Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Медіасистем та технологій
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Дослідження технологій створення бібліотек для збереження тест-кейсів призначених для етапу тестування мобільних додатків
(тема)

Виконала:
студентка 2 курсу, групи КТСВПВм-19-1

Підгурська М. І.

Спеціальності 186 Видавництво та поліграфія

Тип програми Освітньо-професійна

Освітня програма
Комп'ютерні технології та системи видавничо-поліграфічних виробництв

Керівник доц. Колесникова Т.А.

Допускається до захисту
Зав. кафедри МСТ _____ Ткаченко В.П.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистем та технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 186 Видавництво та поліграфія
Тип програми Освітньо-професійна
Освітня програма Комп'ютерні технології
та системи видавничо-поліграфічних виробництв
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри МСТ_____
(підпис)
«26» жовтня 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Підгурській Марії Ігорівні
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Дослідження технологій створення бібліотек для збереження тест-кейсів призначених для етапу тестування мобільних додатків
затверджена наказом по університету від 23.10.2020 № 1430 Ст

2. Термін подання студентом роботи (проекту) «4» грудня 2020 р.

3. Вихідні дані до роботи Мета – дослідження технологій створення бібліотек для збереження тест-кейсів для тестування мобільних додатків та створення рекомендацій щодо функцій, які повинна мати програма-бібліотека.
Об'єкт – процес тестування мобільних додатків.
Предмет – інструменти для тестування мобільних додатків.

4. Перелік питань, що потрібно опрацювати в роботі Вступ; 1 Аналіз стану проблеми та постановка задач дослідження; 2 Теоретичні дослідження; 3 Експериментальна частина дослідження; 4 Економічне обґрунтування проекту; Висновки.

5. Перелік графічного матеріалу із зазначенням обов'язкових креслеників, схем, плакатів, комп'ютерних ілюстрацій Титульна сторінка; Завдання на магістерську атестаційну роботу; Актуальність дослідження; Мета і задачі роботи; Об'єкт і предмет дослідження; Аналіз стану проблеми; Опис процесу проведення експерименту і отримання даних, Економічне обґрунтування проекту; Висновки.

6. Консультанти розділів роботи

Найменування розділу	Консультанта (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		Підпис	Дата
Основна частина	доц. Колесникова Т. А.		
Економічна частина	проф. Полозова Т.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз стану проблеми і постановка задач дослідження	26.10.2020	викон.
2	Теоретичні дослідження	02.11.2020	викон.
3	Експериментальна частина дослідження	09.11.2020	викон.
4	Економічне обґрунтування проекту	16.11.2020	викон.
5	Оформлення пояснювальної записки	23.11.2020	викон.
6	Оформлення графічної частини	30.11.2020	викон.

Дата видачі завдання 26 жовтня 2020 р.

Студент _____ Підгурська М.І.
(підпис)

Керівник роботи _____ доц. Колесникова Т.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 80 сторінок, 14 рисунків, 16 таблиць, 19 використаних джерел, 2 додатки.

ТЕСТУВАННЯ, МОБІЛЬНІ ДОДАТКИ, БІБЛІОТЕКА, ТЕСТ-КЕЙС, ТЕСТ МЕНЕДЖМЕНТ, УПРАВЛІННЯ ТЕСТУВАННЯМ, МЕТОД АНАЛІЗУ ІЄРАРХІЙ, КРИТЕРІЇ.

Метою атестаційної роботи є дослідження технологій створення бібліотек для збереження тест-кейсів для тестування мобільних додатків та створення рекомендацій щодо функцій, які повинна мати програма-бібліотека.

Об’єктом дослідження даної роботи є процес тестування мобільних додатків. Предмет дослідження – інструменти для тестування мобільних додатків.

