Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: **Методи та засоби автоматизації тестування програмного забезпечення комп'ютерних систем з використанням програмних роботів**

Виконав: студент(ка) 6 курсу, групи СІм-62
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

(підпис) Турчиняк Р. В. (прізвище та ініціали)

Керівник (підпис) Стадник Н.Б. (прізвище та ініціали)

Нормоконтроль (підпис) Тиш Є.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
«____» грудня 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Турчиняку Роману Валерійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Методи та засоби автоматизації тестування програмного забезпечення комп'ютерних систем з використанням програмних роботів*

Керівник роботи *Стадник Наталія Богданівна, кандидат технічних наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*01*» *грудня 2023* року № *4/7-1132*

2. Термін подання студентом завершеної роботи *29.12.2023 р.*

3. Вихідні дані до роботи *Технологія RPA, життєвий цикл тестування ПЗ, UiPath Test Suite, Selenium*

4. Зміст роботи (перелік питань, які потрібно розробити)

Вступ
1 Теоретичні основи тестування програмних продуктів
2 Методи і засоби автоматизації тестування
3 Апробація методів тестування з використанням rpa
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

1. Актуальність і мета дослідження.
2. Задачі дослідження, об'єкт і предмет, наукова новизна і практична цінність дослідження.
3. Порівняння ручного і автоматизованого тестування
4. Формалізація етапів роботи автоматизованої системи тестування
5. Схема алгоритму для тестування модуля авторизації і результат тестування
6. Структура модулів UiPath Test Suite
7. Експериментальне дослідження
8. Результати дослідження ефективності методів тестування
9. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Осухівська Г. М., зав. кафедри КС*		
Безпека в надзвичайних ситуаціях	*Стадник І. Я., професор кафедри ОХ*		

7. Дата видачі завдання *20.11.2023*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз сучасних методів і технологій автоматизованого тестування програмного забезпечення*	*25.11.2023*	*виконано*
2.	*Методи і сценарії автоматизованого тестування програмних продуктів на з використанням програмних роботів*	*29.11.2023*	*виконано*
3.	*Апробація методів автоматизованого тестування програмних продуктів з використанням програмних роботів*	*10.12.2023*	*виконано*
4.	*Аналіз ефективності використання програмних роботів в порівнянні з іншими методами тестування*	*16.12.2023*	*виконано*
5.	*Охорона праці та безпека в надзвичайних ситуаціях*	*18.12.2023*	*виконано*
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*19.12.2023*	*виконано*
7.	*Попередній захист кваліфікаційної роботи магістра*	*20.12.2023*	*виконано*
8.	*Захист кваліфікаційної роботи магістра*	*29.12.2023*	*виконано*

Студент _____ *Турчиняк Роман Валерійович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Стадник Наталія Богданівна*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи та засоби автоматизації тестування програмного забезпечення комп’ютерних систем з використанням програмних роботів // Кваліфікаційна робота магістра // Турчиняк Роман Валерійович // ТНТУ, комп’ютерна інженерія, група СІм-62 // Тернопіль, 2023 // с. – 72, рис. – 47, табл. – 8, бібліогр. – 29.

Ключові слова: тестування, алгоритм, автоматизація, програмний робот.

У кваліфікаційній роботі магістра досліджено методи і засоби тестування ІТ-продукту, з використанням програмних роботів. Для розглянутих методів визначено основні концепції та етапи проведення тестування.

Проведено огляд життєвого циклу тестування програмних продуктів. Визначені основні етапи тестування і місце, де використання програмних роботів буде максимально ефективним.

Розглянуті методи і засоби автоматизації тестування ПЗ. Формалізовані основні етапи автоматизованого тестування. Визначені необхідні інструменти для виконання подальших експериментальних досліджень. Також була спроектована модель роботи автоматизованої системи тестування. Наведено обґрунтування вибору програмних засобів..

Складено сценарій проведення автоматизованого тестування з використанням різних методів і програмних засобів. В рамках апробації зроблені висновки щодо ефективності тестування програмного забезпечення з використанням програмних роботів.

ABSTRACT

Methods and means of computer information systems comprehensive testing // Master graduation thesis // Turchyniak Roman Valeriiovych // TNTU, computer engineering, group CIm-61 // Ternopil, 2023 // p. – 72, fig. – 47, tab. – 8, bibliography. - 29.

Keywords: testing, algorithm, automation, software robot.

In the master's qualification work, the methods and means of IT product testing, using software robots, were investigated. The main concepts and stages of testing are defined for the considered methods.

An overview of the life cycle of testing software products was conducted. The main stages of testing and the place where the use of software robots will be most effective are defined.

The considered methods and means of software testing automation. The main stages of automated testing are formalized. Necessary tools for further experimental research are determined. A working model of the automated testing system was also designed. The justification for the choice of software tools is given.

A script for conducting automated testing using various methods and software tools has been compiled. As part of the approbation, conclusions were made regarding the effectiveness of software testing using software robots.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

RPA англ. Robotic process automation роботизована автоматизація процесів.

QA англ. Quality Assurance забезпечення якості.

STLC англ. Software Testing Life Cycle життєвий цикл тестування програмного забезпечення.

ІТ інформаційні технології.

ОС операційна система.

ПЗ програмне забезпечення.

ВСТУП

Актуальність теми. Програмні системи стали невід'ємною частиною нашої повсякденного життя. Вони допомагають нам задовольнити багато наших потреб і навіть вирішити деякі проблеми, з якими ми стикаємося. Ми використовуємо домашні банківські програми, замовляємо їжу, плануємо подорожі та робимо покупки. У нашому житті не так багато областей, які так чи інакше не торкнулися програмним забезпеченням.

Ця еволюція відбулася у всіх галузях. Програмне забезпечення стало технологією номер один у будь-якому бізнесі. Таким чином, програмний збій стає бізнес-невдачею. Саме тому тестування програмного забезпечення має першорядне значення. Це знижує ризик відмови, допомагаючи виявляти дефекти доти, як товар потрапить до користувача. І зрештою, це допомагає створювати якісні продукти.

На даний момент жоден цикл розробки програмного забезпечення не обходиться без етапу тестування, так як тестування націлено на забезпечення якості системи, що розробляється. Під якістю програмного забезпечення розуміють сукупність характеристик, які дозволяють визначити, задовольняє чи програма встановленим вимогам і виконує чи вона те, що від її очікують. Основний завданням тестування програмного забезпечення є зниження вартості розробки шляхом раннього виявлення дефектів.

Раннє виявлення дефектів та своєчасний зворотний зв'язок дозволяють створювати високоякісне та надійне програмне забезпечення Крім того, організація ефективного і безперервного тестування прискорює постачання цього програмного забезпечення і знижує витрати компанії по причині того, що у розробників з'являється можливість вносити в код зміни з мінімальними ризиками, які здатні порушити працездатність програми.

Головне рішення для безперервного тестування є його автоматизація. Автоматизація тестування пройшла довгий шлях. Від самого початку вона впроваджувалась лише для скорочення часу тестування, зараз до цього додається забезпечення оптимального тестового покриття і більш ефективне використання

тест-кейсів. Таким чином, використання автоматизації тестування має на меті отримання високоякісного програмного продукту, і як можна швидше.

Однак не завжди рішення автоматизувати тестування допомагає вирішити вищезазначені завдання, та й саме впровадження може призвести до безлічі проблем. Рішення про використання конкретного типу тестування залежить більшою мірою від завдання, яке вирішується на проекті. Так у даній роботі необхідно визначити області застосування і види тестування, для яких ефективно автоматизоване тестування з використанням програмних роботів, а також розглянути деякі інструменти автоматизації.

Метою кваліфікаційної роботи є підвищення ефективності тестування програмного забезпечення за рахунок розробки автоматизованої системи тестування з використанням програмних роботів, а також коригування загального плану процесу розробки з урахуванням створення даної системи.

Для того, щоб досягти мети, необхідно вирішити наступні задачі:

– провести дослідження літературних джерел сфері автоматизованого тестування;

– провести дослідження основних фреймворків для написання автотестів та здійснити вибір оптимального;

– здійснити моделювання системи автоматизованого тестування;

– реалізувати систему тестування з використанням програмних роботів;

– провести оцінку ефективності цієї розробки в порівнянні з іншими методами.

Об'єкт дослідження: процес автоматичного тестування програмних продуктів з використанням програмних роботів.

Предмет дослідження: сценарій тестування з використанням програмних роботів для підвищення продуктивності та надійності розробки програмних продуктів.

Методи дослідження: методи системного аналізу та дослідження операцій; методи автоматизованого тестування, метод лексико-синтаксичних шаблонів; статистичні методи аналізу текстів природною мовою.

Наукова новизна полягає у розробці засобу тестування з використанням програмних роботів, що дозволить підвищити ефективність розробки ПЗ за рахунок раннього виявлення помилок в розроблюваних програмних продуктах.

Практичне значення результатів кваліфікаційної роботи полягає у розробці автоматизованої системи тестування на базі технології RPA.

Публікації. Результати дослідження апробовано на XI науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології.

1. Турчиняк Р., Стадник Н. Використання RPA технології для тестування програмних продуктів. Матеріали XI науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (13-14 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 249

2. Турчиняк Р., Стадник Н. Використання Uipath test suite для розробки RPA-роботів. Матеріали XI науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (13-14 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 250

Структура роботи. До складу кваліфікаційної роботи магістра входить розрахунково-пояснювальна записка та графічний матеріал. Розрахунково-пояснювальна записка містить вступ, 4 розділи, загальні висновки, список використаної літератури і додатки. Обсяг роботи: розрахунково-пояснювальної записки –72 арк. формату А4, графічна частина – 9 аркушів формату А1.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТЕСТУВАННЯ ПРОГРАМНИХ ПРОДУКТІВ

Тестування ПЗ – це процес дослідження та випробування програмного продукту, має дві основні цілі [2]:

1. Перевірка того, що програмне забезпечення повністю відповідає вимогам і потреб замовника;
2. Виявлення ситуацій, в яких поведінка програми єнеправильним.

Причому, перевірка того, що програмне забезпечення буде відповідати потребам замовника є головний метою тестування. Інакше, немає сенсу перевіряти систему на правильність роботи, якщо вона від самого початку не буде відповідати основним вимогам і потребам замовника.

Так як кінцевою метою тестування є забезпечення якості продукту, то необхідно визначити, що таке якість продукту.

Під якістю розуміють значення, відповідно до якого компонента, система або процес відповідає зафіксованим вимогам і очікуванняи користувача чи замовника.

1.1. Життєвий цикл тестування ПЗ

Життєвий цикл тестування програмного забезпечення (STLC) — це послідовність певних операцій, які виконуються щоб забезпечити достатній рівень якості програмного забезпечення. STLC включає як перевірку, так і валідацію [4].

STLC — це високоякісна стратегія, яка безпосередньо пов’язана з життєвим циклом розробки програмного забезпечення (SDLC) і є його частиною, яка, у свою чергу, є структурою з 6 основними принципами:

–аналіз вимог;

–планування;

–інженерія та дизайн;

–розробка програмного забезпечення;

–тестування;

–розгортання.

Простіше кажучи, SDLC — це цикл, у якому дії на кожному етапі відображаються на наступних етапах. STLC також має свої етапи та найбільш тісно перетинається з SDLC на п’ятому етапі, який я опишу нижче.

Ці два поняття тісно пов’язані один з одним (рис.1.1), але вони одночасно переслідують різні завдання з тією самою метою, а саме:

–збір вимог в потрібному вигляді та розробка заявленого функціоналу (як для SDLC);

–аналіз вимог, супровід клієнта та команди розробників, підтвердження якості реалізованого функціоналу (як для STLC).

Загальною метою є задоволення клієнта та досягнення найвищого можливого балу на етапах перевірки та підтвердження.

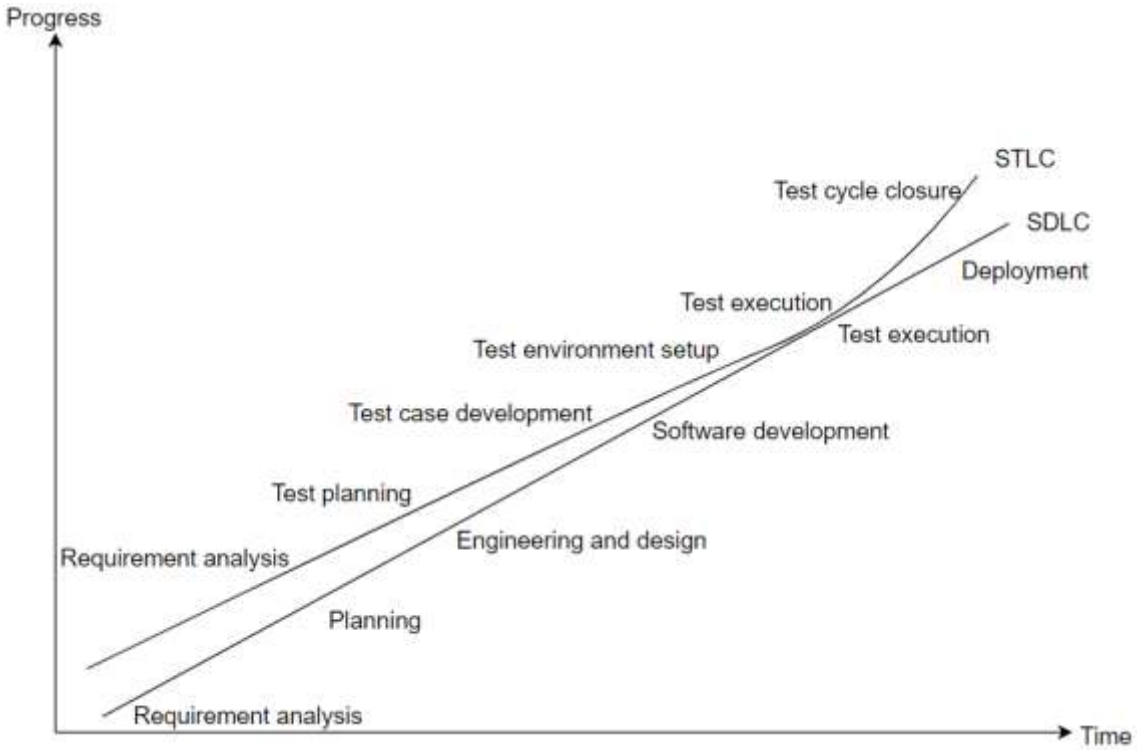

Рис 1.1. Роль STLC у SDLC

Лінії SDLC і STLC рухаються паралельно протягом більшої частини проекту, але починають швидко зближуватися на етапі розробки програмного забезпечення з глибокою синхронізацією на етапі тестування SDLC. Цей графік актуальний для

багатьох різних типів проектів, а не лише для великих чи незалежних, і залишатиметься незмінним для виконання завдання в межах проекту та зворотним для проекту з величезною кількістю ітерацій (але в цьому випадку лінії будуть частіше розходитися і сходитися).

STLC має кілька взаємопов'язаних фаз і загалом дуже схожа на систему SDLC. Ці фази є послідовними і називаються:

–аналіз вимог;

–планування тестування;

–розробка тестових випадків;

–налаштування тестового середовища;

–виконання тесту;

–закриття випробувального циклу.

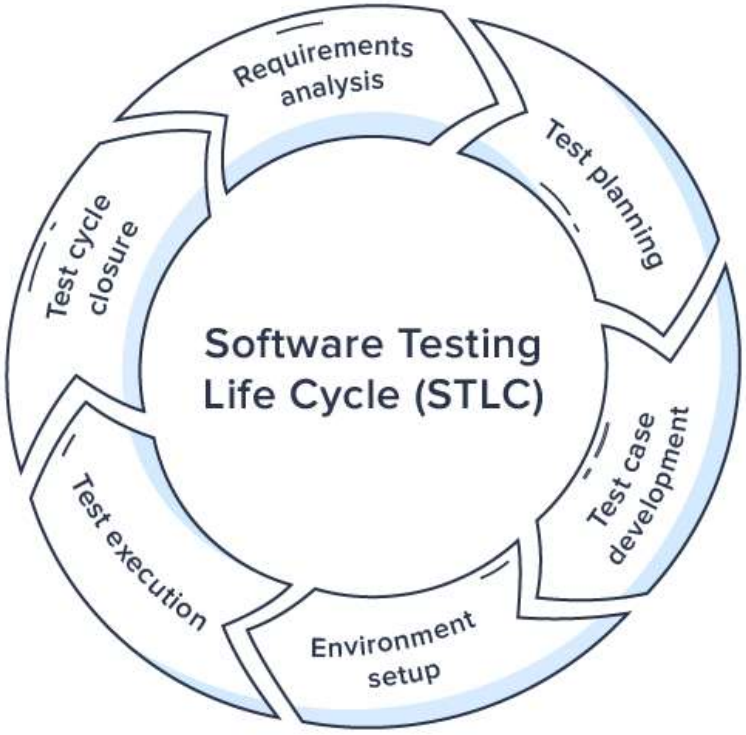

Рис.1.2. Життєвий цикл тестування ПЗ

Перш ніж перейти до опису фаз життєвого циклу тестування, необхідно сказати, що на будь-який фазі є вхідні параметри, без яких ця фаза не може розпочатись.

1.1.1.Аналіз вимог

Аналіз вимог є однією з перших стадій розробки програмного забезпечення. Під час цього етапу також визначається потенційна необхідність у впровадженні автоматизованого тестування та дозволяє здійснювати економічні розрахунки трудових витрат на основі оцінки проекту. Також цей час використовується для обговорення та документування критеріїв входу та виходу.

Критерії входу та виходу в тестуванні програмного забезпечення є необхідними на всіх рівнях ЖЦ розробки ПЗ (STLC). Критерії входу визначають передумови, які повинні бути виконані перед початком тестування, тоді як критерії виходу визначають передумови для його завершення [7].

