіМВ було розроблено три варіанти шаблону: без передумов, з передумовами для розвантаження за конкретною функціональністю та з передумовами для розвантаження по всьому проекту.

Висновки до розділу:

У розділі проведено всебічний аналіз та моделювання бізнес-процесів тестування програмного забезпечення. Було детально змодельовано ключові етапи тестування, а саме: підготовчий етап, аналіз вимог, розробку тестів, безпосереднє тестування та підготовку документації. Це дало змогу виявити вузькі місця та визначити шляхи підвищення ефективності кожного етапу.

Для моделювання бізнес-процесів використано систему Bizagi, яка підтримує нотацію BPMN. Цей інструмент дозволив візуалізувати бізнес-процеси, а також підготувати їх для імітаційного моделювання та аналізу результатів.

У підрозділах розглянуто налаштування бізнес-процесів для імітаційного моделювання, проведено аналіз отриманих результатів, а також змодельовано оптимізовані бізнес-процеси, які враховують виявлені недоліки.

Отримані результати імітаційного моделювання оптимізованих процесів підтвердили доцільність впровадження заходів щодо їх вдосконалення.

Таким чином, результати цього розділу створюють підґрунтя для впровадження практичних заходів з оптимізації бізнес-процесів тестування програмного забезпечення, що сприятиме підвищенню якості продукту та ефективності всієї діяльності в цій галузі.

ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи було виявлено проблему, пов’язану з процесом життєвого циклу тестування. Основна суть проблеми полягає в тому, що більшості компаній необхідно скоротити час випуску реалізованої продукції, при цьому не втрачаючи якість готового продукту та підвищуючи лояльність клієнтів і замовників. Дослідження цієї проблеми було проведено на прикладі бізнес-процесів життєвого циклу тестування у відділі тестування або відділі контролю якості компанії.

Дослідження бізнес-процесів відділу тестування компанії здійснювалося шляхом вивчення регламентів, які безпосередньо складають співробітники відділу, а також моніторингу роботи працівників. Моніторинг проводився за допомогою інтерв’ю, у якому взяли участь 6 працівників відділу: керівник відділу, два старших фахівці та три спеціалісти. Список питань для інтерв’ю було структуровано за блоками, які корелюють із процесом життєвого циклу тестування програмного забезпечення. Після дослідження робочих процесів відділу було проведено моделювання бізнес-процесів, що дозволяє візуалізувати їх для подальшого аналізу.

Аналіз змодельованих бізнес-процесів здійснювався методом імітаційного моделювання, яке замінює досліджувану систему моделлю з метою вивчення її поведінки. Для моделювання були налаштовані такі параметри, як ресурси та вартість роботи ресурсу за годину. Проведене імітаційне моделювання дозволило виявити проблемні місця у розглянутих бізнес-процесах. На основі цього було розроблено план заходів з оптимізації роботи відділу тестування, спрямований на скорочення часу та витрат без втрати якості, при збереженні всіх функцій, що виконуються відділом.

Основними заходами оптимізації стало впровадження системи управління тестуванням Xray, що дозволяє централізовано зберігати та формувати тестову документацію, а також проводити тестування програмних продуктів. Крім того, впровадження Xray автоматизувало роботу зі складання

документації, зокрема звітів про тестування, ПіМВ та Release Notes, що дало змогу значно скоротити час їх підготовки. Також було автоматизовано процеси збору інформації для проведення аудиту та створено шаблони для заведення баг-репортів із використанням програмних продуктів компанії, таких як Confluence та Jira. Враховуючи перелічені заходи, були переглянуті та оптимізовані окремі бізнес-процеси роботи відділу тестування.

Для оцінки ефективності впровадження розроблених заходів було розраховано вартість роботи відділу до їх впровадження, а також вартість самих заходів. Після реалізації заходів визначено вартість оптимізованих бізнес-процесів відділу тестування. Враховуючи, що заходи виконуються одноразово, а компанія має декілька проектів, економія у грошовому виразі стане очевидною вже на третьому проекті. Ці результати можна вважати успішними, оскільки компанія постійно працює над більш ніж 8 власними проектами, що гарантує безперервне виконання оптимізованих бізнес-процесів.