В процесі виконання роботи було проведено аналіз стану проблеми тестування мобільних додатків, аналітичний огляд літератури за темою атестаційної роботи, а ще проведено аналіз стану проблеми процесу створення та збереження тест-кейсів. Також було зроблено порівняльний аналіз існуючих програм-бібліотек для збереження тест-кейсів. Методом ієрархій була виявлена найбільш актуальна програма, на базі якої буде будуватися бібліотека. За результатами було сформовано критерії, які повинні бути присутні у бібліотеці тест-кейсів та розроблено рекомендацій щодо програм-бібліотек для збереження тест-кейсів.

Проведено економічне обґрунтування науково дослідної роботи та розраховано економічну ефективність даного дослідження.

РЕФЕРАТ

Пояснительная записка содержит 80 страниц, 14 рисунков, 16 таблиц, 19 использованных источников, 2 приложения.

ТЕСТИРОВАНИЕ, МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ, БИБЛИОТЕКА, ТЕСТ-КЕЙС, ТЕСТ МЕНЕДЖМЕНТ, УПРАВЛЕНИЕ ТЕСТИРОВАНИЕМ, МЕТОД АНАЛИЗА ИЕРАРХИЙ, КРИТЕРИИ.

Целью аттестационной работы является исследование технологий создания библиотек для хранения тест-кейсов для тестирования мобильных приложений и создание рекомендаций по функциям, которые должна иметь программа библиотека.

Объектом исследования данной работы является процесс тестирования мобильных приложений. Предмет исследования – инструменты для тестирования мобильных приложений.

В процессе выполнения работы был проведен анализ состояния проблемы тестирования мобильных приложений, аналитический обзор литературы по теме аттестационной работы, а также проведен анализ состояния проблемы процесса создания и сохранения тест-кейсов. Также был сделан сравнительный анализ существующих программ библиотек для сохранения тест-кейсов. Методом иерархий была обнаружена самая актуальная программа, на базе которой будет строиться библиотека. По результатам были отобраны критерии, которые должны присутствовать в библиотеке тест-кейсов и разработаны рекомендации по программам библиотекам для хранения тест-кейсов.

Проведено экономическое обоснование научно-исследовательской работы и рассчитана экономическая эффективность данного исследования.

ABSTRACT

The explanatory note contains 80 pages, 14 pictures, 16 tables, 19 sources used, 2 applications.

TESTING, MOBILE APPLICATIONS, LIBRARY, TEST CASE, TEST MANAGEMENT, TEST MANAGEMENT, METHOD OF ANALYSIS OF HIERARCHIES, CRITERIA.

The purpose of the certification work is to study the technology of creating libraries to save test cases for testing mobile applications and create recommendations for the functions that should a library program have.

The object of research of this work is the process of testing mobile applications. The subject of research – tools for testing mobile applications.

In the process of performing the work, an analysis of the state of the problem of testing mobile applications was made, an analytical review of the literature on the topic of certification work was made, and was made an analysis of the state of the problem of the process of creating and saving test cases. A comparative analysis of existing library programs for saving test cases was also performed. The method of hierarchies revealed the most relevant program on the basis of which the library will be built. Based on the results, the criteria that should be present in the library of test cases were formed and recommendations for library programs for saving test cases were developed.

The economic substantiation of the research work is carried out and the economic efficiency of this research is calculated.

ЗМІСТ

С.

ВСТУП

Тестування є невід'ємною частиною життєвого циклу розробки програмного забезпечення. Тестування програмного забезпечення – це спосіб перевірити, чи відповідає фактичний програмний продукт очікуваним вимогам, і забезпечити відсутність дефектів програмного продукту. Він передбачає виконання програмних чи системних компонентів за допомогою ручних або автоматизованих інструментів для оцінки одного або декількох цікавих властивостей. Метою тестування програмного забезпечення є виявлення помилок, прогалин або наявність відсутності очікуваного результату на відміну від фактичного результату.