А тепер уявіть, що нова технічна компанія розробляє платформу онлайн-навчання для студентів СІ, яка буде запущена через шість місяців. Платформа включатиме різні типи освітнього контенту, такі як відео, вікторини та інтерактивні вправи і буде доступна на настільних і мобільних пристроях.

Стартап edtech проводить дослідження ринку та опитування користувачів, щоб визначити цільову аудиторію та її переваги та потреби. Вони виявили, що більшість учнів СІ віддають перевагу інтерактивному та гейміфікованому контенту, а вчителі віддають перевагу контенту, який відповідає стандартам їхньої навчальної програми та забезпечує функції аналітики та відстеження прогресу.

На основі цих висновків компанія визначає вимоги до платформи, включаючи типи контенту, стандарти навчальної програми, а також інтерфейс користувача та досвід. Наприклад, вони вирішили включити анімаційні відео, ігри-вікторини та віртуальні лабораторії та узгодити їх із загальними основними стандартами в науці та математиці. Вони також розробляють інтуїтивно зрозумілий і привабливий інтерфейс користувача з графікою та звуковими ефектами, а також функції для відстеження прогресу, спілкування з викладачами та доступу до навчальних ресурсів.

Мета проходження фаз життєвого циклу тестування полягає в наданні докладної інформації про якість ПЗ, що дозволить виявити ризики, пов'язані з випуском системи в експлуатацію набагато раніше.

1.1.2.Планування тестування

На цьому етапі складають план тестування [9]. Це конкретизація всіх етапів самого тестування, термінів, учасників та обов’язків. В результаті ми отримуємо дані про:

–учасників та їхні ролі в тестуванні;

–необхідні засоби тестування;

–необхідне тестове середовище.

Повернемося до нашого прикладу. Це коли команда тестування переглядає вимоги та визначає цілі тестування, стратегії та критерії. Вони вирішили зосередитися на функціональному тестуванні, тестуванні зручності використання та продуктивності, а також використовувати як ручні, так і автоматизовані підходи до тестування.

Вони визначають типи тестування, такі як тестування функціональності, зручності використання, продуктивності, безпеки та доступності, а також планують графік тестування, ресурси та інструменти. Наприклад, вони вирішили використовувати JIRA для керування тестами, Selenium для автоматизованого тестування та LoadRunner для тестування продуктивності. Вони також розподіляють конкретні тестові завдання для окремих членів команди та встановлюють регулярні тестові зустрічі та контрольні точки.

Вони також встановлюють цілі та показники якості для відстеження та вимірювання прогресу та результатів тестування. Наприклад, вони поставили за мету досягти 95% проходження для функціонального тестування, 80% рівня задоволеності для тестування зручності використання та 3-секундний час завантаження для кожної сторінки.

1.1.3.Розробка тестових випадків

Цей процес ґрунтується на заздалегідь встановлених вимогах [12]. Зазвичай тестові кейси, що виконуються в автоматичному режимі, розробляються окремо, оскільки кейси що виконуються вручну можуть мати форму шпаргалок. Кожен

тестовий кейс має свій власний життєвий цикл, який включає в себе створення, перевірку та, за необхідності, переробку.

Команда тестування з нашого прикладу створює тестові випадки для кожного типу тестування на основі вимог і цілей тестування. Наприклад, вони створюють тестові випадки для кожного типу інтерактивного вмісту, такого як відео, тести та лабораторні роботи, і перевіряють їх на функціональність, зручність використання та продуктивність.

Вони розробляють тестові сценарії, вхідні дані, очікувані результати, а також тестове середовище та дані, необхідні для кожного тестового випадку. Наприклад, вони створюють різні сценарії для гри-вікторини, як-от режим для одного гравця, режим для кількох гравців і режим з обмеженим часом, і перевіряють їх із різними вхідними параметрами та очікуваними результатами. Вони також створюють тестове середовище, яке імітує різні користувальницькі пристрої та мережі, такі як настільні комп’ютери, мобільні пристрої та 3G, і тестують платформу за різних умов.

Вони також розробляють тестові сценарії та автоматизують процес тестування, де це можливо, щоб підвищити ефективність і зменшити ручні зусилля. Наприклад, вони використовують Selenium для створення автоматизованих сценаріїв для регресійного тестування, а також для перевірки функціональності та сумісності платформи з різними веб-браузерами та операційними системами.

1.1.4.Налаштування тестового середовища

На цьому етапі STLC налаштовується необхідне програмне забезпечення необхідне для проведення тестування. Сюда входять як необхідні ОС так і різні інструменти Selenium, Katalon Studio, та бази даних проекту.

У нашому прикладі команда тестувальників налаштовує тестове середовище, яке включає апаратне забезпечення, програмне забезпечення, мережу та дані, необхідні для тестування. Наприклад, вони встановлюють сервер тестування, який копіює виробниче середовище і встановлюють необхідне програмне забезпечення та драйвери для використовуваних інструментів тестування та фреймворків.

Вони також налаштовують інструменти тестування та інтегрують їх із середовищем розробки для забезпечення безперервного тестування та зворотного зв’язку. Наприклад, вони використовують JIRA для відстеження дефектів і проблем, а також для пов’язування їх із конкретними тестами та вимогами. Вони також використовують Jenkins для автоматизації процесу тестування та ініціювання попереджень і сповіщень про невдалі тести.

1.1.5.Виконання тесту

Команда тестування виконує тестові випадки відповідно до плану тестування та розкладу тестування. Скажімо вони спочатку проводять функціональне тестування, потім тестування зручності використання та, нарешті, тестування продуктивності. Вони аналізують результати тестування, виявляють дефекти та проблеми та повідомляють про них групі розробників для вирішення. Зрештою вони виявляють, що деякі запитання вікторини не відображаються належним чином.

1.1.6.Закриття випробувального циклу

Вкінці необхідно згенерувати звіти про тестування. Вони повинні включати затрачений час, співвідношщення знайдених помилок до позитивних результатів, загальну кількість знайдених і виправлених помилок.

І завершальна частина нашого прикладу. Команда розробників виправляє дефекти та проблеми, про які повідомляє команда тестування, і повторно запускає відповідні тестові випадки, щоб перевірити виправлення. Вони змінюють код тесту, щоб правильно відображати запитання, і повторно перевіряють його за допомогою тих самих тестів, що й раніше.

Потім команда тестувальників проводить остаточне регресійне тестування, щоб переконатися, що виправлення не внесли нових дефектів і що платформа відповідає цілям якості та показникам, встановленим раніше. Вони знову запускають усі тестові випадки, включно з тими, які раніше були невдалими або заблокованими і перевіряють чи всі вони пройшли цей раз.

Після цього команда тестування створює підсумковий звіт про тестування, який містить усі дії з тестування, їх результати та рекомендації, і представляє його керівництву та зацікавленим сторонам. У звіті можна підсумувати виконання тесту та результати для кожного типу тестування, а також надати пропозиції щодо подальших покращень і вдосконалень.

Нарешті, стартап-компанія edtech запускає платформу на ринок і стежить за її ефективністю та відгуками користувачів і викладачів. Вони також планують наступний цикл тестування, який включає нові функції та оновлення на основі відгуків користувачів і ринкових тенденцій. Наприклад, вони можуть захотіти додати нову функцію для персоналізованих шляхів навчання та запланувати новий цикл тестування з урахуванням нових функцій.

1.1.7.Критерії входу та виходу в STLC

Цикл тестування розділений на різні етапи, і кожен має власний набір критеріїв входу та виходу. Критерії пов’язані з деякими видами діяльності та результатами.

Критерії входу. Набір умов або цілей, які повинні бути виконані для створення належного та сприятливого середовища тестування, відомий як критерії входу.

Критерії входу, які остаточно визначені та обрані після комплексного аналізу ПЗ та вимог, гарантують точність процесу тестування і їх ігнорування може знизити якість процесу.

Критерії виходу. Критерії виходу описують усі вимоги, які мають бути виконані до завершення тестування на певній фазі.

Усі результати повинні бути завершені, щоб задовольнити критерії виходу. Не повинно бути недоліків, помилок і всі помилки високої інтенсивності та високого пріоритету виправлено.

В табл. 1.1. наведені основні фази і критерії на кожному етапі.

Таблиця 1.1

Фази STLC та критерії їх входу та виходу

Фаза STLS	критерії входу	Критерії виходу	Результати
Аналіз вимог	Документація щодо функціональних і нефункціональних вимог. Визначені критерії прийняття. Документація щодо архітектури програми	Підписана матриця відстеження вимог Техніко-економічний звіт про автоматизацію тестування, підписаний клієнтом	Матриця відстеження вимог Звіт про здійсненність автоматизації
Планування тестування	Документація вимог Матриця відстеження вимог Техніко-технічне обґрунтування автоматизації випробувань	Затверджений план тестування та стратегічний документ. Підписана документація з оцінки зусиль	Документація щодо плану тестування та стратегії Документація оцінки зусиль
Розробка тестових випадків	Документація щодо вимог. Матриця відстеження вимог і план тестування. Звіт про аналіз автоматизації	Переглянуті та підписані тестові випадки та сценарії Переглянуті та підписані тестові дані	Тестові випадки та сценарії Тестові дані
Налаштування тестового середовища	Документація щодо проектування та архітектури системи План налаштування середовища	Налаштування робочого середовища відповідає плану та контрольному списку. Завершено налаштування тестових даних. Успішний дим тест	Набір тестових даних і готове середовище Результати тесту на дим
Виконання тесту	Базова матриця відстеження вимог План тестування Тестовий приклад і сценарії Тестове середовище Налаштування завершених тестових даних Звіти про тестування пристрою та інтеграції	Завершені заплановані випробування Зареєстровані та відстежені дефекти до закриття	Завершена матриця відстеження вимог і статус виконання Оновлені тестові випадки та їх результати Звіти про дефекти
Закриття випробувального циклу	Завершене тестування Результати тесту Доступні журнали дефектів	Звіт про закриття тесту, підписаний клієнтом	Звіт про закриття тесту Показники тесту

1.2. Рівні і види тестування

В залежності від ступеня деталізації існують різні рівні і види тестування Ці ж різновиди тестування тісно повєязані з ЖЦ розробки ПЗ (рис.1.3).

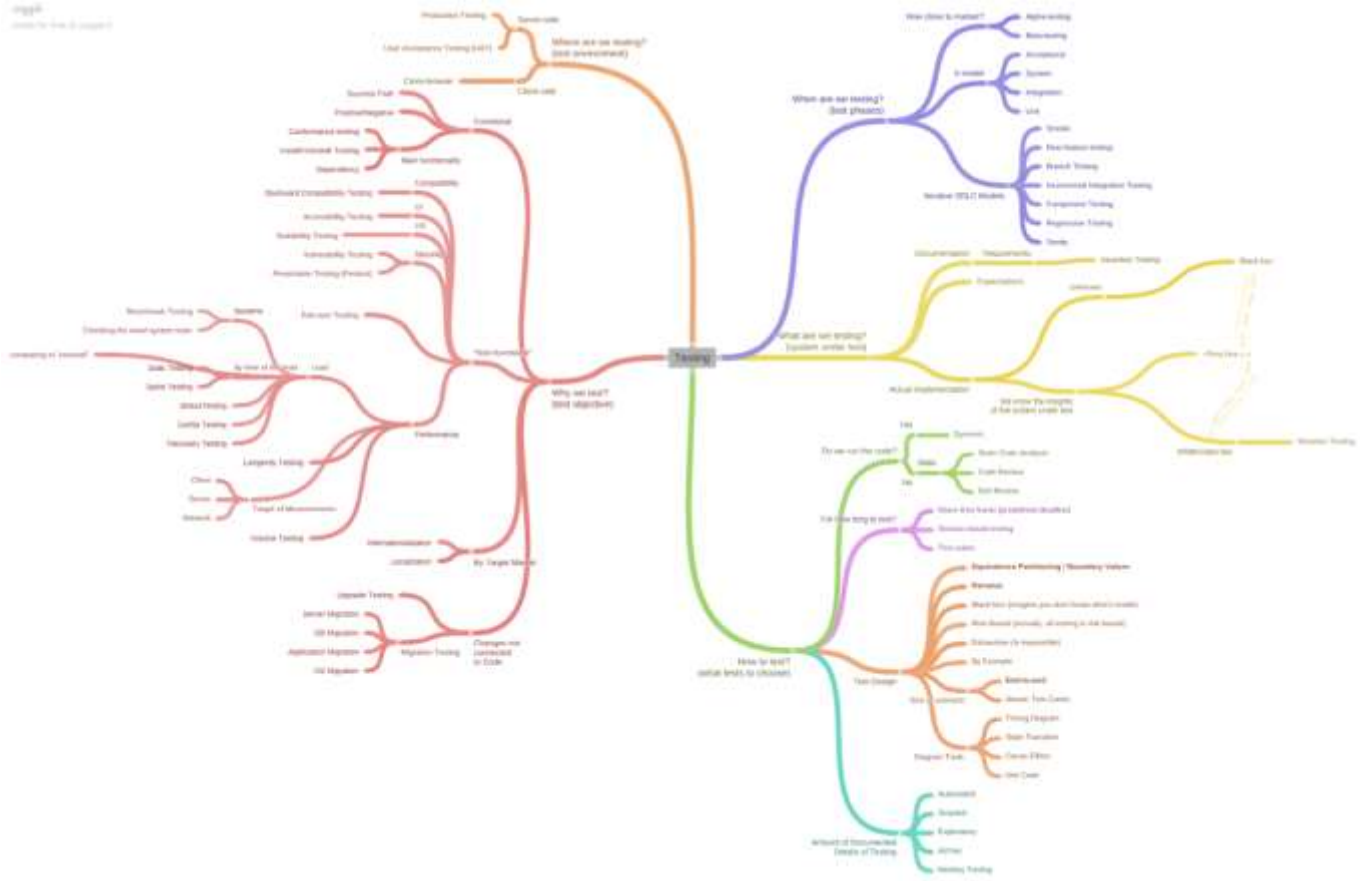

Рис.1.3. Види тестування ПЗ

Інтеграційне тестування (Integration Testing) - спрямовано на виявлення помилок та проблем взаємодії між різними компонентами (модулями, підсистемами) системи під час їх об'єднання в єдину цілісну систему.

Основна мета інтеграційного тестування - перевірка правильності об'єднання окремих компонентів програмного продукту, а також визначення та вирішення можливих проблем в їх взаємодії. Цей вид тестування допомагає виявляти помилки, які можуть виникнути при інтеграції різних частин системи, такі як неправильна передача даних, невідповідність інтерфейсів та інші аспекти взаємодії.

Системне тестування (System Testing) - проводиться на всій системі як єдиній інтегрованій одиниці. Основна мета системного тестування - перевірка всіх аспектів системи під час її роботи як цілісної інтегрованої системи перед виведенням її в експлуатацію.

Приймальне тестування (User Acceptance Testing - UAT) є критичним етапом у ЖЦ розробки ПЗ. Здійснює перевірку того, чи відповідає продукт (система чи програмне забезпечення) вимогам та очікуванням кінцевих користувачів перед його впровадженням в експлуатацію.

Регресійне тестування (Regression Testing) - перевіряє чи впливають внесені зміни в програму на вже існуючий функціонал. Основна мета регресійного тестування - запобігти появі нових помилок у вже перевірених частинах програми після внесення змін, а також впевнитися в тому, що новий код не впливає негативно на існуючий.

"Смок-тест" (Smoke Testing), також відомий як "бейсік-тест" або "бейсікс-тест", використовується для швидкої перевірки основного функціоналу системи чи програми з метою виявлення серйозних проблем перед глибшими етапами тестування.

Програму, що не проходить цей тест, не має сенсу віддавати на більше глибоке тестування.

1.3. Тестування по ступеню автоматизації

Одним з ознак в класифікації тестування є тестування по ступеню автоматизації. Дана ознака припускає два підходи до тестування - це ручне тестування і автоматизоване тестування.

Нерідко виділяють третій підхід - це змішане або напівавтоматизоване тестування.

Ручне тестування передбачає виконання тест-кейсів без допомоги будь-яких програм, що автоматизують роботу людини. Тобто фахівець з тестування перевіряє працездатність системи вручну, як ймовірний користувач.

Автоматизоване тестування передбачає виконання тест-кейсів за заздалегідь написаною програмою спеціальним інструментальним засобом. Основні цілі, для яких розробляють засоби автоматизації тестування, наведено на схемі (рис.1.4.).

Рис 1.4. Цілі автоматизації тестування ПЗ

Змішане або напівавтоматизоване тестування поєднує у собі ручне і автоматизоване тестування, де частина функціональності піддається автоматизації, а інша частина виконується вручну.

Автоматизація тестування представляє набір технік, підходів і інструментальних засобів, що дозволяють виключити людину з виконання деяких завдань в процесі тестування. Незважаючи на це будь-яка автоматизація починається з ручного тестування, так як спочатку тестування необхідно чітко спланувати і написати ручні тест-кейси на основі яких будуть розроблятися автоматизовані тестові сценарії.

Як ручне, так і автоматизоване тестування мають ряд переваг та недоліків, які необхідно визначити, щоб докладніше розібратися в яких ситуаціях слід застосовувати автоматизоване тестування, а в яких випадках ефективніше буде використовувати ручне тестування.

Аналіз ручного і автоматизованого тестування проводився по наступним ознаками:

1) Швидкість виконання тест-кейсів. Автоматизація дозволяє проводити

тести швидше, чим якщо це робила б людина. Особливо це відчувається при проходженні тестів, керування даними, так як необхідно проводити ті самі перевірки велику кількість раз, але з різними наборами даних.

2) Можливість повторного використання. При автоматизованому тестуванні написані тестові сценарії можна використовувати надалі на проекті при його черговому оновленні скільки завгодно раз і з найменшими витратами за часом на відміну від проведення ручного тестування.