Розроблені заходи та проведене дослідження в рамках випускної кваліфікаційної роботи можуть бути використані не лише у розглянутій компанії, а й у сторонніх організаціях.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1 Roger Pressman, Bruce Maxim. Software Engineering: A Practitioner's Approach, 9th Edition. Published by McGraw-Hill Education, 2022. 977 pp.

2 Nina S. Godbole. Software Quality Assurance: Principles and Practices for the new Paradigm. Published by Alpha Science International, 2017. 1066 pp.

3 Boris Beizer. Software Testing Techniques. Published by Van Nostrand Reinhold, 2007, 550 pp.

4 Молодцова О.П. Управління якістю програмної продукції: навчальний посібник. К : КНЕУ, 2001. 248 с.

5 Cem Kaner et al. Testing Computer Software, 2nd ed. International Thompson Computer Press, 1999. 496 p

6 Dorothy Graham, Rex Black, Erik van Veenendaal. Foundations of Software Testing ISTQB Certification. Engage Learning, 2019. 243 p.

7 Horch, John W. Practical guide to software quality management. Artech House, 1996. 259 p.

8 Hackr.io: URL: https://hackr.io/blog/what-is-software- testing-life-cycle (Дата звернення: 23.04.2025).

9 Intellect.icu URL: https://intellect.icu/tipy-i-vidy-testirovaniya-urovni-testirovaniya-metody- testirovaniya-5187 (Дата звернення: 23.04.2025).

10 MadDevs: URL: https://maddevs.io/blog/test-documentation-in-software-testing/ (Дата звернення: 23.04.2025)

11 QA Lead:URL: https://theqalead.com/tools/test-management-tools-for-jira/ (Дата звернення: 23.04.2025).

12 Дідковська М.В., Тимошенко Ю.О. Тестування: Основні визначення, аксіоми та принципи. Текст лекцій. Частина І. НТУУ «КПІ». Кафедра математичних методів системного аналізу, 2010. 62 с.

13 Дідковська М.В., Тимошенко Ю.О. Тестування: Критерії та методи. Текст лекцій. Частина ІІ. НТУУ «КПІ». Кафедра математичних методів системного аналізу, 2010 90 с.

14 Молодцова О.П. Управління якістю програмної продукції: навчальний посібник. К : КНЕУ, 2001. 248 с.

15 Коробєйник А.Н. Основи тестування ПЗ. К : 2012. 126 с.

16 Types of Testing Environments. url: https://www.testenvironmentmanagement.com/types-of-testing-environments (дата звернення: 15.04.2025)

17 Visual Paradigm Online. url: https://online.visual-paradigm.com (дата звернення: 15.04.2025)

18 Моркун Н. В., Завсегдашня І. В. Методичні вказівки до виконання кваліфікаційної роботи бакалавру для студентів спеціальності 122 “Комп’ютерні науки”. Кривий Ріг : Видавничий центр КНУ, 2021. 50 с.

19 ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

20 ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

21 ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація).

ДОДАТОК А

Питання щодо інтерв'ю у фахівців відділу

Питання	Ціль питання
Яку посаду ви обіймаєте?	Збір загальної інформації
Скільки часу ви працюєте у цьому відділі?	
Як часто ви проводите аудит на своєму проекті?	Збір загальної інформації підготовчого етапу робіт
Які переваги та недоліки ви бачите у процесі проведення аудиту?	
Чи вважаєте ви за можливе оптимізацію та автоматизацію цього процесу?	
Після проведення аудиту чи відстежуєте ви усунення зауважень, чи це буде помітно лише після проведення наступного аудиту?	
Чи вважаєте ви за необхідне відслідковувати усунення зауважень?	
Висловіть свою думку про цей процес. Чи хотілося б щось покращити/змінити крім вищесказаного?	
Скільки в середньому часу у вас займає складання тест-плану (з нуля)?	
Чи виникають у вас складнощі з оцінкою робіт по тестуванню?	
Яка мета збереження і створення карти переходів на проекті?	Збір інформації про проведення етапу аналізу вимог
Скільки часу в середньому у вас займає складання карти переходів проекту?	
Розкажіть про ваш досвід використання карти переходів. Як часто доводиться звертатися до карти переходів?	
Як довго ви очікуєте уточнення вимог від аналітика (відповіді на ваші питання)?	
Розкажіть, як відбувається ваше взаємодія з аналітиком?	
Скільки в середньому ітерацій питань-уточнень у вас припадає на одну фіче-сторінку?	
Якщо більше однієї ітерації, скажіть, чому, на вашу думку, не вдається вирішити всі питання за одну ітерацію?	
Чи вважаєте ви необхідним етап написання чек-листа при тестуванні аналітики? Якщо так, чому?	