Автоматизація допомагає скоротити час тестування і спростити його процес, використовуючи програмні засоби для виконання тестів і перевірки результатів виконання. Найбільш поширеною формою автоматизації є тестування додатків через графічний користувальницький інтерфейс.

Створення автоматичних тестів, які відповідають потребам компанії, це актуальна і складна задача, так як необхідно не тільки домогтися відповідності вимогам якості тестування продукту, але і забезпечити необхідну економію ресурсів.

Вважається, що створювати тест кейси для кожної нової проблеми, що виникає, не є ефективним рішенням. Для того, щоб якісно керувати процесом тестування, було прийнято зберігати тестові кейси у бібліотеку. Згодом стало зрозуміло, що такий підхід є найбільш ефективним, адже використання бібліотек значно скорочує час створення тест-планів та тест-ранів.

В першому розділі атестаційній роботі пропонується проаналізувати стан проблеми тестування, розглянути літературу по темі дослідження, а також зробити постановку мети і задач дослідження.

У розділі теоретичних досліджень розглянути, насамперед, концепцію тестування, види тестування, життєвий цикл процесу тестування програмного забезпечення та актуальність етапу тестування. Також, другий розділ містить інформацію про процес тестування мобільних додатків, методики створення тест-кейсів, а також відомості про інструменти для управління тестами (бібліотеки тест-кейсів).

Експериментальна частина атестаційної роботи містить планування дослідження, його проведення, формування рекомендацій щодо бібліотек для збереження тест-кейсів, а також перевірку результатів дослідження, використовуючи альтернативний метод дослідження.

У четвертому розділі проаналізовано науково-дослідну роботу в економічному плані. Наведено характеристику дослідження, розраховано трудомісткість і заробітну плату на кожному етапі науково-дослідної роботи, а також розраховано одноразові витрати на етапах. Також, проведено оцінку економічних результатів науково-дослідної роботи та визначено її економічну ефективність.

1 АНАЛІЗ СТАНУ ПРОБЛЕМИ І ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану проблеми

Тестування програмного забезпечення визначається як процедура забезпечення якості програмних продуктів або послуг, що надаються замовникам організацією. Забезпечення якості фокусується на вдосконаленні процесу розробки програмного забезпечення та робить його ефективним та дієвим відповідно до стандартів якості, визначених для програмних продуктів. Забезпечення якості ще відоме як тестування якості.

Метою забезпечення якості є найефективніший і найрізноманітніший спосіб реалізації процесів, пов'язаних із зовнішніми та сервісними процесами, вчасно та в межах бюджету, а також надходження та перевищення всіх операцій та взаємодій із споживачем. Іншими словами, забезпечення якості – це обіцянка того, що клієнт матиме максимально позитивний досвід роботи з компанією незалежно від мети і часу.

Забезпечення якості виходить за рамки просто задоволення очікувань. Компанія, яка охоплює забезпечення якості вищого рівня, включатиме своїх користувачів у всі куточки своєї організаційної структури та діяльності, а не лише в кінцевий фізичний продукт.

Забезпечення якості допомагає компанії створювати продукцію та послуги, що відповідають потребам, очікуванням та вимогам клієнтів. Це дає високоякісні пропозиції продуктів, які формують довіру та лояльність серед споживачів. Стандарти контролю якості змінювались та оновлювались з часом, і стандарти ISO повинні змінюватися, щоб залишатися актуальними для сучасного бізнесу.

Тестування програмного забезпечення допомагає доопрацювати програмне забезпечення або продукт відповідно до вимог бізнесу та

користувачів. Наявність хорошого тестового покриття дуже важливо для забезпечення повного тестування програмних програм та отримання впевненості в тому, що воно працює добре і відповідно до специфікацій. Окрім того, існує незначний дисонанс між бізнес-потребами та вимогами кінцевого споживача, і це також слід враховувати. Якщо ви робите щось на рівні підприємства, то окремі побажання користувачів та клієнтів можна ігнорувати заради загальної стабільності програми. Вирішальним моментом тут є ефективність програми як інструменту для бізнесу. Часто корпорація може навіть пропустити деякі незначні проблеми, якщо бізнес працює і заробляються гроші. Але якщо ви працюєте над програмами для мобільних платформ та масового ринку, то кожен окремий користувач стає важливою людиною, думка якої повинна бути врахована для досягнення успіху в цьому бізнесі.