3) Людський фактор. Так як деякі перевірки необхідно повторювати велику кількість разів, та й тестів на проекті може бути дуже багато, то можна стверджувати, що людина гарантовано помилиться. Тоді як програма не допустить цих помилок по неуважності чи необережності.

4) Виконання тестування продуктивність. Такі тести можуть бути непосильними людині за низкою факторів, наприклад, їх складність або швидкість виконання. Наприклад, дослідження продуктивності, коли необхідно з високою швидкістю виконувати певні дії і фіксувати дії для великого обсягу параметрів.

5) Кваліфікація персоналу. Часто автоматизацію тестування називають проектом усередині проекту, тому у людей, які займаються автоматизацією тестування, повинна бути, технічна кваліфікація вище, чим у людей, займаються ручним тестуванням.

6) Розробка і супровід тест-кейсів. Розробка автоматизованих тестових сценаріїв займає більше кількість часу, а якщо в проекті відбувається ситуації, коли вносяться суттєві зміни, то тести необхідно переробляти. Так на супровід тестів теж йде велике кількість часу.

7) Планування і управління ризиками. Планування і управління ризиками при автоматизації тестування конче важливо, так як може завдати величезних збитків проекту як за часом, так і бюджету.

8) Проведення дослідницького тестування. Автоматизовані тести виконуються суворо по планом, в то час як при ручному тестуванні можна звертати увагу на деталі і можна знайти несподівані помилки.

9) Вартість впровадження. Більшість засобів автоматизації стоять дорого, а

безкоштовні засоби не завжди дають потрібний ефект. Тому важливо планувати впровадження автоматизації.

10) Людський погляд. Можливе існування помилок, які може помітити лише людина. Також існують види тестування, які здатна виконувати тільки людина, наприклад, тестування зручності використання або тестування вимог.

Виходячи з даних ознак були сформульовані основні переваги і недоліки ручного тестування і автоматизованого тестування, які представлені в табл. 1.2

Таблиця 1.2.

Порівняння ручного і автоматизованого тестування

Автоматизоване тестування	Ручне тестування
Переваги	
1. Швидкість виконання тест-кейсів; 2. Більше раннє виявленнядефектів; 3. Можливість повторноговикористання; 4. Відсутнє людський фактор; 5. Економія часу; 6. Ефективність тестових даних; 7. Здатність виконувати тест-кейси непосильні для людини; 8. Здатність збирати, зберігати і аналізувати великі обсягиданих.	1. Дешевизна; 2. Низький поріг входу спеціалістів; 3. Проведення дослідницького тестування; 4. Тестування зручності використання користувача інтерфейсу; 5. Перевірка документації.
Недоліки	
1. Наявність висококваліфікованого персоналу; 2. Розробка та супровід автоматизованих тест-кейсів танеобхідною інфраструктури; 3. Необхідність планування і управління ризиками; 4. Виконання тестів суворо по плану; 5. Дорожнеча; 6. Відсутність людського погляду.	1. Низька швидкість проведення тестування; 2. Неможливо виконувати тестування продуктивність; 3. Вплив людського фактор А; 4. Трудомісткість повторного використання.

Виявлені переваги автоматизації тестування дозволяють збільшити тестове покриття за рахунок впровадження тестів, про які раніше не могло йти мови, тому

що людина з ним не впоралася б, за рахунок можливості багаторазово використовувати тести з різними вхідними даними і за рахунок звільнення ресурсів QA-інженера на створення нових тестів.

Висновки до розділу 1.

Отже в даному розділі проведено огляд етапів життєвого циклу тестування ПЗ. Визначені основні вимоги і етапи планування тестування програмних продуктів.

Також проведено порівняння двох основних видів тестування (автоматичне і ручне), що дало можливість виявити оптимальні сфери їх використання.

Потрібно розуміти, що автоматизація не завжди ефективна. Так як автоматизація вимагає часу та коштів, то її впровадження необхідне добре планувати, оцінюючи все ризики.

РОЗДІЛ 2
МЕТОДИ І ЗАСОБИ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ

Завдяки інструментам автоматизованого тестування можна запускати попередньо записані та попередньо визначені дії, порівнювати результати з очікуваною поведінкою та повідомляти про результати цих ручних тестів інженеру-випробувачу. Після створення автоматизованих тестів їх можна легко повторити та розширити для виконання завдань, які не під силу ручному тестуванню. У результаті автоматизоване тестування ПЗ стало критично важливим компонентом успішних проектів розробки.

Основними перевагами автоматизації є:

Економічна доцільність. Після створення автоматичні тести можна запускати необмежений час без додаткових витрат і значно швидше, ніж ручні тести. Час, необхідний для виконання повторюваних тестів, можна скоротити. Це метод економії часу, який одразу означає економію коштів;

Збільшує охоплення тестом. Дозволяє покращити якість програмного забезпечення за рахунок збільшення складності та обсягу тестів. Більш тривалі тести можна навіть виконувати на багатьох комп’ютерах із різними конфігураціями. Під час кожного тестового запуску автоматизація тестування може легко виконувати тисячі різних складних тестів, забезпечуючи покриття, яке не можуть забезпечити ручні тести;

Покращує точність. Автоматизовані тести щоразу виконують одні й ті самі кроки й завжди записують докладні результати. Тестувальники, які більше не піддаються повторним ручним тестам, мають більше часу для розробки нових автоматизованих тестів програмного забезпечення та роботи зі складними функціями;

Більше можливостей. Автоматизоване тестування може імітувати взаємодію десятків, сотень або тисяч віртуальних користувачів через мережу, програмне забезпечення або веб-програми;

Допомагає розробникам і тестувальникам. Розробники можуть використовувати спільні автоматизовані тести, щоб швидко виявити проблеми перед тим, як відправити їх на QA. Коли зміни вихідного коду перевіряються, тести запускаються автоматично та інформують команду або розробника, якщо вони не вдаються.

У даній роботі буде практика по написанням тест-кейсів для тестування інтерфейсу користувача. Такі тести перевіряють повну роботу системи та імітують дії користувача.

2.1. Robotic Process Automation (RPA) в автоматизованій системі тестування

Robotic Process Automation (RPA) – технологія автоматизації бізнес- процесів, з використанням програмних роботів і штучного інтелекту. У свою чергу програмний робот представляє з себе програму яка імітує дії людини взаємодіючи з інтерфейсом інформаційної системи, наприклад, для збирання інформації або маніпулювання додатками.

Програмний робот має свій віртуальний робочий простір, в якому використовує клавіатуру та мишу для внесення даних та переміщення по екранним формам. Роботи здатні виконувати роботу не тільки на програмному рівні, але і з використанням графічного інтерфейсу, що є відмінною здатністю технології RPA. З використанням графічного інтерфейсу робот спочатку запам'ятовує ті чи інші дії, а потім виконує їх скільки завгодно разів.

Таким чином, впровадження RPA надає користувачам інструмент для вирішення рутинних завдань, що дозволяє суттєво скоротити час роботи бізнес-процесу. Це знижує завантаження співробітників і підвищує ефективність виконання бізнес-завдань, практично виключаючи можливість появи помилок.

Для вирішення цих завдань можна виділити основні навички робота:

– робота із даними. Роботи добре показують себе у завданнях, де потрібно працювати з різними інформаційними системами, наприклад, переносити інформацію з одного місця в інше, розбивати і розносити інформацію в різні;

– приймати прості рішення. Робот здатний приймати прості рішення, які піддаються алгоритмізації. І в випадку, якщо виникає виняток, робот може звертатися за допомогою до співробітника;

– перевірка і валідація даних. Робот може звіряти інформацію з різних джерел між собою, що дозволяє ефективно проводити звірку інформації на відповідність певним критеріям;

– робота з неструктурованими даними. Роботи можуть здійснювати пошук певної інформації у документах;

– управління іншими роботами і співробітниками. Якщо роботів стає багато, то існує можливість впровадження ще одного робота, який здатний контролювати роботу решти роботів і складати розклад їх роботи. Також робот здатний сигналізувати співробітнику про настання певної події;

Відповідно до параметрів робота можна виділити такі переваги впровадження технології RPA:

– оптимізація продуктивності людських ресурсів. Один робот може замінити роботу кількох співробітників;

– підвищення швидкості автоматизованого процесу;

– відсутність помилок, які можуть виникати через потреби людини. Точність виконання завдання - 100%;

– робот працює 24 години на добу, без відпочинку і перерв;

– при змінах процесу, перебудувати робота набагато простіше, ніж людину;

– робот може вести журнали своїх дій.

2.2. Формалізація етапів автоматизованої системи тестування

Функціональна модель системи (рис.2.1.) призначена для вивчення її роботи та взаємодії з внутрішніми та зовнішніми елементами. Ця модель визначає основні функції системи, взаємодію внутрішніх компонентів, взаємодію з зовнішніми елементами, призначення системи та архітектурні особливості. Вона служить

основою для розробки, дозволяючи чітко визначити цілі та завдання системи, а також аналізувати її ефективність. [35].

Рис.2.1. Контекстна діаграма

Вхідними параметрами системи є: вхідні налаштування системи, нова збірка, скрипт автотесту (RPA); вихідними: результат виконання автотесту.

Механізмами на цій схемі представлений відділ автотестування. А керуючими стрілками є: чинна збірка програми і вимоги замовника.

На рис.2.2. представлена декомпозиція діаграми, що показує детальну роботу автоматизованого тесту, що складається з трьох етапів:

- читання налаштувань;
- перевірка необхідності установки збірки;
- перевірка наявності тестів для виконання.

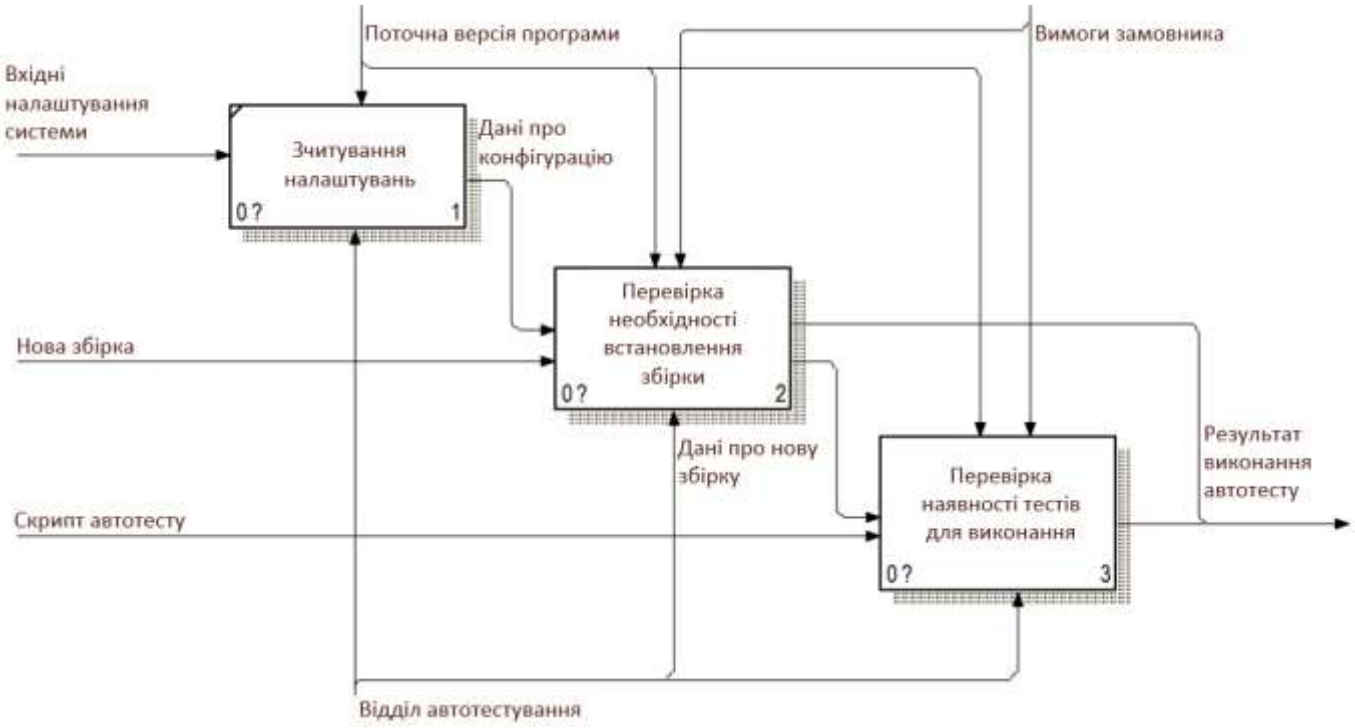

Рис.2.2. Алгоритм автоматизованого тестування

Процес «Перевірка необхідності встановлення збірки» також декомпозується (рис.2.3.):

- перевірка необхідності установки нової збірки;
- перевірка наявності збірки;
- видалення старої збірки;
- встановлення нової збірки;
- перевірка коректності установки;
- перевірка кількості допустимих установок;
- перехід до автотесту.

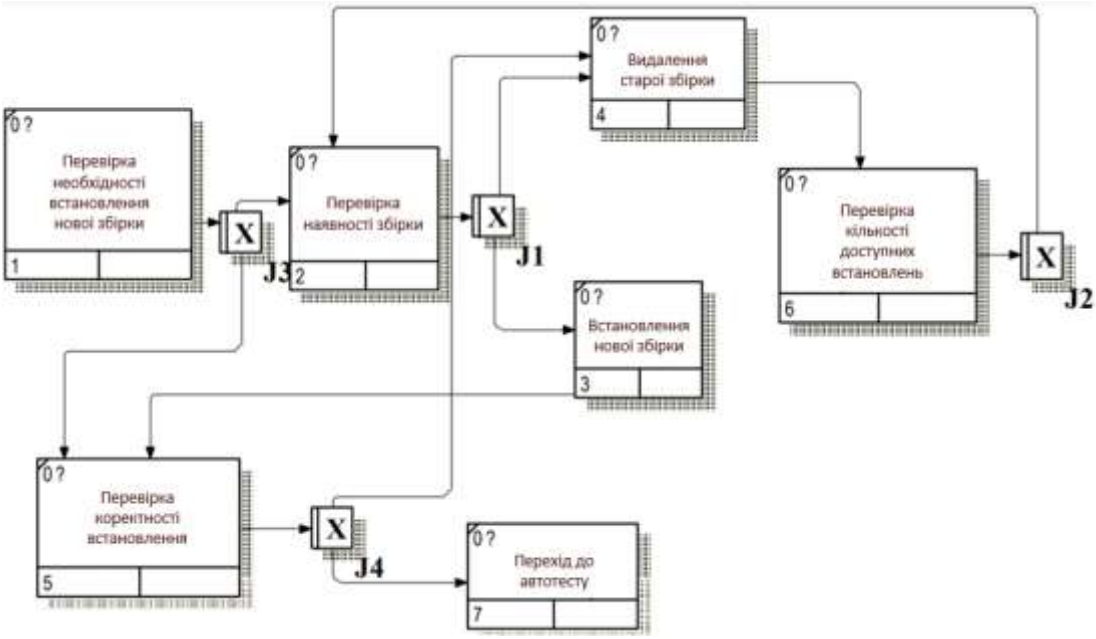

Рис.2.3. Перевірка необхідності установки збірки

Процес «Перевірка наявності тестів для виконання» також декомпозується (рис.2.4.):

– перевірка наявності тіста для виконання;
– виконання чергового тіста;
– збереження результату;
– повідомлення о виконанні тестів.

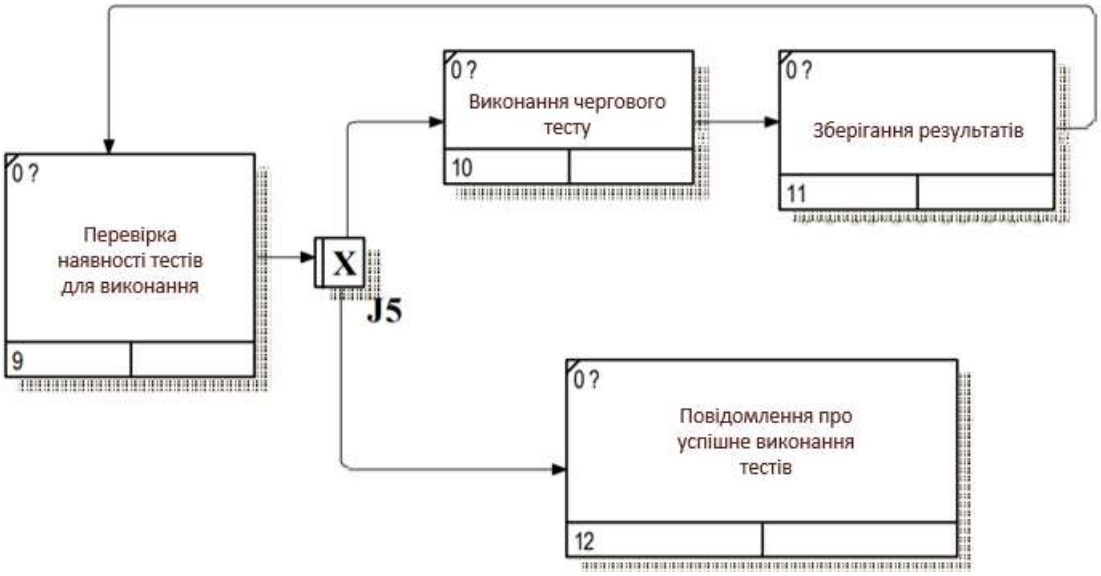

Рис.2.4. Перевірка наявності тестів для виконання

Процес проведення тестування з обліком впровадженої автоматизації представлений на рис.2.5 у нотації BPMN.