Складений чек-лист допомагає вам у майбутніх процесах? Ви ним користуєтесь? Якщо так, опишіть у яких випадках і за яких обставин.	
Чи бачите ви можливість підключення до тестування фіче-сторінок раніше? Якщо так, яким чином?	
Висловіть свою думку про існуючий процес. Чи хотілося б щось покращити/змінити, окрім вищезазначеного?	
Скільки в середньому часу у вас займає написання одного тест-кейса?	Збір інформації про проведення етапу розробки тестів
Чи складно вам підтримувати тест-кейси в актуальному стані? Якщо так, чому і що заважає?	
Скільки в середньому часу у вас займає актуалізація тест-кейсів? Як часто це доводиться робити на вашому проекті?	
Як ви відслідковуєте актуальність написаних тест-кейсів?	
Чи зручний вам поточний формат написання тест-кейсів? Розкажіть про його переваги та недоліки.	
Як ви готуєте тестові дані для тестування? Де вони зберігаються?	
Як відбувається ваша підготовка до тестування (виділення тест-кейсів, складання шаблону звіту)?	Збір інформації про проведення етапу тестування і документування
Скільки в середньому часу у вас займає підготовка до тестування?	
Які види тестування ви зазвичай проводите?	
Як часто відбувається тестування на вашому проекті (скільки ітерацій, наприклад, регрес і приймальне)?	
Скільки в середньому часу у вас займає заведення одного баг-репорта?	
Баг-репорти у вас заводяться за єдиним шаблоном? Чи це залежить від спеціаліста?	
Чи зручний вам поточний формат звіту про тестування? Розкажіть про його переваги та недоліки.	
Скільки в середньому часу у вас займає складання ПМІ?	

ДОДАТОК Б

Налаштування імітаційної моделі

Таблиця Б.1 – Для підготовчого етапу

Елемент процесу	Параметри налаштування
Формування шаблону сторінки аудиту	Час обробки – 30 хвилин Ресурс – спеціаліст ВКЯ.
Збір інформації для аудиту	Час обробки – 3 годин. Ресурс – спеціаліст ВКЯ..
Виставлення оцінок	Час обробки – 45 хвилин. Ресурс – спеціаліст ВКЯ.
Підготовка рекомендацій	Час обробки – 1 година. Ресурс – спеціаліст ВКЯ.
Шлюз «Наявність рекомендацій»	Ймовірність «Рекомендації відсутні» – 10%. Ймовірність «Рекомендації присутні» – 90%
Опрацювання рекомендацій	Час обробки – 8 годин. Ресурс – керівник проєкту.
Подія таймера	Інтервал – 30 днів.
Заповнення шаблону тест-плану	Час обробки – 30 хвилин. Ресурс – спеціаліст ВКЯ.
Розробка стратегії тестування	Час обробки – 2 години. Ресурс – спеціаліст ВКЯ
Оцінка робіт	Час обробки – 1 година. Ресурс – спеціаліст ВКЯ.
Узгодження оцінки робіт	Час обробки – 2 години. Ресурс – керівник проєкту.
Шлюз «Оцінку узгоджено?»	Ймовірність «Так» – 75%. Ймовірність «Ні» – 25%.

Таблиця В.2 - Для етапу аналізу вимог комерційного проекту

Елемент процесу	Параметри налаштування
Тестування аналітики	Час обробки – 45 хвилин. Ресурс – спеціаліст ВКЯ.
Шлюз «Питання є?»	Ймовірність «Так» – 80%. Ймовірність «Ні» – 20%.
Обробка питань	Час обробки – 30 хвилин. Ресурс – аналітик.
Внесення правок	Час обробки – 1 години. Ресурс – аналітик.
Складання чек-листа перевірок	Час обробки – 30 хвилин. Ресурс – спеціаліст ВКЯ.
Складання переходів картки	Час обробки – 2 години. Ресурс – спеціаліст ВКЯ.