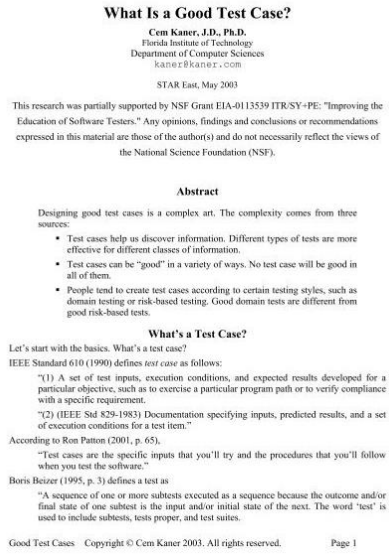
Останньою з серії ISO 9000 є ISO 9001: 2020. Керівні принципи ISO 9001: 2020 включають сильнішу орієнтацію на клієнтів, практику вищого керівництва та способи їх зміни, а також постійне вдосконалення. Поряд із загальними вдосконаленнями ISO 9001, ISO 9001: 2020 включає вдосконалення своєї структури та додаткову інформацію для прийняття рішень на основі ризиків.

1.2 Огляд літератури по темі дослідження

Напрямок тестування розвивається досить стрімко. Основна частина інформації, яка дається в книгах дуже швидко застаріває, однак їх не можна залишати без уваги, оскільки в них даються ті основи тестування, котрі повинен знати кожен майбутній фахівець.

У роботі Кема Кенера (рис. 1.1) «Що є хорошим тестовим кейсом?» [1] добре розглянута проблема створення та подальшого зберігання тест-кейсів, а також проблематика їх оновлення та підтримки. Насамперед, автор зазначає, що зберігати тестові сценарії у програмі для створення цих сценаріїв не є дуже

добрим рішенням, адже в момент оновлення кейсів тестувальники можуть не отримати нотифікацію чи повідомлення про оновлення або не для всіх ці оновлення чи їх формулювання буде зрозумілі.



What Is a Good Test Case?

Cem Kaner, J.D., Ph.D.
Florida Institute of Technology
Department of Computer Sciences
kaner@kaner.com

STAR East, May 2003

This research was partially supported by NSF Grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

Abstract

Designing good test cases is a complex art. The complexity comes from three sources:

- Test cases help us discover information. Different types of tests are more effective for different classes of information.
- Test cases can be "good" in a variety of ways. No test case will be good in all of them.
- People tend to create test cases according to certain testing styles, such as domain testing or risk-based testing. Good domain tests are different from good risk-based tests.

What's a Test Case?

Let's start with the basics. What's a test case?

IEEE Standard 610 (1990) defines *test case* as follows:

> "(1) A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.
>
> "(2) (IEEE Std 829-1983) Documentation specifying inputs, predicted results, and a set of execution conditions for a test item."

According to Ron Patton (2001, p. 65),

> "Test cases are the specific inputs that you'll try and the procedures that you'll follow when you test the software."

Boris Beizer (1995, p. 3) defines a test as

> "A sequence of one or more subtests executed as a sequence because the outcome and/or final state of one subtest is the input and/or initial state of the next. The word 'test' is used to include subtests, tests proper, and test suites.

Рисунок 1.1 – Робота Кема Кенера «Що є хорошим тестовим кейсом?»