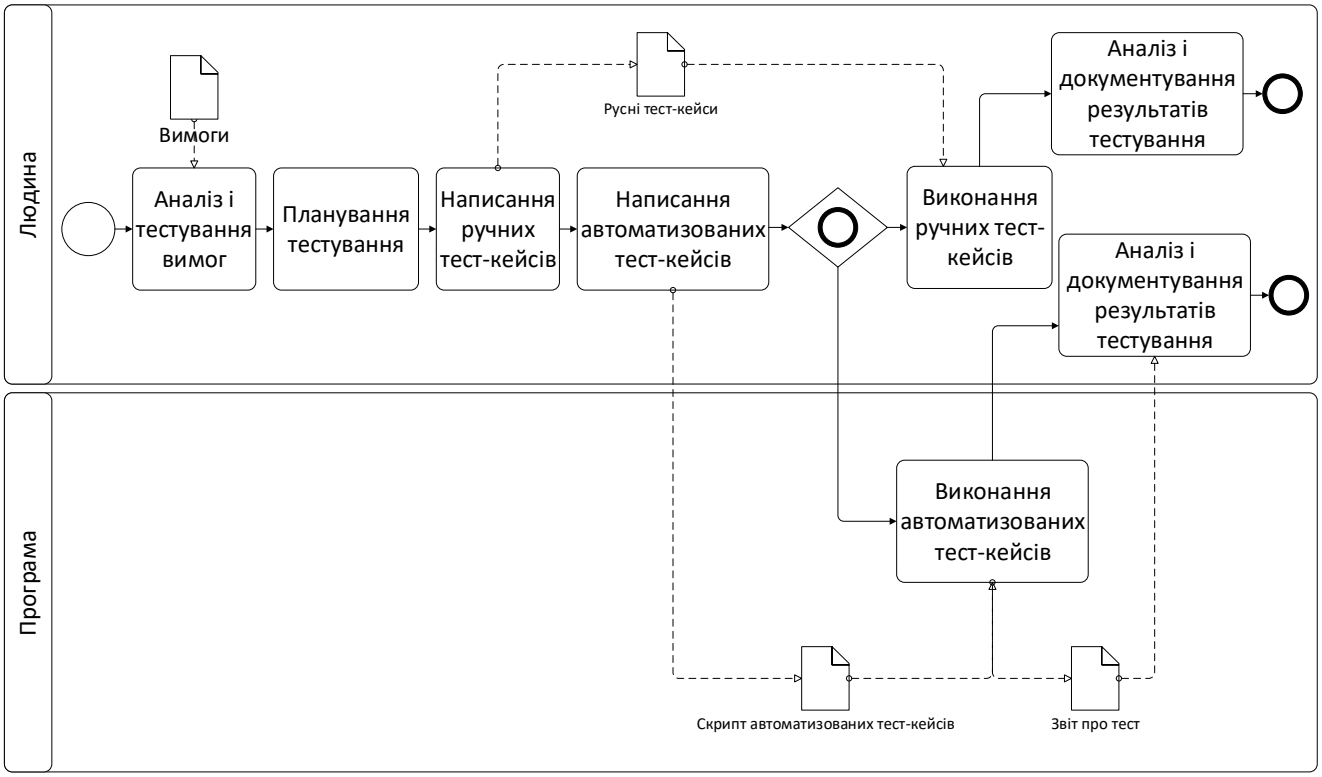

Рис.2.5. Процес проведення тестування у нотації BPMN

Схема (рис.2.5.) описує участь людини і програми в процесі тестування на рис. 2.5 видно, що велику частину процесів виконує людина, а програма для автоматизації займається виключно виконанням тест-кейсів. Але навіть такий, невеликий внесок автоматизації в процес тестування може бути відчутно ефективний, з точки зору швидкості і звільнення ресурсів спеціаліста по тестуванню.

2.3. Використовувані інструменти і технології

Selenium – це інструмент із відкритим вихідним кодом, який був розроблено досить давно, проте останніми роками він активно розвивається. Дана платформа підтримує різні операційні системи і велику кількість популярних браузерів, а також скрипти, які можна писати на різних мовах програмування, наприклад, Python, C#, PHP, Java та інші.

Даний інструмент має свої переваги і недоліки. Серед переваг можна відзначити те, що Selenium має гнучкістю і можливістю написання складних скриптів для тестування програм. А одним із недоліків є те, що фахівець із тестування повинен мати високу технічну кваліфікацію, то є мати знаннями в розроблення програмного забезпечення, а також бути готовим до витрат на часу на написання спеціальних бібліотек і кадрів, які забезпечують виконання необхідних функцій в процесі автоматизованого тестування.

У Selenium існує кілька продуктів:

– Selenium IDE - це доповнення до браузеру Firefox, яке використовується для записи, редагування і налагодження тестів. Selenium IDE спрощує автоматизацію тестування додатківі веб-програм.

– Selenium WebDriver містить набір бібліотек для різних мов програмування, які дозволяють керувати браузером з програми, написаною на даному мовою програмування.

– Selenium Server приймає команди віддалено та виконує їх у браузер.

– Selenium Grid - це мережа з кількох серверів Selenium, масштабуюча процес автоматизації.

Для написання скриптів для UI-тестів використовуватиметься Selenium WebDriver, який представляє собою універсальний інтерфейс для роботи з різними браузерами безпосередньо з коду на мовою програмування. Його універсальність полягає в тому, що тести написані один раз можна запускати для різних браузерів. У якості мови програмування був обраний Python.

2.3.1.Бібліотека для написання автоматизованих тест-кейсів

Python містить спеціалізовану бібліотеку для написання автоматизованих тест-кейсів PyTest. Перевага і недоліки використання даної бібліотеки представлені в табл. 2.1.

Таблиця 2.1

Переваги і недоліки PyTest

Переваги	Недоліки
Детальний звіт з підтримкою колірних схем з коробки. PyTest не вимагає написання додаткових специфічнихконструкцій в тести. Для перевірок використовується стандартний assert з Python. Можливість створення динамічних фікстур	PyTest потрібно встановлювати додатково, оскільки він не входить у стандартний пакет бібліотек Python. Використання PyTest вимагає більше глибокого розуміння мови Python, щоб розібратися, як застосовувати фікстури, параметризацію і інші можливості PyTest.
(спеціальних функцій, які налаштовують тестові оточення таготують тестові дані). Додаткові можливості по на будівництві фікстури. Параметризація тестів - для одного тесту можна задати різні параметри (тест запуститься кілька раз з різними тестовимиданими). Наявність маркувань (marks), які дозволяють маркірувати тести для їх вибіркового запуску. Можливість передавати додаткові параметри через командний рядок для налаштування тестових оточення. Велика кількість плагінів, які розширюють можливості PyTest і дозволяють вирішувати вузькоспеціалізовані проблеми, що може заощадити багато часу.	

Одним із важливих моментів у використанні бібліотеки PyTest єзапуск тестів, так як він відбувається по певним правилам, яким необхідно слідувати під час створення скриптів для тест-кейсів:

– якщо ми не передали жодного аргументу до команди, а написали просто pytest, тест-раннер почне пошук в поточній директорії;

– як аргумент можна передати файл, шлях до директорії чи будь-яку комбінацію директорій і файлів;

– далі відбувається рекурсивний пошук: тобто PyTest обійде всі вкладені

директорії;

– у всіх каталогах PyTest шукає файли, які задовольняють правилу test_*.py або *_test.py (тобто починаються на test_ або закінчуються _test і мають розширення .py);

Всередині всіх цих файлів знаходить тестові функції по наступному правилу:

– всі тести, назва яких починається з test, які знаходяться поза класів;

– всі тести, назва яких починається з test усередині класів, ім'я яких починається з Test.

2.3.2.Паттерн програмування для автоматизації

Page Object - це патерн програмування, котрий дуже популярний в автоматизації тестування і є одним з стандартів при автоматизації тестування веб-продуктів. Це також один з зручних способів структурувати код таким чином, щоб його було зручно підтримувати, міняти і працювати з ним.

Основна ідея складається в тому, що кожну сторінку веб-додатку можна описати в вигляді об'єкта класу. Способи взаємодії користувача зі сторінкою можна описати з допомогою методів класу. У ідеалі тест, який буде використовувати Page Object, повинен описувати бізнес-логіку тестового сценарію та приховувати Selenium-методи взаємодії браузера зі сторінкою. При змінах у верстці сторінки не доведеться виправляти тести, пов'язані з цією сторінкою. Натомість потрібно буде поправити тільки клас, описуючий цю сторінку.

Тут застосовуються ті ж принципи, що і в розробці: ми хочемо підвищити читаність коду і винести в абстрактні методи усі деталі. Тести повинні бути просто і зрозуміло написані, а повторювані шматки коду виділені на окремі функції. У Page Object ми відокремлюємо логіку дій, наприклад, авторизувати користувача від конкретної реалізації (знайти поле пошти, ввести туди дані, знайти поле пароля, ввести туди дані, знайти кнопку і і т.д.).

2.4. Процес автоматизації тестових сценаріїв

2.4.1.Розробка тест-кейсів для ручного тестування

В рамках даної роботи для тестування було обрано дві основні функціональності які зустрічаються у всіх розроблюваних веб-додатках:

– модуль авторизації;

– форма для додавання елементу.

Для кожного з цих модулів були розроблені тест-кейси для проведення ручного тестування.

Тест-кейси або тестові випадки – це правила тестування, які описують сукупність кроків та вхідних даних для перевірки реалізації тестованої функції або її частини.

Тест-кейси поділяються по очікуваному результату на позитивні і негативні. Позитивний використовує тільки вірні дані іперевіряє чи правильно було виконана функція. Негативний - поєднує операції з коректними, так і некоректними параметрами і здійснює перевірку окремих операцій.

Тест-кейс має свою структуру:

– ID. Можуть розділятися в залежності від тестового набору;

– назва тест-кейсу. Повинно бути зрозумілими відображати суть виконуваної перевірки;

– передумова. Список дій, які наводять систему до станом для проведення основний перевірки;

– кроки для відтворення. Перелік дій, переходів системи з одного стану в інший;

– очікуваний результат. Результат, який очікується при відтворенні кроку;

– післяумови. Список дій, що переводять систему в початковий стан. Не є обов'язковою частиною.

Найчастіше тест-кейси групуються в тестові набори, які відносяться до одного тестованого модуля. Це можливо виділяти також з допомогою ID.

Тест-кейс не повинен мати залежність від інших тест-кейсів, нечітких формулюваннь кроків і очікуваного результату, відсутність необхідної інформації про проходження тестів, а також зайвою деталізації.

Приклад розроблених тест-кейсів для ручного тестування представлений на рис. 2.6.

ID	Назва тест-кейсу	Передумова	Кроки з відтворення	Очікуваний результат	Тестові дані
1.1	Авторизація з коректними даними	Перехід на сторінку авторизації	1. У полі Логін ввести коректні дані. 2. У полі "Пароль" ввести коректні дані 3. Натиснути кнопку "Увійти".	1. У полі Логін відображаються введені дані. 2. У полі «Пароль» відображаються введені дані у зашифрованому вигляді (у вигляді крапок). 3. Вхід виконано. Здійснено перехід у профіль користувача.	Логін: admin Пароль: qweQWE123!
1.2	Авторизація з невірним значенням поля "Логін"	Перехід на сторінку авторизації	1. У полі Логін ввести некоректні дані. 2. У полі "Пароль" введіть коректні дані. 3. Натиснути кнопку "Увійти".	1. У полі "Логін" відображаються введені дані. 2. У полі "Пароль" відображаються введені дані у зашифрованому вигляді (у вигляді крапок). 3. Авторизація не здійснено. З'являється повідомлення про помилку.	Логін: admni Пароль: qweQWE123!
1.3	Авторизація з неправильним значенням поля "Пароль"	Перехід на сторінку авторизації	1. У полі Логін ввести коректні дані. 2. У полі "Пароль" ввести неправильні дані. 3. Натиснути кнопку "Увійти".	1. У полі "Логін" відображаються введені дані. 2. У полі "Пароль" відображаються введені дані у зашифрованому вигляді (у вигляді крапок). 3. Авторизація не здійснено. З'являється повідомлення про помилку.	Логін: admin Пароль: 123 q \ veQWE !

Рис. 2.6. Розроблені тест-кейси для ручного тестування

2.4.2.Розробка автоматизованих тест-кейсів

Для розробки автоматизованих тест-кейсів використовувалися інструменти, про яких було сказано в пункті 2.3. Розробка автоматизованих тест-кейсів для тестування користувальницького інтерфейсу проводиться після того, як розроблена HTML-сторінка веб- програми. Так як основа автоматизованих тест-кейсів для користувальницького інтерфейсу будується на застосуванні CSS-селекторів або мови запитів XPath.

Отже, по розробленим тест-кейс, приклад яких представлений на рис. 2.6., будуються автоматизовані тест-кейси.

Структура автоматизованих тест-кейсів будується на патерні програмування – Page Obgect, тому взаємодія браузера з кожною сторінкою веб-додатку описується в окремому класі, так як і набір локаторів для кожної сторінки. Локатори містять константи, які описують доступ до елементів. Винос цих констант спрощує подальшу підтримку тестів. Наприклад, якщо зміниться доступ до елемента, то в коді його доведеться змінити всього один раз. Так діаграма класів програми представлена на рис. 2.7.

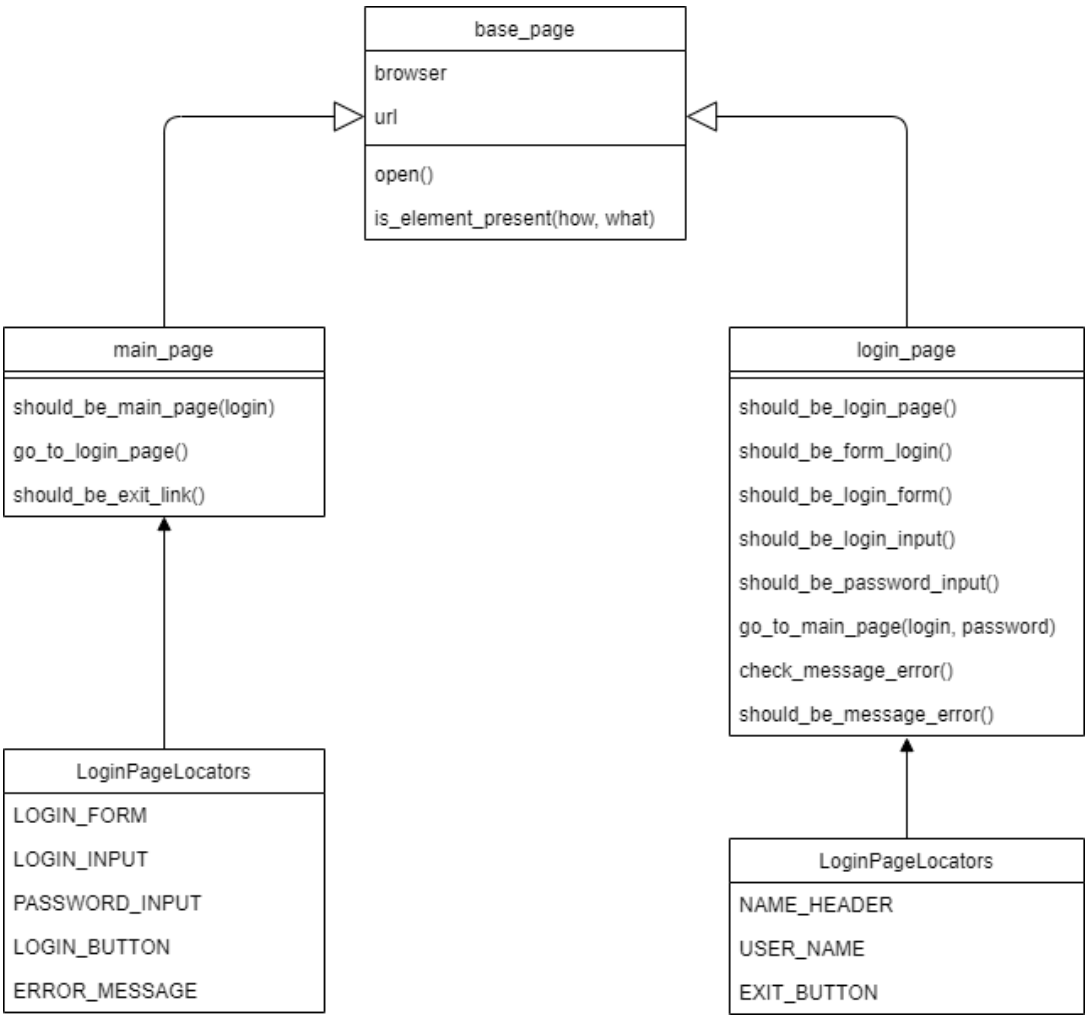

Рис. 2.7. Діаграма класів

Крім файлів, що описують взаємодії з кожною сторінкою та локаторів існує конфігураційний файл.

Конфігураційний файл "conftest.py" призначений для зберігання фікстур або глобальних налаштувань.

Фікстури представляють з себе допоміжні функції, які є частиною тестових сценаріїв. У даному прикладі використовується фікстура для відкриття і закриття браузера. Крім фікстури також використовується і параметризація, що дозволяє налаштовувати тестове оточення. У цьому випадку вибір мови сторінки.

Робиться це з допомогою спеціальною функції "pytest_addoption" і фікстури "request". Так при запуску тестів можна вибирати мову, а якщо мова не вибрана, то він буде встановлений за замовчуванням. Файл конфігурації представлений на рис. 2.8.

```python
import pytest
from selenium import webdriver
from selenium.webdriver.chrome.options import Options


@pytest.fixture(scope="function")
def browser(request):
    language = request.config.getoption("--language")
    print("\nstart browser for test..")
    options = Options()
    options.add_experimental_option('prefs', {'intl.accept_languages': language})
    browser = webdriver.Chrome(options=options)
    yield browser
    print("\nquit browser..")
    browser.quit()


def pytest_addoption(parser):
    parser.addoption('--language', action='store', default="ru",
                     help="Choose language: en or ru")


@pytest.fixture
def language(request):
    return request.config.getoption("--language")
```

Рис. 2.8.Файл конфігурації

Також на прикладі модуля авторизації розглянемо файл, що містить отримані автоматизовані тест-кейси.