Таблиця В.3 – Для етапу аналізу вимог продуктового проекту

Елемент процесу	Параметри налаштування
Валідація аналітики	Час обробки – 30 хвилин. Ресурс – product owner.
Шлюз «Зауваження присутні?»	Ймовірність «Так» – 75%. Ймовірність «Ні» – 25%.
Внесення правок	Час обробки – 1 година. Ресурс – аналітик.
Тестування аналітики	Час обробки – 45 хвилин. Ресурс – спеціаліст ВКЯ.
Шлюз «Питання є?»	Ймовірність «Так» – 80%. Ймовірність «Ні» – 20%.
Опрацювання питань	Час обробки – 30 хвилин. Ресурс – аналітик.
Шлюз «Потрібне погодження?»	Ймовірність «Так» – 75%. Ймовірність «Ні» – 25%.
Погодження доопрацювань	Час обробки – 30 хвилин. Ресурс – product owner.
Внесення правок	Час обробки – 1 година. Ресурс – аналітик.
Складання перевірок чек-листа	Час обробки – 30 хвилин. Ресурс – спеціаліст ВКЯ.
Складання переходів карти	Час обробки – 2 години. Ресурс – спеціаліст ВКЯ.

Таблиця В.4 - Для етапу розробки тест-кейсів

Елемент процесу	Параметри налаштування
Формування шаблону сторінки	Час обробки – 15 хвилин. Ресурс – спеціаліст ВКЯ.
Складання тест-кейсів	Час обробки – 3 години. Ресурс – спеціаліст ВКЯ.
Шлюз «Потрібні тестові дані?»	Ймовірність «Так» – 75%. Ймовірність «Ні» – 25%.
Підготовка тестових даних	Час обробки – 45 хвилин. Ресурс – спеціаліст ВКЯ.

Таблиця 2.5 - Для етапу тестування

Елемент процесу	Параметри налаштування
Копіювання карти переходів	Час обробки – 15 хвилин. Ресурс – спеціаліст ВКЯ.
Формування шаблону звіту	Час обробки – 1 година. Ресурс – спеціаліст ВКЯ.
Прогін тест-кейсів	Час обробки – 3 години. Ресурс – спеціаліст ВКЯ.
Шлюз «Помилки знайдені?»	Ймовірність «Так» – 75%. Ймовірність «Ні» – 25%.
Заведення баг-репортів	Час обробки – 1 година. Ресурс – спеціаліст ВКЯ.
Актуалізація звіту/карти	Час обробки – 15 хвилин. Ресурс – спеціаліст ВКЯ.
Виправлення помилок	Час обробки – 8 годин. Ресурс – розробник.
Ре-тест помилок	Час обробки – 1 година. Ресурс – спеціаліст ВКЯ.
Шлюз «Помилку виправлено?»	Ймовірність «Так» – 40%. Ймовірність «Ні» – 40%. Ймовірність «Помилка зачепила інший тест» – 20%.
Актуалізація звіту/карти	Час обробки – 15 хвилин. Ресурс – спеціаліст ВКЯ.

Таблиця В.6 – Для етапу підготовки документації

Елемент процесу	Параметри налаштування
Передача звіту про тестування	Час обробки – 30 хвилин. Ресурс – спеціаліст ВКЯ.
Шлюз «Потрібен Release Notes?»	Ймовірність «Так» – 80%. Ймовірність «Ні» – 20%.
Підготовка Release Notes	Час обробки – 1 година 30 хвилин. Ресурс – керівник проєкту.
Видача замовнику	Час обробки – 30 хвилин. Ресурс – керівник проєкту.
Складання ПіМВ	Час обробки – 24 години. Ресурс – спеціаліст ВКЯ.
Складання керівництва користувача	Час обробки – 40 годин. Ресурс – спеціаліст ВКЯ.