Книга Бориса Бейзера (рис. 1.2) «Тестування методом чорного ящика» [2] заповнює давню потребу розробників програмного забезпечення та загальних систем у тому, щоб зробити основні аспекти тестування чорної скриньки доступними в розумінні для однієї всебічної роботи. Написана одним із найвідоміших у світі діячів у галузі тестування, книга є і цінним робочим ресурсом для тестувальників та програмістів, і чудовим практичним вступом для студентів. Доктор Борис Бейзер чітко пояснює принципи тестування в цілому та найважливіші основи тестування методом чорного ящику, що використовуються сьогодні, які передбачають тестування системи на основі бажаної поведінки або функції та на відповідність її специфікаціям. Потім, із повністю опрацьованими прикладами, автор веде читача крок за кроком від специфікацій до готових тестів. У книзі наведено повне пояснення всіх

важливих методів тестування, включаючи ті, що застосовуються до об'єктно-орієнтованого програмного забезпечення.

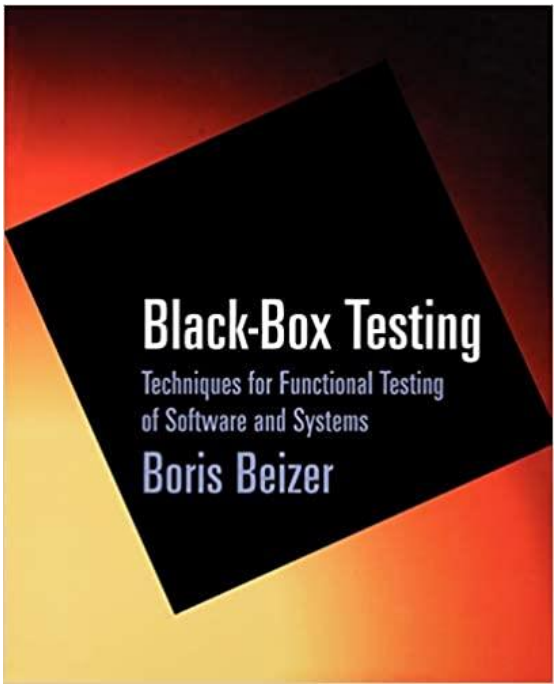


Рисунок 1.2 – «Тестування методом чорного ящика», Борис Бейзер

Дві з найдосвідченіших практиків та консультантів з процесу тестування, Ліза Кріспін та Джанет Грегорі (рис. 1.3), об'єдналися, щоб запропонувати точні відповіді на питання щодо створення тестових сценаріїв та багато інших. У книзі «Agile Testing» [3] Кріспін та Грегорі визначають sprint-тестування та ілюструють роль тестувальника прикладами справжніх sprint-команд. Вони вчать читача, як за допомогою гнучких квадрантів тестування визначити, яке тестування потрібно, хто повинен це зробити та які інструменти можуть допомогти. Книга описує гнучку ітерацію розробки програмного забезпечення з точки зору тестувальника та пояснює сім ключових факторів успіху гнучкої моделі тестування.