Файл із готовими тест-кейсами для модуля авторизації представлений на рис. 2.9. Цей файл описує виключно послідовність кроків для відтворення, завдяки

винесенню взаємодії браузера зі сторінкою в окремі класи. Крім того, для кожного тесту використовуються маркування та параметри.

Маркування призначені для віднесення тест-кейсів у окремі групи, що допомагає при запуску тестів. А параметри представляють вхідні дані для виконання тест-кейсів, наприклад, значення логіна і пароля.

```python
import pytest

from .pages.new_login_page import NewLoginPage
from .pages.main_page import MainPage


@pytest.mark.login
def test_guest_should_be_login_page(browser):
    link = "https://demoqa.com/login"
    login_page = NewLoginPage(browser, link)
    login_page.open()
    login_page.should_be_login_page()


@pytest.mark.login
@pytest.mark.parametrize('login', ["admin"])
@pytest.mark.parametrize('password', ["qweQWE123!"])
def test_guest_go_to_main_page_valid(browser, login, password):
    link = "https://demoqa.com/login"
    login_page = NewLoginPage(browser, link)
    login_page.open()
    login_page.should_be_login_page()
    login_page.go_to_main_page(login, password)
    main_page = MainPage(browser, browser.current_url)
    main_page.should_be_main_page(login)
```

Рис. 2.9.Файл з готовими тест-кейсами для модуля авторизації

Загальний алгоритм перевірки модуля авторизації представлений на рис. 2.10. Цей алгоритм описує дії всіх отриманих автоматизованих тест-кейсів.

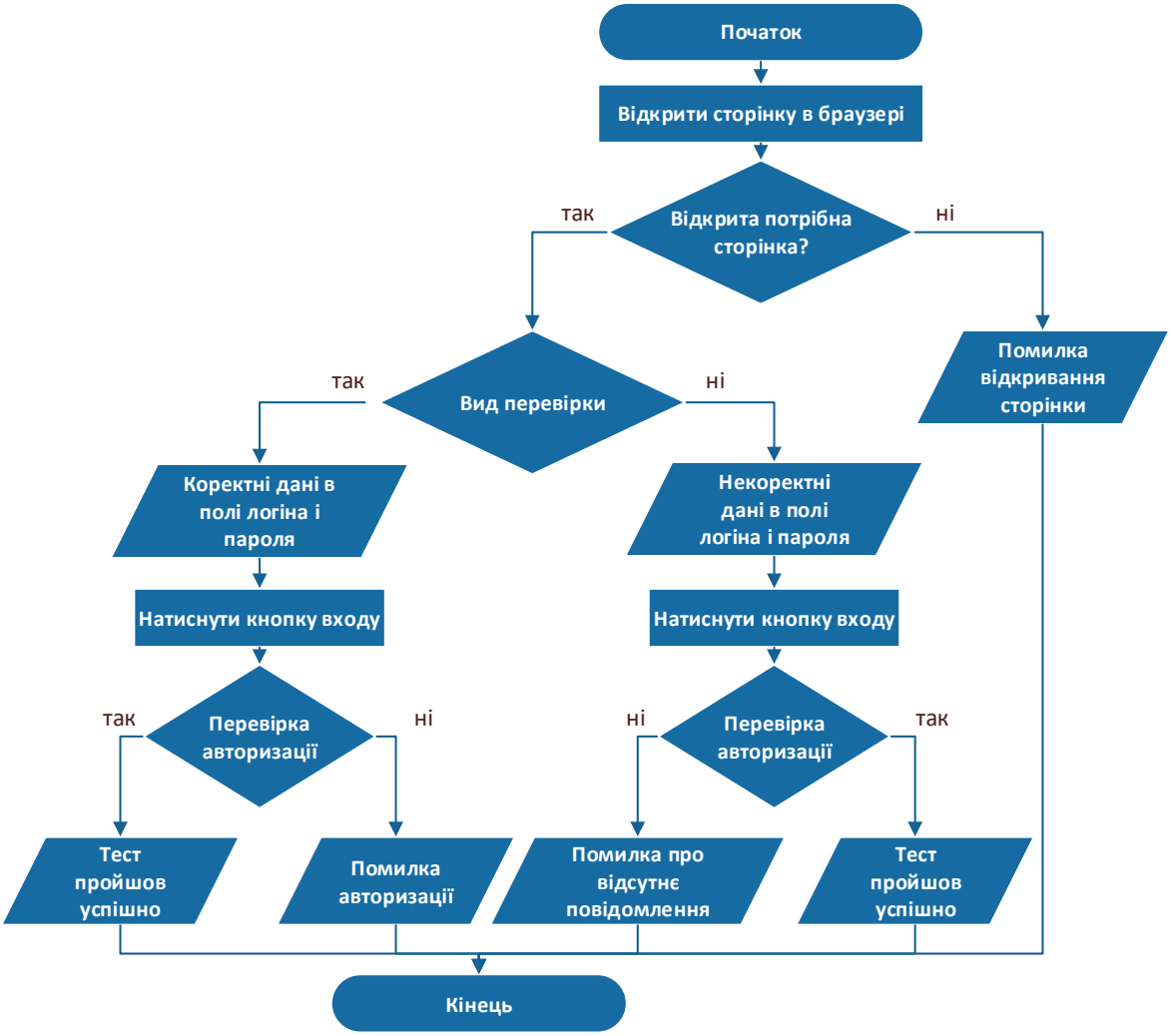

Рис. 2.10.Схема алгоритму для тестування модуля авторизації

2.4.3. Виконання автоматизованих тест-кейсів

Запуск автоматизованих з використанням бібліотеки PyTest тест - кейсів здійснюється через консоль. У команди запуску виконання автоматизованих тест-кейсів є різні параметри, якідозволяють виводити звіти про тестування в зручному вигляді. А також існують параметри, які запускають певний набір тестових сценаріїв. Тест-кейси можна організувати в набори за допомогою маркувань. Звіт про тестування виведений в консолі необхідно аналізувати і документувати, чим і займається спеціаліст по тестуванню.

Результат, що виводиться на консоль, представлений на рис. 2.11. Там також відображається час, за який виконувалось тестування системи.

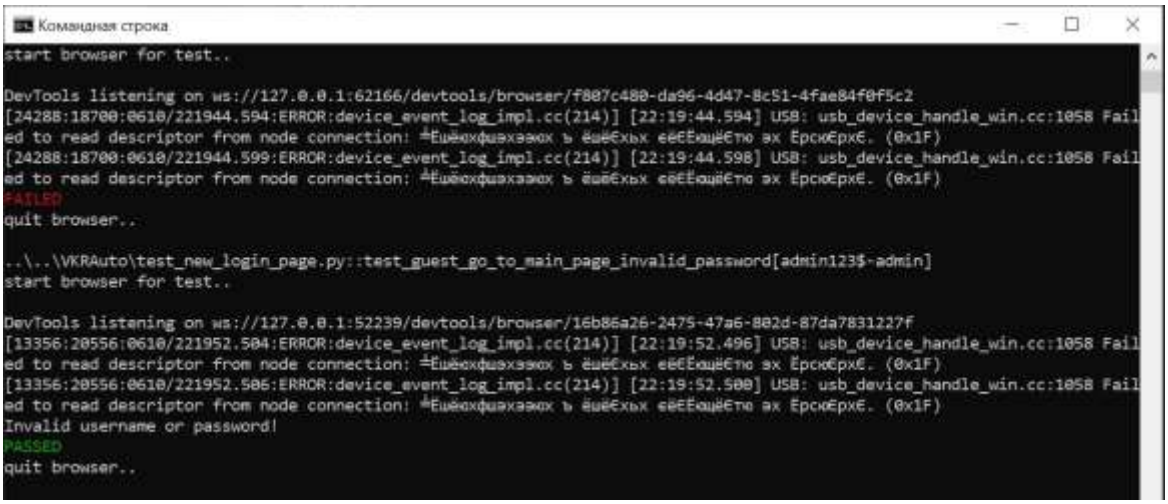

Рис. 2.11. Результат автоматизованого тестування для модуля

За результатами тестування видно, що один тест-кейс пройшов з помилкою, а 6 тест-кейсів були пройдені успішно. Також виявилося повідомлення "Повідомлення про помилку не з'явилося», яке дозволяє зрозуміти суть помилки і швидше визначити її локалізацію.

Висновок до розділу 2.

В даному розділі розглянуті методи і засоби автоматизації тестування ПЗ. Формалізовані основні етапи автоматизованого тестування. Визначені необхідні інструменти для виконання подальших експериментальних досліджень. Також була спроектована модель роботи автоматизованої системи тестування. Наведено обґрунтування вибору програмних засобів.

РОЗДІЛ 3
АПРОБАЦІЯ МЕТОДІВ ТЕСТУВАННЯ З ВИКОРИСТАННЯМ RPA

3.1. UiPath Test Suite

Існує кілька платформ для розробки RPA-роботів. Одна з цих платформ UiPath, яка є світовим лідером. Для автоматизації тестування дана платформа випустила спеціальний продукт UiPath Test Suite.

UiPath Test Suite (рис. 3.1.) побудований для охоплення всього процесу тестування, від планування і проектування до реалізації, виконання і аналізу результатів тестування. Рішення складається з кількох компонентів і кожен компонент відповідає конкретним потребам: Test Manager призначений для управління тестами, StudioPro - для їх автоматизації, Orchestrator - для їх поширення, а Robots - для виконання тестів.

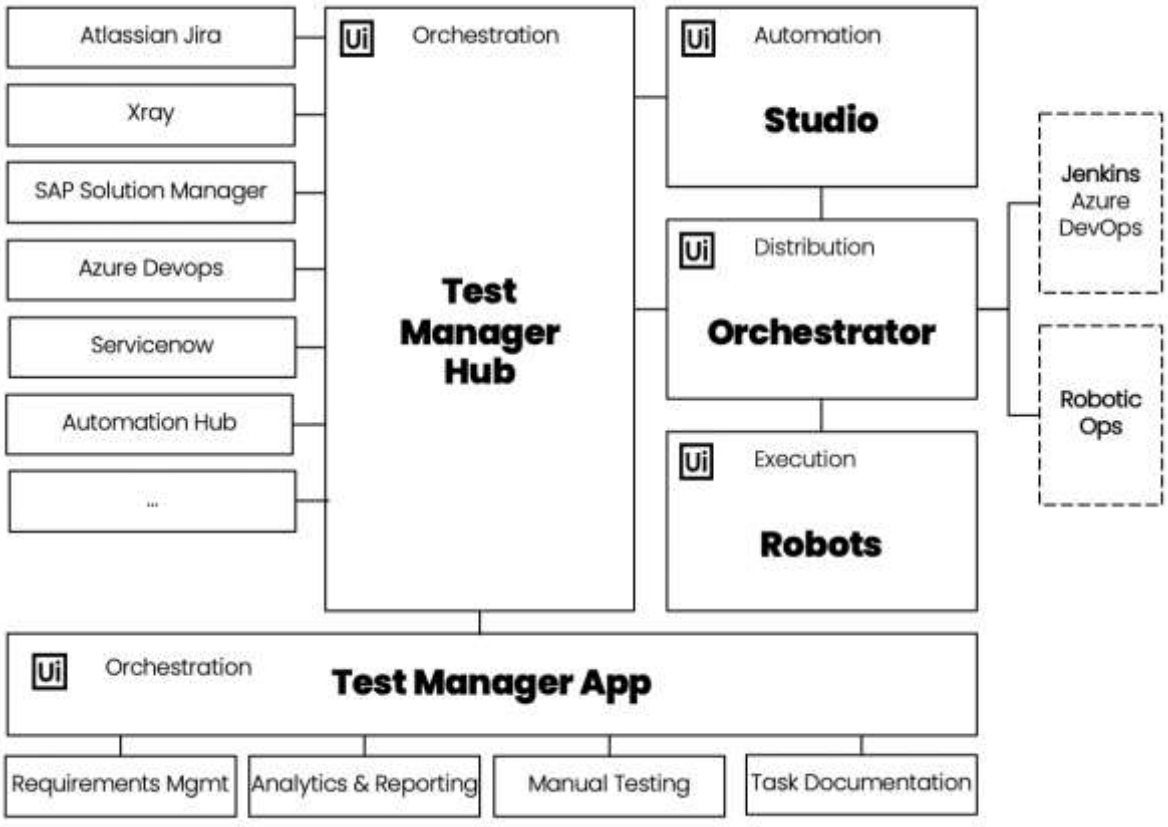

Рис. 3.1. Продукти UiPath Test Suite

Розглянемо кожен компонент окремо і спроектуємо їх призначення на певний ЖЦ тестування програмного забезпечення.

Test Manager призначений для визначення вимог, написання і виконання ручних тестових сценаріїв, прив'язку автоматизованих тестових сценаріїв до вимог, а також моніторинг та аналіз результатів тестування. Диспетчер дає можливість створити та визначити вимоги у Test Manager, але в більшості випадків вимоги фіксуються в інших системах, наприклад, Jira. Також існує Test Manager Hub, котрий призначений для імпорту вимог із зовнішніх систем через певні з'єднувачі.

Тепер перейдемо до UiPath StudioPro (рис. 3.2.), який представляє своєрідну IDE для створення автоматизованих тестових сценаріїв і їх локального налагодження.

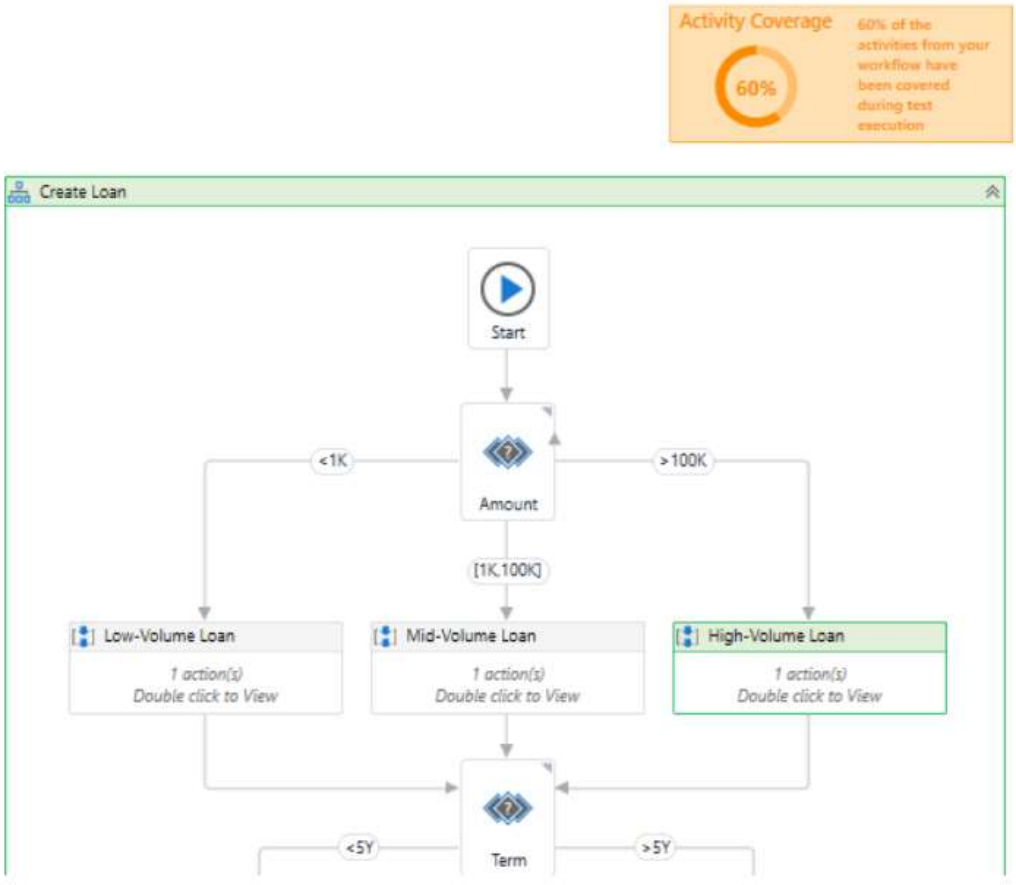

Рис. 3.2. UiPath StudioPro

StudioPro дозволяє запускати тестові набори локально в міру їх створення. Однак є можливість керувати тестовими сценаріями з одного місця і запускати паралельно на кількох машинах з допомогоюUiPath Orchestrator.

Orchestrator – це веб-додаток, який доступний як локально, так і у хмарі. Тестові сценарії потрапляють до Orchestrator після їх публікації з StudioPro. Хід виконання тестових наборів можна відстежувати в Orchestrator. Також Orchestrator має різні способи запуску тестів: при кожній збірці проекту з допомогою плагіна Jenkins, по запланованому розкладу або вручну в будь-який час.

І останнє, але не менш важливе: роботи – це агенти виконання UiPath. Вони підключені до Orchestrator, що дозволяє запускати тестові приклади з конкретними роботами або групами роботів, використовуючи різні налаштування машини.

3.1.1.UiPath Test Suite в життєвому циклі тестування

Як уже було сказано вище рішення UiPath Test Suite охоплює весь життєвий цикл тестування програмного забезпечення. Зв'язок продуктів UiPath Test Suite і етапів життєвого циклу тестування представлено на рис. 3.3.

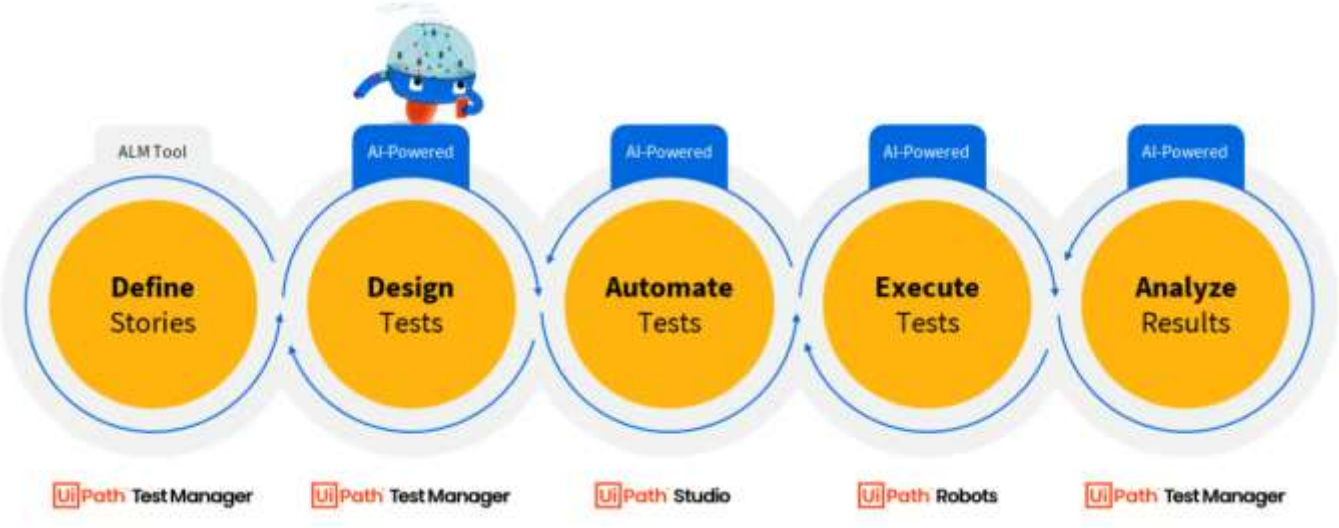

Рис. 3.3. Зв'язок продуктів з життєвим циклом тестування

Функція продуктів для кожного етапу:

– аналіз вимог. Для цього етапу призначений Test Manager, який дозволяє

створювати та керувати тестовими проектами та відповідними вимогами;

– тестове планування. Для даного етапу також призначений Test Manager, Котрий дозволяє записувати і відстежувати тестові сценарії;

– розробка тестових випадків. Для даного етапу призначена StudioPro, яка дозволяє створювати автоматизовані тестові сценарії, пов'язувати тестові сценарії з Test Manager та публікувати їх в Orchestrator;

– виконання тестів. Для цього етапу призначені Orchestrator і Robots, які дозволяють керувати опублікованими тестовими сценаріями, формувати тестові набори, виконувати тести з допомогою груп роботів, створювати і вести розклад виконання тестів, а також керувати журналами виконання тестів та результатами;

– аналіз результатів виконання тестів. Для даного етапу також призначений Test Manager, котрий дозволяє стежити за результатами виконання тестів і аналізувати їх.

3.2. Створення тестових сценаріїв в StudioPro

Тестові сценарії RPA в UiPath StudioPro будуються за формулою Given- When-Then:

– Given - визначає контекст і попередні умови тестового сценарію, це можуть бути: використовуваний додаток, вхідні дані або інша відповідна інформація;

– When - містить дії, які будуть виконані;

– Then - відображає очікуваний результат.

Для реалізації тестових сценаріїв в StudioPro доступні три пакети які виконують усі необхідні дії:

– UiPath.System.Activities - пакет системних дій, який містить усі основні дії використовувані для створення проектів автоматизації;

– UiPath.Testing.Activities - пакет дій по тестуванню, який включає дії, що дозволяють легко перевіряти тестовані системи;

– UiPath.UIAutomation.Activities - пакет дій UIAutomation, який містить усі

основні дії, використовувані для створення проектів автоматизації.

За допомогою цих дій будуються тестові сценарії в вигляді блок-схем.

3.2.1.Перевірки

На етапі Then у побудові тестових сценаріїв необхідно здійснити перевірку по завершенню тестового сценарію. Для цього в UiPath StudioPro передбачені спеціальні дії. Зараз їх три види:

– Verify Expression (Перевірити вираз) - це дія що перевіряє вираз (наприклад, чи рівні дві змінні), його результат є істинним або хибним;

– Verify Expression with Operator (Перевірити вираз з допомогою оператора) - ця дія порівнює результати двох виразів, змінних або аргументів з використанням шести визначених операторів, його результат також є істинним або хибним;

– Verify Control Attribute (Перевірити атрибут управління) – це найбільш універсальна дія, яка дозволяє порівнювати властивості, порівнюється з іншою дією, з виразом, змінною або аргументом. Його результат також може бути істинним або помилковим.

3.2.2. Тестові сценарії

Прості тестові сценарії в StudioPro припускають такі перевірки, де немає різних варіацій у вхідних тестових даних. Наприклад, такими перевірками можна рахувати перехід по посиланням, працездатність або наявність кнопок і елементів на сторінці.

У даному випадку простим тестовим сценарієм можна рахувати попередню авторизацію перед виконанням інших дій у системі. Так в тестовому сценарії буде всього один набір вхідних даних - це коректні логін і пароль.

Щоб побудувати обумовлений вище приклад тестового сценарію, необхідно скористатися набором дій, наданих StudioPro. Спочатку для будь-якого тестового сценарію будується робочий процес, який полягає в блоці "Sequence". Цей процес включає в себе послідовність кроків тестового сценарію. Приклад процесу для виконання авторизації представлений на рис. 3.4.

Рис. 3.4. Робочий процес тестового сценарію

Робочий процес для тестового сценарію представляє з себе:

– відкриття браузера з допомогою дії «Open Browser». Використана дія дозволяє відкривати браузер на сторінці з вказаним URL, а також вибирати тип браузера. "Open Browser" підтримується такими браузерами як IE, Firefox, Chrome, Edge і Custom;

– блок «open browser» містить великий блок "do", котрий виконує дії у даному браузері;

– дія «Delay» встановлює затримку, перш ніж виконувати наступні дії;

– дія «Type Into» дозволяє вводити дані в елементи містять поля введення. Вибір елементів для введення зчитується візуально зі сторінки браузера, при цьому автоматично визначається перебування елемента в ієрархії елементів сторінки. Завдяки цьому можна в налаштуваннях змінювати доступ до елементу;

– дія «Click» ініціює натискання на кнопку або посилання. Існують різні взаємодії з мишею, наприклад, подвійний клік, одинарний клік або ж наведення на

елемент без кліка.

На основі створеного процесу формуються тестові сценарії, представлені у формі Given-When-Then, яка представлена на рис. 3.5.

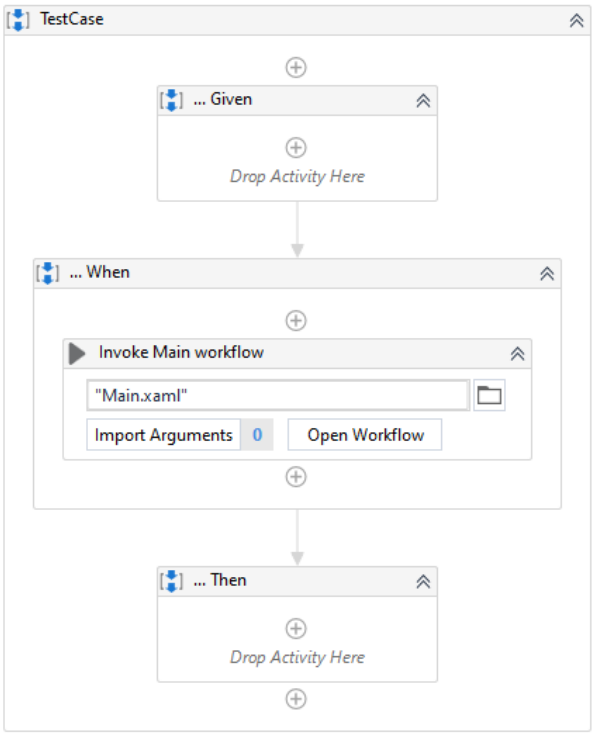

Рис. 3.5. Форма Given-When-Then

У етап When автоматично виставляється створений раніше робочий процес, а етапи Given та Then формуються вручну. На етапі Given зазвичай ініціалізуються вхідні дані. Вхідні дані в простих тестових сценаріях представляють з себе аргументи, приклад переліку яких представлений на рис. 3.6.

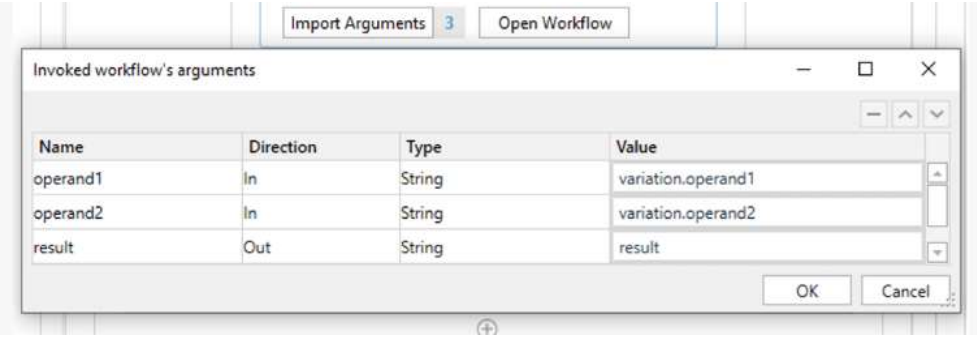

Рис. 3.6 - Перелік аргументів

Цей список визначає значення за замовчуванням у випадку, якщо воно не задано у блоці Given. Але зазвичай у блоці Given слід ініціалізувати ці дані. Це робиться за допомогою дії "Multiple Assign". Результат блоку Given представлений на рис. 3.7.

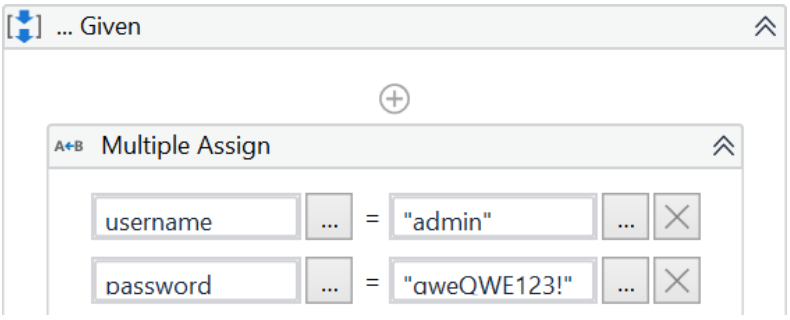

Рис. 3.7. Блок Given

Як вже було сказано вище, блок When автоматично містить створений раніше робочий процес, але в нього необхідно передати аргументи, які ініціалізуються на етапі Given. Готовий блок When представлений на рис. 3.8.

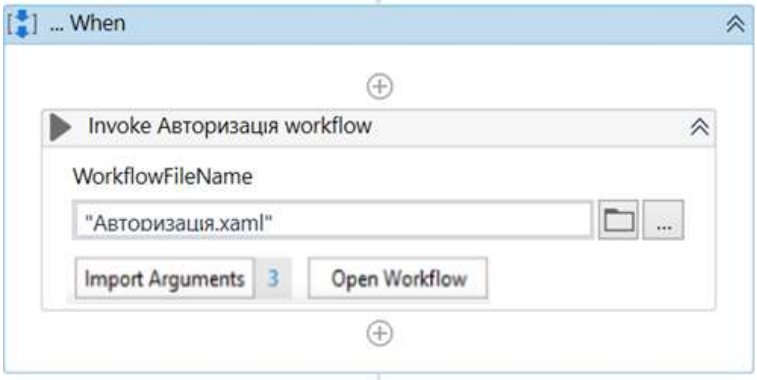

Рис. 3.8. Блок When

Після виконання основних кроків тестового сценарію, необхідно перевірити його успішність. Це перевіряється у блоці Then. У пункті 3.2.1 були розглянуті перевірки, які надає StudioPro. В даному тестовому сценарії перевірка здійснювалася з використанням дії «Verify Expression». Вміст блоку представлено на рис. 3.9.

Тут зі сторінки з допомогою дії «Get Text» в змінну зчитується текст певного елементу. Після чого з допомогою дії «Verify Expression» відбувається перевірка змінної, в яку зчитався текст з сторінки зі змінною, яка була визначена заздалегідь та вказувала правильну поведінку системи після успішної авторизації. І для того, щоб успішно виконувати наступні дії необхідно вийти з профілю.

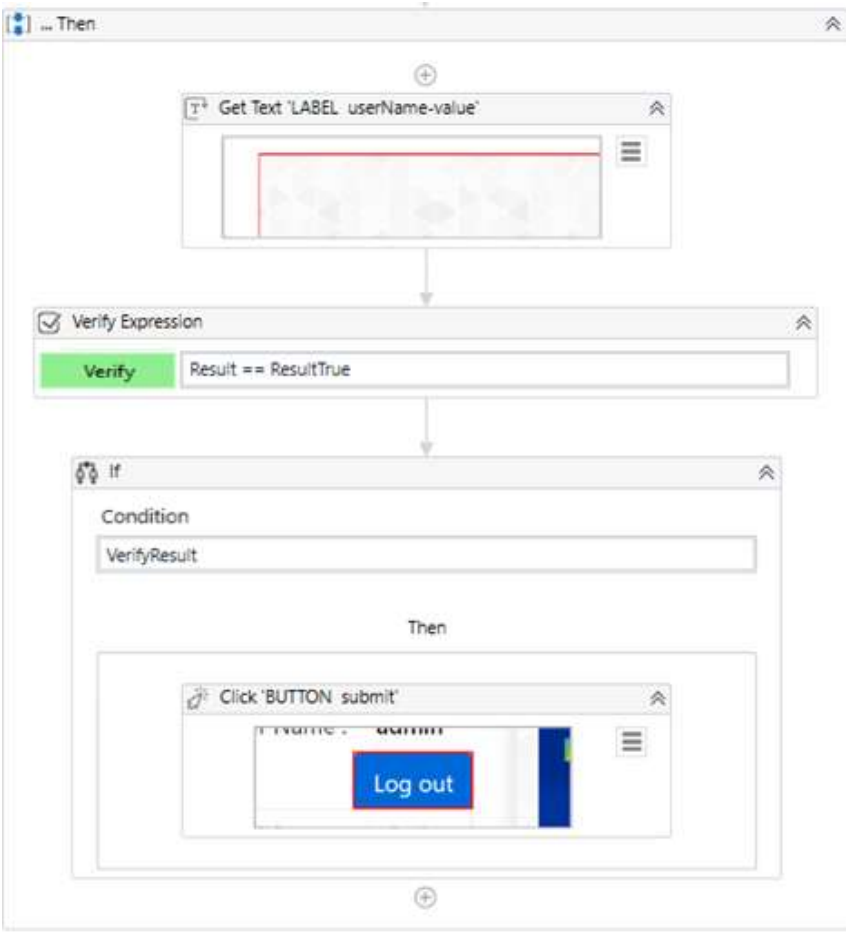

Рис. 3.9. Блок Then

Для цього використовувався блок «If», де за успішної авторизації здійснювалося натискання на кнопку виходу з допомогою дії "Click".

У даному розділі були описані дії для створення простого тестового сценарію.

3.2.3.Тестові сценарії на основі даних

У попередньому розділі описувалося створення простого сценарію, де якості вхідних даних використовувався один набір. Але здебільшого необхідно

здійснювати такі перевірки, де в якості вхідних даних використовується велика кількість наборів цих даних для того, щоб охопити більшу частину варіантів використання системи. В такому випадку автоматизація суттєво дозволяє знизити час виконання таких тестів. У UiPath StudioPro також є варіант створення тестів на основі даних.

При створенні тестового сценарію на основі даних необхідно завантажити набір вхідних значень. Зазвичай набір представляється в файлі Excel, де перший рядок позначає назву майбутніх атрибутів. Приклад файлу представлений на рис. 3.10.

	A	B	C
1	**Username**	**Password**	**Message**
2	admni	qweQWE123!	Invalid username or password!
3	admni	123qweQWE$	Invalid username or password!
4	admin	123qweQWE$	Invalid username or password!
5	admin	QWEqwe123!	Invalid username or password!
6	admni	QWEqwe123!	Invalid username or password!
7			Invalid username or password!

Рис. 3.10. Набір вхідних даних

Після також створюється сценарій в формі Given-When-Then. Відмінність полягає в тому, що блок Given немає необхідності передавати значення, тому що вони автоматично збереглися в системі у вигляді аргументів, які передаються в блок When. У блоці Then також здійснюються перевірки і в цих тестах використовувалася перевірка «Verify Expression with Operator», де два вирази перевірялися за допомогою певних операторів. Блок- схему отриманого тестового сценарію можна побачити на рис. 3.11.