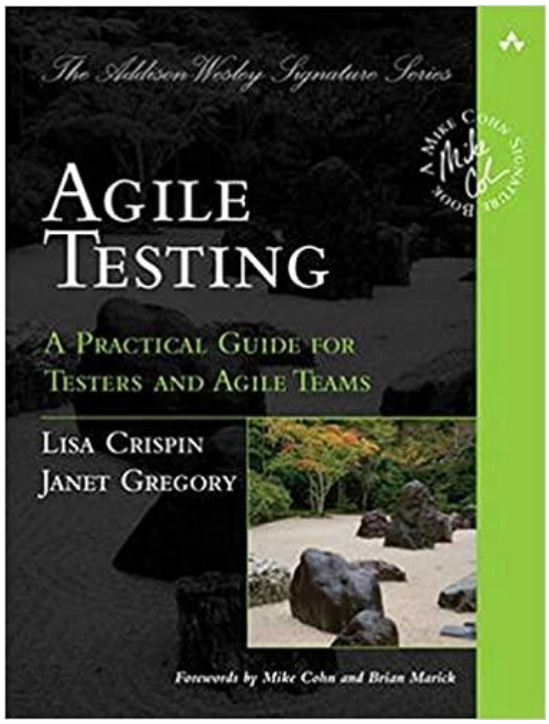


Рисунок 1.3 – «Agile тестування», Ліза Кріспін та Джанет Грегорі

У новому виданні своєї книги «Управління процесом тестування: практичні інструменти та методи управління апаратним та програмним тестуванням» [4] Рекс Блек (рис. 1.4) знайомить читача із кроками, необхідними для управління суворими програмами тестування апаратного та програмного забезпечення. Автор спирається на багаторічний досвід роботи президентом Міжнародної та Американської комісій з тестування програмного забезпечення, щоб запропонувати цей великий ресурс із усіх стандартів, методів та інструментів, які знадобляться фахівцям. Книга висвітлює основні концепції тестування та ретельно вивчає найкращі практики управління тестами та інструменти провідних постачальників обладнання та програмного забезпечення. Покрокові вказівки та реальні сценарії допомагають читачам слідувати всім необхідним процесам та уникати помилок.

Книга охоплює всі стандарти, методи та інструменти, необхідні для великих та малих проектів. Представляє бізнес-кейс для тестування продукції та оглядає останні оцінки автора. Теми включають гнучкі методи тестування,

тестування на основі ризиків, стандарти IEEE, сертифікацію ISTQB, розподілене та аутсорсингове тестування.

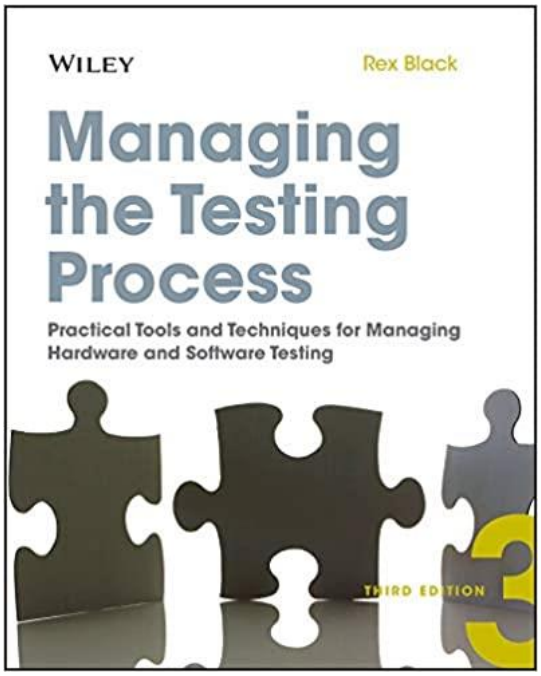


Рисунок 1.4 – «Управління процесом тестування: практичні засоби та технології для управління апаратним та програмним забезпеченням», Рекс Блек

1.3 Постановка мети та завдань дослідження

Метою атестаційної роботи є дослідження технологій створення бібліотек для збереження тест-кейсів для тестування мобільних додатків та створення рекомендацій щодо функцій, які повинна мати програма-бібліотека.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналіз стану проблеми тестування мобільних додатків;

– провести аналітичний огляд літератури за темою атестаційної роботи

– провести порівняльний аналіз існуючих програм-бібліотек для збереження тест-кейсів;

– провести аналіз стану проблеми процесу створення та збереження тест-кейсів;

– сформувати критерії для бібліотеки тест-кейсів;

– розробити рекомендацій щодо програм-бібліотек для збереження тест-кейсів.

Об'єкт дослідження – це те, що буде взято дослідником для вивчення і дослідження. Зазвичай назва об'єкта дослідження міститься у відповіді на питання: що розглядається?