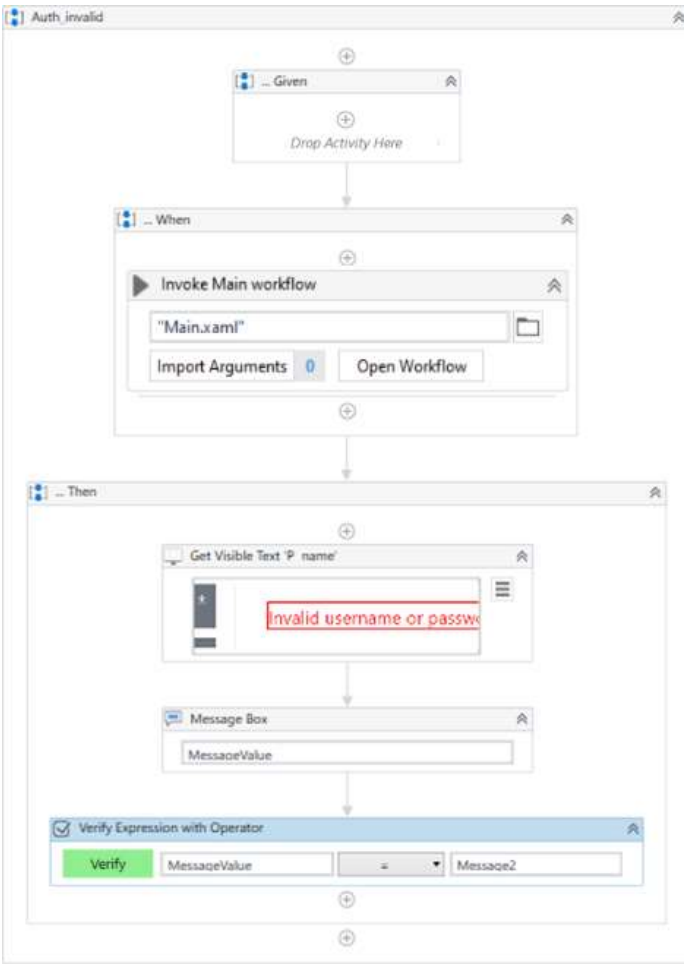

Рис. 3.11. Блок-схема тестового сценарію

Запуск тестових сценаріїв на основі даних можна здійснювати двома способами:

– Запуск файлу. Запускається тестовий сценарій з одним набором тестових даних, визначених за замовчуванням. Зазвичай це перший набір в файлі Excel;

– Запуск файлу з варіаціями даних. Запуск тестового сценарію запускається зі усіма обраними наборами вхідних даних. Перед запуском є можливість вибрати необхідні набори.

3.3. Виконання тестових сценаріїв

Запускати тестові сценарії, розроблені в UiPath Test Studio, можна у різний спосіб. Перший і найпростіший спосіб - це безпосередньо з StudioPro. Приклад звітності після запуску та проведення тестових сценаріїв представлений на рис. 3.12.

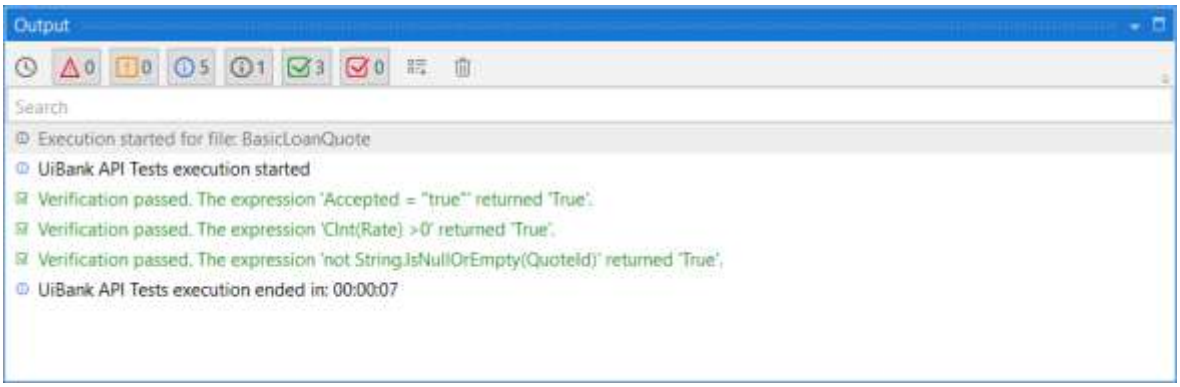

Рис. 3.12. Результат запуску автоматизованих тест-кейсів

Але також тести можна запускати в спеціалізованому середовищі Orchestrator, який керує роботами, виконуючими тестові сценарії. Після публікації тест-кейсів з StudioPro у Orchestrator вони стають пакетами. Об'єднання пакету та середовища робот представляє процес. У свою черга середовище роботів - це набір роботів, який використовується для розгортання процесів. Запуск тест-кейсів провадиться спеціальними роботами, які мають тип «Testing». Також тест-кейси в оркестратори можна об'єднувати тест-набори, які можливо запускати самостійно, за розкладом або з використанням плагіна CI/CD.

Приклад результатів запуску тест-кейсів з Orchestrator представлений на рис. 3.13.

Рис. 3.13. Запуск тестів з Orchestrator

Для кожного результату можна переглянути деталі і детальну інформацію про перевірку і про те, як вона була побудована в StudioPro.

3.4. Порівняння інструментів автоматизації тестування

Внаслідок проведення автоматизації тестування за допомогою двох інструментів було здійснено їх порівняльний аналіз, який представлений у табл. 3.1.

Таблиця 3.1

Порівняльний аналіз інструментів автоматизації тестування

Характеристика	Автоматизація з допомогою Selenium	Автоматизація з допомогою RPA
Знання мов програмування	Необхідно знати хоча б одна мова програмування, підтримуваний даними продуктом	Ні необхідності знати мови програмування
Тестові сценарії	Будуються в вигляді скрипта	Будуються в вигляді блок схем
Доступ до елементам на сторінці	Здійснюється з допомогою CSS-селекторів і мови запитів XPath	Здійснюється запис елемент на сторінці, доступ до елементу визначається системою
Залучення до ЖЦ тестування	Тільки на етапі розробки і виконаннятестових сценаріїв	На всіх етапах ЖЦ тестування
Налаштувати запуск тестів по розкладом	Можна, але використовуючи інші засоби дляпланування	Можливо з допомогою даної системи
Вартість продукту	Безкоштовно	Платно, необхідно отримати ліцензію

За результатами порівняльного аналізу можна виділити одну велику перевагу використання технології RPA і інструменту UiPath для автоматизації тестування - це наявність єдиного автоматизованого простору, який охоплює усі етапи ЖЦ тестування. При використанні даного інструменту немає необхідності використовувати інші системи та засоби, наприклад, для зберігання ручних тестових сценаріїв або зберігання вимог.

Крім цього, цей продукт легко інтегрується з іншими продуктами, що дозволить швидше залучити його в існуючий робочий процес тестування.

3.5. Порівняння результатів проведення тестування

Як об'єкти для порівняння результатів проведення тестування був взятий процес проведення ручного тестування і автоматизованого тестування з використанням Selenium та RPA від розробки тест-кейсів до їх виконання.

Основні характеристики і результати по даними характеристикам представлені в таблиці 3.2.

Таблиця 3.2

Порівняльний аналіз проведення тестування

Характеристика	Ручне тестування	Автоматизоване тестування з допомогою Selenium	Автоматизоване тестування з допомогою RPA
Час розробки тест-кейсів, хв	21	53	32
Час виконання тест-кейсів, хв	7	0,7	0,57

За результатами порівняльного аналізу видно, що часу на розробку автоматизованих тест-кейсів йде більше, чим на розробку ручних тест-кейсів. При цьому технологія тест-кейсів на RPA швидше, ніж на Selenium. Але час на розробку тест-кейсів компенсується їх виконанням, оскільки виконання автоматизованих тест-кейсів проходить в рази швидше, чим виконання ручних тест-кейсів. І також виконання з допомогою RPA виявилося швидше, ніж за допомогою Selenium. Але це також не означає, що автоматизувати потрібно всі модулі і функції.

3.6. Оцінка ефективності впровадження автоматизації

Для оцінки ефективності від впровадження автоматизації було використано показник ефективності, що враховує не грошову ефективність, а ефективність, що залежить від часу роботи.

Як показник для розрахунку ефективності було обрано ROI. ROI - це коефіцієнт повернення інвестицій.

Так як розрахунок розглянуто з токи зору використання ресурсів часу, то формула буде виглядати наступним чином:

$$ROI = \frac{TM - TA}{TM},$$

де TM - витрати часу на проведення ручного тестування, TA - часові витрати на проведення автоматизованого тестування.

При цьому витрати часу на проведення ручного тестування

$$TM = TM_0 + \sum_{i=1}^{n}(TM_e + TM_a + TM_m),$$

де TM_0 - це час, витрачений на розробку початкових тест-кейсів, TM_e - час, витрачений на одноразове виконання набору тест-кейсів, TM_a - передбачуваний час на аналіз результатів, TM_m – час на супровід тест-кейсів.

А витрати часу на проведення автоматизованого тестування

$$TA = TA_0 + \sum_{i=1}^{n}(TA_e + TA_a + TA_m),$$

де TA_0 - це час, витрачений на розробку початкових тест-кейсів, TA_e - час, витрачений на одноразове виконання набору тест-кейсів, TA_a - прогнозований час на аналіз результатів, TA_m - час на супровід тест-кейсів.

При розрахунку даного показника припустимо, що TM_a =5 хв., а TM_m =5 хв.

Для автоматизованого тестування за допомогою Selenium, припустимо TA_a = 5 хвилин, а TA_m = 8 хвилин .

Для автоматизованого тестування з допомогою RPA, припустимо TA_a = 7 хвилин, а TA_m = 7 хвилин .

Розрахунок буде проводиться для 15 випусків продукту, результати ефективності представлені в таблиці 3.3.

Таблиця 3.3

Результати ефективності

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Selenium	-0,8	-0,5	-0,3	-0,2	-0,14	-0,09	-0,06	-0,03	-0,01	0,01	0,02	0,03	0,04	0,05	0,06
RPA	-0,2	-0,1	-0,05	-0,01	0,01	0,03	0,04	0,05	0,06	0,07	0,075	0,08	0,085	0,09	0,092

За отриманим результатам були побудовані діаграми:

–діаграма витрат часу, яка показує час загальний час, витрачений на тестування;

–діаграма ефективності, яка показує ефективність від впровадження автоматизації.

Дані діаграми представлені на рис. 3.14.

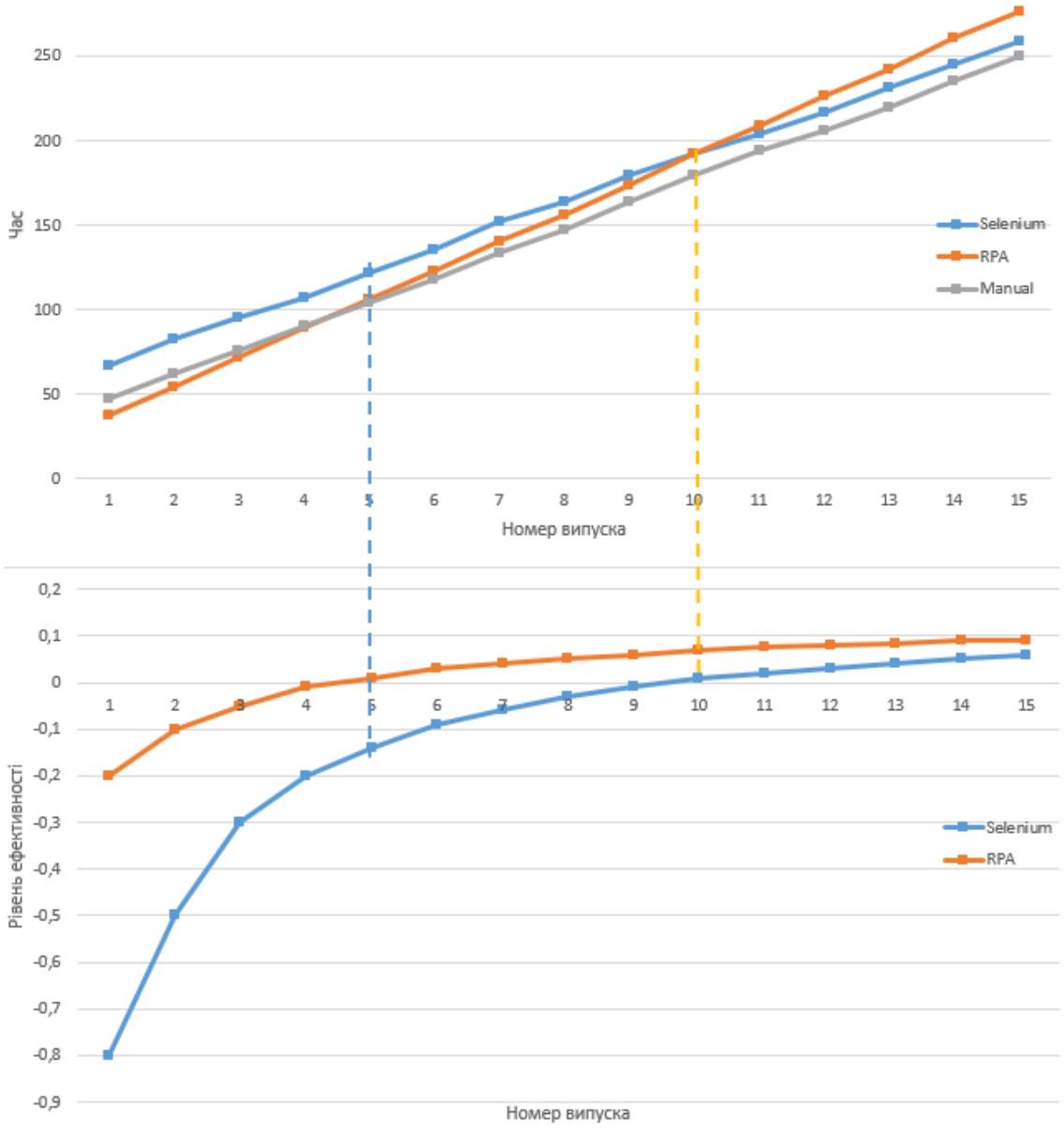

Рис. 3.14. Діаграми ефективності

За результатами аналізу діаграм можна сказати, що ефективність від автоматизації тестування на Selenium виникає лише після 10 випуску, тоді як ефективність від автоматизації на RPA виникає вже після 5 випуску. З цього можна зробити висновок, що для досліджуваного модуля ефективність від автоматизації на RPA вище, чим від автоматизації на Selenium.

Але також потрібно відзначити, що ефективність Selenium стане вище ефективності RPA до 23 випуску. А це означає, що не завжди RPA буде ефективніше. Все залежить від запланованої кількості випусків у системі.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Метою кваліфікаційної роботи магістра є дослідження роботизованих методів тестування програмного забезпечення. Оскільки, проведення робіт по тестуванню програмного забезпечення передбачає використання комп'ютерної техніки, зокрема ПК та периферійних пристроїв, то обов'язковим є дотримання вимог з охорони праці і техніки безпеки.

Для ефективної і безпечної роботи колективу працівників з тестування ПЗ, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці. Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Приміщення, в яких розміщуються робочі місця операторів сервера загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог ДБН В.2.5-56:2014, НАПБ А.01.001-2014 і вимог нормативно-технічної та експлуатаційної документації виробника. Проходи до засобів пожежогасіння мають бути вільними.

Лінія електромережі для живлення комп'ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток

електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97.

У приміщенні, де одночасно експлуатуються понад п’ять комп’ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Комп’ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

Не допускається підключати комп’ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв. Електромережі штепсельних з'єднань та електророзеток для живлення комп’ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення. Нормативні величини освітленості робочих місць для різних видів робіт та відповідних зорових навантажень визначаються ДБН В.2.5-28:2018 «Природне і штучне освітлення».

Організація робочого місця фахівця із дослідження методів та засобів тестування ПЗ повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні

вимоги». Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана комп’ютера, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля згідно з вимогами ДСанПіН 3.3.2.007-98.

Таким чином, у результаті аналізу вимог щодо охорони праці користувачів комп’ютерів, визначено особливості організації робочих місць, вимог з електробезпеки, природного та штучного освітлення для ефективної і безпечної роботи фахівців з дослідження автоматичного тестування програмного забезпечення з використанням програмних роботів.

4.2. Підвищення стійкості роботи об'єктів господарської діяльності у воєнний час

4.2.1.Вплив проникаючої радіації ядерного вибуху на надійність роботи електронного обладнання

Істотним вражаючим чинником ядерного зброї є проникаюча радіація ядерного вибуху, яка являє собою потік гамма-випромінювання і нейтронів. Нейтронне і гамма-випромінювання різні за своїм фізичним властивостям, а спільним для них є те, що вони можуть поширюватися у повітрі на всі боки на відстані до 2,5-3 км. Час дії проникаючої радіації при вибуху зарядів ділення і комбінованих зарядів не перевищує кількох секунд і визначається часом підйому хмари вибуху на таку висоту, при якій γ-випромінювання поглинається товщею повітря і практично не досягає поверхні землі.

В результаті впливу ядерних випромінювань у всіх твердих тілах незалежно від типу структури можуть відбуватися зміщення атомів з утворенням вакантних вузлів і впроваджених атомів.

Експериментально доведено, що при опроміненні більшість параметрів біполярних транзисторів змінюється. Однак серед них можна виділити основний -

статичний коефіцієнт передачі струму, зменшення якого при опроміненні обмежує радіаційну стійкість багатьох класів схем на транзисторах. У загальному випадку зміна цього параметра обумовлено зміною як об'ємних, так і поверхневих властивостей напівпровідників. Випромінювання, що втрачають основну частину своєї енергії в процесі пружного розсіювання, створюють, головним чином, радіаційні дефекти в обсязі напівпровідника, що приводить до зміни часу життя, концентрації і рухливості носіїв заряду. Випромінювання, які при проходженні через речовину втрачають свою енергію за рахунок непружного розсіювання, іонізують газ в корпусі приладу, генерують і збуджують вільні носії заряду, що може привести до зміни поверхневих властивостей напівпровідників внаслідок захоплення генеруються носіїв поверхневими рівнями або осадження заряджених іонів, на поверхню кристала.

П'єзокварцові вироби є найбільш відповідальними функціональними елементами радіоелектронної апаратури. Завдяки вдалому поєднанню механічних, електричних і оптичних властивостей кристалічний кварц зайняв виняткове становище в науці і техніці (кварцові високостабільні генератори, електричні фільтри, ультразвукові пристрої).

Зміна параметрів кварцу в результаті впливу випромінювання має оборотний і необоротний характер. Незворотні зміни параметрів обумовлені процесами зсуву атомів, а оборотні - їх іонізацією.

Конденсатори за своєю конструкцією є системи що складається з електродів (обкладок), розділених діелектриком. Залежно від виду діелектрика конденсатори поділяються на такі класи: паперові, плівкові, слюдяні, керамічні, склокерамічні, електролітичні і оксидно-напівпровідникові. Якщо в слюдяних фольгових конденсаторах використовується просочення секцій органічними матеріалами для підвищення стабільності параметрів, то радіаційна стійкість таких виробів стає порівнянної з радіаційною стійкістю конденсаторів з органічним діелектриком.

При дії іонізуючих випромінювань в електролітичних конденсаторах відбувається радіоліз електроліту, що супроводжується виділенням газоподібних продуктів. В результаті цього спостерігається порушення ущільнень і катастрофічне

погіршення електричних параметрів (в першу чергу, ємності). У рідинних електролітичних конденсаторах з об'ємним пористим анодом з оксидів танталу і ніобію найменш радіаційно-стійким елементом є герметизуюча прокладка з фторо-органічної гуми, яка при дозах втрачає необхідні пластичні властивості. В результаті розчин сірчаної кислоти може вилитися за межі герметизованого обсягу і викликати додаткові зміни характеристик конденсаторів і навіть привести до порушення його працездатності. Причиною погіршення електричних параметрів оксидо-напівпровідникових конденсаторів в умовах впливу радіації при великих інтегральних потоках є зниження опору двоокису марганцю і, відповідно, порушення працездатності конденсаторів.

4.2.2.Шум, вібрація, ультразвук, електромагнітні випромінювання у виробничих приміщеннях для роботи з ВДТ та захист від них

Під шумом розуміють набір багаточисельних звуків, які швидко змінюються за частотою, силою і складаються з ряду гармонік. Шум є загально-біологічним подразником, що діє не тільки на органи слуху, але може викликати порушення роботи серцево-судинної і нервової систем, зумовлювати професійні захворювання.

Основними характеристиками звукових коливань є інтенсивність (сила), частота і форма звукової хвилі. Інтенсивність визначається енергією, що переноситься за 1 с звуковою хвилею через поверхню площею 1 м2, яка перпендикулярна напрямку розповсюдження звукової хвилі. Одиниця вимірювання – Вт/м2.

Гранично-допустимі рівні шумів санітарними нормами встановлені для кожного класу:

− для високочастотних шумів (вище 800 Гц) – 75-85 дБ;

− для середньо частотних шумів (300-800 Гц) – 85-90 дБ;

− для низькочастотних шумів (до 300 Гц) – 90-100 дБ.

З розвитком промисловості все більший контингент людей підпадає під вплив вібрацій, які являють собою механічні коливання, що передаються тілу людини. Основні параметри вібрацій – частота і амплітуда коливань, але на відміну від шуму,

при якому енергія механічних коливань передається через повітряне середовище, при дії вібрацій вона розповсюджується по тканинах і викликає їх коливання або тіла людини в цілому. Найбільш небезпечна вібрація частотою 16-250 Гц, дія якої призводить до вібраційної хвороби. Нормування шуму здійснюється згідно з «Санітарними нормами допустимих рівнів шуму на робочих місцях».

Для запобігання шкідливої дії шуму і вібрації на організм працюючих проводяться технічні, організаційні і медико-профілактичні заходи. Одним з основних технічних заходів є зменшення при експлуатації та на стадії проектування, конструювання обладнання причин шуму і вібрації в самому джерелі утворення. Якщо неможливо ізолювати чи знизити шум і вібрацію самого джерела, потрібно:

− ізолювати джерело шуму або вібрації від навколишнього середовища засобами вібро- та звукоізоляції;

− раціонально планувати виробничі приміщення, що мають інтенсивні джерела шуму;

− збільшувати звукопоглинання внутрішніх поверхонь приміщення шляхом звукопоглинальних покриттів.

Електромагнітне випромінювання – взаємопов'язані коливання електричного і магнітного полів, що утворюють електромагнітне поле а також, процес утворення вільного електромагнітного поля за нерівномірного руху та взаємодії електричних зарядів. Розповсюдження випромінення здійснюється за допомогою електромагнітних хвиль.

До заходів щодо зменшення впливу на працівників ЕМП належать: організаційні, інженерно-технічні та лікарсько-профілактичні. Організаційні заходи здійснюють органи санітарного нагляду. Інженерно-технічні заходи передбачають таке розташування джерел ЕМП, яке б зводило до мінімуму їх вплив на працюючих, використання в умовах виробництва дистанційного керування апаратурою, що є джерелом випромінювання, екранування джерел випромінювання, застосування засобів індивідуального захисту.

Взагалі, засоби індивідуального захисту необхідно використовувати лише тоді, коли інші захисні засоби неможливі чи недостатньо ефективні: при

проходженні через зони опромінення підвищеної інтенсивності, при ремонтних і налагоджувальних роботах в аварійних ситуаціях, під час короткочасного контролю та при зміні інтенсивності опромінення. Такі засоби незручні в експлуатації, обмежують можливість виконання трудових операцій, погіршують гігієнічні умови.

Можна зробити висновок, що при проектуванні та експлуатації електронного обладнання варто враховувати вплив проникаючої радіації внаслідок ядерних вибухів. Розробка та впровадження надійних систем захисту дозволить зменшити ризик виникнення серйозних проблем та забезпечити стабільну роботу обладнання в екстремальних умовах.

Крім того необхідно дотримуватись рекомендацій і використовувати заходи по зменшенню шкідливого впливу в приміщеннях де встановленні ВДТ.

ВИСНОВКИ

У рамках підготовки та написання випускної кваліфікаційної роботи був проведений повний процес тестування програмного забезпечення від аналізу системи до аналізу результатів виконання тестових сценаріїв.

Були досліджені різні методи для розробки тестових сценаріїв та проведення тестування. Ці методи поділяться на два види: ручне і автоматизоване тестування.

Було розроблено тестові сценарії для ручного тестування, а також проведено виконання розроблених тестів вручну.

Крім ручного тестування було проведено автоматизоване на двох різних платформах, таких як: Selenium, який передбачає написання скриптів мовою програмування (був обраний Python) і технологія RPA з використанням платформи UiPath, яка дозволяє розробляти тестові сценарії в вигляді блок схем.

За результатами проведення трьох різних циклів тестування були розраховані показники ефективності від автоматизації на Selenium і на UiPath. А також прораховано тимчасові витрати на тестування. Результати показують, що ефективність від автоматизації настає не відразу, а лише через певну кількість випусків системи.

Виходячи з даної роботи автоматизація RPA на початковому етапі виграє у автоматизації за Selenium, але був помічений факт того, що через кілька випусків автоматизація на Selenium стає ефективніше.

Звідси можна дійти до висновку, що автоматизація на Selenium ефективна для довгострокових систем, передбачуваних велику кількість випусків в процесі розробки, а RPA більше підходить для систем, що мають невелику кількість випусків (приблизно від 5-10). Але для зовсім невеликих систем автоматизація буде не ефективна і краще використовувати ручне тестування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Elfriede D. Automated Software Testing: Introduction, Management, and Performance: Introduction, Management, and Performance. Addison-Wesley Professional, 2008. 608 p.

What is automated testing? URL: https://www.functionize.com/automated-testing (дата звернення: 05.12.2023).

Cem K. Testing Computer Software / Wiley; 2nd edition, 2001. 480 p.

Challenges of Large-Scale Software Testing and the Role of Quality Characteristics. URL: https://www.diva-portal.org/smash/get/diva2:1421638/ FULLTEXT01.pdf (дата звернення: 10.12.2023).

Сучасні підходи до інтеграційного та навантажувального тестування на базі Spring. URL: http://ekmair.ukma.edu.ua/bitstream/handle/123456789/18914/ Nikitchenko_Bakalavrska_robota.pdf?sequence=1 (дата звернення: 10.12.2023).

Beck K. Test Driven Development: By Example. Addison-Wesley Professional, 2021. 242 p.

Rex Black Critical Testing Processes: Plan, Prepare, Perform, Perfect. 1st Edition. AddisonWesley Professional, 2003. 608 p.

Graham Lee Test-Driven iOS Development (Developer's Library). Addison-Wesley Professional, 2012. 256 p.

Dustin E. Automated Software Testing: Introduction, Management, and Performance: Introduction, Management, and Performance. Addison-Wesley Professional, 1999. 608 p.

Crispin L., Gregory J., Agile Testing: A Practical Guide for Testers and Agile Teams 1st Edition. AddisonWesley Professional, 2008. 576 p.

Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. Черкаси: ЧНУ імені Богдана Хмельницького, 2017. 284 с.

Канер К., Фолк Д., Нгуєн Е. Тестування програмного забезпечення. Фундаментальні концепції управління бізнес-додатків. К.: ДіаСофт, 2018. 544

с.

Software Quality Management Techniques and Best Practices. URL: https://www.xenonstack.com/insights/what-is-software-quality (дата звернення: 12.12.2023).

Elijah J. Automation of Requirement Analysis in Software Engineering. International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 5 Issue: 5, 2017. 1173-1188 p.

The Art of Unit Testing: with examples in C# 2nd Edition. Manning; 2nd edition, 2013. 296 p.

Чайковський А.В., Жаровський Р.О., Лещишин Ю.З Конспект лекцій з дисципліни «Дослідження і проєктування комп'ютерних систем та мереж» для студентів спеціальності 123 - Комп'ютерна інженерія. Тернопіль, 2021. 148 с.

Свергун С., Жаровський Р. Тестування програмного забезпечення побудованого на мікросервісній архітектурі. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (7-8 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 92.

Свергун С., Жаровський Р. Тестування програмного продукту, побудованого на мікросервісній архітектурі на основі BDD. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (7-8 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 93.

Newman S. Building Microservices, 2nd Edition. O`Reilly, 2015. 280 p.

James A. Whittaker, Jason Arbon, Jeff Carollo How Google Tests Software. Addison-Wesley, 2012. 281 p.

Фуфаєв Д.Е. Розробка та експлуатація автоматизованих інформаційних систем. К.: Академія, 2017. 304 с.

Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2010. 512 p.

Baboi M., Iftenea A., Gîfu D. Dynamic Microservices to Create Scalable and Fault

Tolerance Architecture. URL: https://www.sciencedirect.com/science/article/pii/S187705091931467X (дата звернення: 12.12.2023).

Ravi Akash. Exploring RPA (Robotic Process Automation) as a Means to Test and Develop User Interfaces. 2023.

Tran Duc; Ho Tran Minh Thu. Workflow methodology development of RPA solution for a Vietnamese bank: a case study of Korkia Oy. 2018.Divya K., Mishra K. The Impacts of Test Automation on Software's Cost, Quality and Time to Market. URL: https://www.sciencedirect.com/science/article/pii/S1877050916001277?via%3Dih ub (дата звернення: 1.12.2023)

Jiménez-Ramírez A., et al. Automated testing in robotic process automation projects. Journal of Software: Evolution and Process, 2023, 35.3: e2259.

Турчиняк Р., Стадник Н. Використання RPA технології для тестування програмних продуктів. Матеріали XI науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (13-14 грудня 2023 року). Тернопіль: ТНТУ. 2023. С.249.

Турчиняк Р., Стадник Н. Використання Uipath test suite для розробки RPA-роботів. Матеріали XI науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі системи та технології» (13-14 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 250.

Лупенко С.А., Луцик Н.С., Луцків А.М., Осухівська Г.М., Тиш Є.В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти усіх форм навчання. Тернопіль. 2021. 34 с.

Додаток А.

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.415.53
Р.В. Турчиняк; Н.Б. Стадник, к.т.н.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ВИКОРИСТАННЯ RPA ТЕХНОЛОГІЇ ДЛЯ ТЕСТУВАННЯ ПРОГРАМНИХ ПРОДУКТІВ

R.V. Turchyniak; N.B. Stadnyk, Ph.D.

USE OF RPA TECHNOLOGY FOR SOFTWARE TESTING

Розробка програмного забезпечення – одна з основних галузей, яка виграє від автоматизації. Роботизація процесів та автоматизація тестування можуть допомогти полегшити навантаження на роботодавців та працівників. Однак, все ще існує багато плутанини щодо цих технологій, оскільки багато людей вважають, що вони описують одне й те ж саме.

Перш ніж ми заглибимося в порівняння RPA та автоматизації тестування, варто дати визначення кожного з цих термінів [1].

Test Automation – це інструмент для розробки програмного забезпечення. Вона має деякі схожі цілі з RPA, оскільки прагне заощадити час, гроші та звільнити працівників від монотонних завдань. Замість дорогого і трудомісткого ручного тестування проектів з розробки програмного забезпечення, програмне забезпечення Test Automation дозволяє командам виконувати швидке, ретельне і глибоке тестування своїх проектів. Цей процес зменшує витрати і призводить до швидших релізів.

Роботизована автоматизація процесів (RPA) – це програмне забезпечення, яке має на меті вивчати та відтворювати комп'ютерні завдання, які традиційно виконує людина. Цей тип автоматизації обмежується простими завданнями на основі правил, які виконують передбачувані кроки [2]. RPA допомагає механізувати великооб'ємні та повторювані завдання. Простіше кажучи, інструменти RPA – це програмні "боти", які можуть спостерігати і вивчати людські завдання з метою їх відтворення без ручного втручання. Роботизована автоматизація процесів (RPA), яку часто називають автоматизацією процесів, – це інноваційний тип програмного забезпечення, що виконує завдання, які традиційно були сферою діяльності людини-оператора.

Інструменти RPA працюють з користувацьким інтерфейсом (UI) так само, як це робить людина.

Однак існують певні недоліки в застосуванні RPA для тестування програмного забезпечення [3]. Інструменти RPA мають обмежений тип інтелекту та підходять переважно для повторюваних завдань з великим обсягом роботи. Вони не можуть адаптуватися до змін, які вимагають творчого мислення чи глибокого розуміння програмної логіки. Також, витрати на впровадження RPA та навчання персоналу можуть бути великими. Отже, вибір між RPA та автоматизацією тестування повинен бути обдуманим, з урахуванням конкретних потреб проекту та можливостей, які пропонують обидва підходи.

Література

1. Jiménez-Ramírez, Andres, et al. "Automated testing in robotic process automation projects." Journal of Software: Evolution and Process 35.3 2023.

2. Singh, Akshay, and Omar Al-Azzam. "Artificial Intelligence Applied to Software Testing." CS & IT Conference Proceedings. Vol. 13. No. 20. CS & IT Conference Proceedings, 2023.

3. Головко, Руслан Джаббарович. Спосіб підвищення ефективності автоматизованого тестування програмного забезпечення. MS thesis. КПІ ім. Ігоря Сікорського, 2022.

УДК 004.415.53
Р.В. Турчиняк; Н.Б. Стадник, к.т.н.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ВИКОРИСТАННЯ UIPATH TEST SUITE ДЛЯ РОЗРОБКИ RPA-РОБОТІВ

R.V. Turchyniak; N.B. Stadnyk, Ph.D.

USING UIPATH TEST SUITE FOR RPA-ROBOT DEVELOPMENT

Robotic Process Automation (RPA) - технологія автоматизації бізнес-процесів, з використанням програмних роботів та штучного інтелекту. У свою чергу програмний робот є програмою, яка імітує дії людини, взаємодіючи з інтерфейсом інформаційної системи, наприклад, для збору інформації або маніпулювання додатками.

Програмний робот має свій віртуальний робочий простір, в якому використовує клавіатуру та мишу, для внесення даних та переміщення за екранними формами. Роботи здатні виконувати роботу не тільки на програмному рівні, а й з використанням графічного інтерфейсу, що є відмінною здатністю технології RPA. З використанням графічного інтерфейсу робот спочатку запам'ятовує ті чи інші дії, а потім виконує їх скільки завгодно разів.

Існує кілька платформ для розробки RPA- роботів. Одна із цих платформ UiPath

UiPath Test Suite побудований для охоплення всього процесу тестування, від планування та проектування до реалізації, виконання та аналізу результатів тестування. Рішення складається з кількох компонентів і кожен компонент відповідає конкретним потребам: Test Manager призначений для керування тестами, StudioPro – для їх автоматизації, Orchestrator – для їх поширення, а Robots – для виконання тестів.

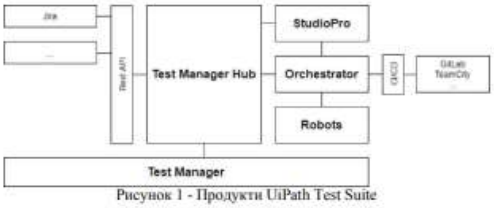

Рисунок 1 - Продукти UiPath Test Suite

Тестові сценарії RPA в UiPath StudioPro будуються за формулою Given-When-Then. Given - визначає контекст і попередні умови тестового сценарію, це можуть бути: додаток, вхідні дані або інша відповідна інформація. When - містить дії, які будуть виконані. Then – відображає очікуваний результат.

Для реалізації тестових сценаріїв у StudioPro доступні три пакети, які містять усі необхідні дії:

– UiPath.System.Activities – пакет системних дій, який містить усі основні дії, що використовуються для створення проектів автоматизації;

– UiPath.Testing.Activities - пакет дій з тестування, який включає дії, що дозволяють легко перевіряти тестовані системи;

– UiPath.UIAutomation.Activities - пакет дій UIAutomation, який містить усі основні дії, що використовуються для створення проектів автоматизації.

З допомогою цих процесів будуються тестові сценарії як блок-схем.