Предмет дослідження – це особлива проблема, окремі сторони об'єкта, його властивості та особливості, які, не виходячи за рамки досліджуваного об'єкта, будуть досліджені в роботі (проекті). Зазвичай назва предмета дослідження міститься у відповіді на питання: що вивчається?

У дослідницькій роботі об'єкт і предмет дослідження, мета, завдання та методи дослідження формулюються і записуються у введенні проекту.

Об’єктом дослідження даної роботи є процес тестування мобільних додатків. Предмет дослідження – інструменти для тестування мобільних додатків.

Таким чином, в ході дослідження буде проводитися огляд існуючих програм-бібліотек для збереження тест-кейсів, виділення основних критеріїв, які повинні бути присутні у бібліотеках, проводитися порівняльний аналіз даних бібліотек, а також буде розроблено набір рекомендацій для створення бібліотек для збереження тест-кейсів.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Концепція тестування

Тестування програмного забезпечення – це процес оцінки функціональних можливостей програмного забезпечення з метою з'ясувати, чи відповідає розроблене програмне забезпечення зазначеним вимогам чи ні, та виявити дефекти, щоб забезпечити безвідмовність продукту для отримання якісного продукту.

Ручне (мануальне) тестування – це процес ручного тестування програмного забезпечення, щоб дізнатись більше про нього, знайти, що є, а що не працює. Зазвичай це включає перевірку всіх функцій, зазначених у документах з вимогами, але часто також включає тестувальників, які пробують програмне забезпечення з урахуванням точки зору їхнього кінцевого користувача. Плани ручного тестування варіюються від повністю написаних сценаріїв тестів, надаючи тестувальникам детальні кроки та очікувані результати, до керівництв високого рівня, які спрямовують на пошукові сесії.

Автоматизоване тестування – це процес тестування програмного забезпечення за допомогою інструменту автоматизації для виявлення дефектів. У цьому процесі тестувальники виконують тестові сценарії та автоматично генерують результати тестування за допомогою засобів автоматизації. Деякі з відомих засобів автоматичного тестування для функціонального тестування – QTP/UFT та Selenium.

Тестування забезпечує виявлення (констатацію наявності) фактів розбіжностей з вимогами (помилок). Як правило, на фазі тестування здійснюється і виправлення ідентифікованих помилок, що включає локалізацію помилок, знаходження причин помилок і відповідне коригування тестованої програми (Application (AUT) або Implementation Under Testing (IUT)). Якщо

програма не містить синтаксичних помилок (пройшла трансляцію) і може бути виконана на комп'ютері, вона обов'язково обчислює будь-яку функцію, яка здійснює відображення вхідних даних у вихідні. Отже, судити про правильність чи неправильність результатів виконання програми можна, тільки порівнюючи специфікацію бажаної функції з результатами її обчислення, що і здійснюється в процесі тестування.

Тестування програмного забезпечення є невід’ємною частиною забезпечення якості програмного забезпечення. Тестування економить час, зусилля та витрати, і це дозволяє оптимально виробляти якісний кінцевий продукт. Існує безліч програмних інструментів та платформ, які розробники можуть використовувати для автоматизації та організації тестування для досягнення цілей тестування програмного забезпечення.

Тест-кейс – це задокументований набір передумов, процедур та постумов (очікуваних результатів), які використовує тестувальник, щоб визначити, чи відповідає випробовувана система вимогам чи працює належним чином. Тестовий кейс може мати один або кілька тестових сценаріїв, а колекція тестових кейсів називається набором тестів.

Хороший тестовий сценарій повинен включати адекватну інформацію про те, які дії повинні відбутися та які результати тестувальники повинні очікувати. Якомога інформації також потрібно записати для подальшого використання.

Полями, які потрібно включити в тест-кейс, є:

– ідентифікатор: унікальний ідентифікаційний номер для тесту;

– заголовок: короткий опис тесту;

– пріоритет: це допомагає визначити, наскільки важливим є тест, щоб переконатися, що програмне забезпечення працює належним чином. Перш ніж випустити програмне забезпечення, необхідно пройти тестові випадки з високим пріоритетом. Низький пріоритет означає, що проект можна розгорнути, навіть якщо тест не пройде;

– виконавець: особа, яка проходить тест;

– дата виконання: дата запуску тесту;

– передумови: певні умови, які необхідно виконати перед запуском тесту;

– вхідні дані: фактичні значення, які нам потрібно ввести перед виконанням тесту;

– крок тесту: серія дій, які повинен виконати тестувальник, щоб переконатися в бажаних результатах;

– очікуваний результат: результати, які розробники очікують побачити з урахуванням наданих даних;

– фактичний результат: фактичний результат тесту;

– статус: пройшов тест (відповідав очікуванням) чи не пройшов (не відповідав очікуванням).

Для спрощення тестування та процесів було вирішено додавати усі тест-кейси до бібліотеки.

Бібліотека містить загальні тестові кейси, які можна використовувати повторно в різних проектах. Наприклад, функціональність логіну майже однакова у всіх проектах, а також тестування логіну теж ідентичне. Таким чином, ви можете створити кілька тестових кейсів для функції «користувач для входу» та зберегти їх у бібліотеці тестових кейсів.

2.2 Види тестування

Загалом, існує два основних види тестування – функціональне та не функціональне (структурне).

Функціональне тестування – це тип тестування програмного забезпечення, який перевіряє програмну систему на відповідність функціональним вимогам чи специфікаціям. Мета функціональних тестів – перевірити кожну функцію

програмного додатку, надаючи відповідні вхідні дані та перевіряючи вихідні дані відповідно до функціональних вимог.

Функціональне тестування в основному передбачає тестування методом чорного ящику, і його не турбує вихідний код програми. Це тестування перевіряє користувальницький інтерфейс, API, базу даних, безпеку, взаємодію клієнт/сервер та інші функції тестованої програми. Тестування може проводитися як вручну, так і за допомогою автоматизації.

Функціональне тестування вимагає ідентифікації тестових даних, обчислення очікуваних результатів, проходження тестових кейсів та порівняння фактичних та очікуваних результатів, щоб на практиці визначити, що робить програма і чи робить це так, як планується.

Нефункціональне тестування перевіряє, як працюють додатки, перевіряючи такі речі, як продуктивність, доступність та UX. Нефункціональне тестування є також критичним, як і функціональне тестування. І оскільки командам потрібно проводити поєднання різних видів тестування, тестувальникам потрібно робити обидва.

Нефункціональне тестування робить додатки більш корисними та надійнішими. На жаль, його часто пропускають, намагаючись дотриматися термінів випуску програми.

Якщо не враховувати функціональне тестування, дефекти продуктивності та UX можуть призвести до поганого досвіду та спричинити шкоду бренду [5]. Гірше того, програми можуть привести до збою з напливом користувачів. Дефекти доступності можуть спричинити штрафні санкції. І їх безпека може бути під загрозою.

Єдина мета цього типу тестування – забезпечити тестування нефункціональних аспектів програми та належну роботу програми в користувацькому контексті. Мета – охопити тестування всіх характеристик

програми, які допомагають забезпечити функції, що відповідають очікуванням бізнесу.

Оскільки вичерпне функціональне тестування неможливе, мова може йти про розробки методів, що дозволяють підбирати тести не «наосліп», а з великою імовірністю виявлення помилок в програмі.

При структурному тестуванні програма розглядається як «білий ящик» (тобто її код чи текст відкритий для користування). Відбувається перевірка логіки програми. Повним тестуванням в цьому випадку буде таке, яке приведе до перебору всіх можливих шляхів на графі передач управління програми (її керуючому графові). Навіть для середніх по складності програм числом таких шляхів може досягати десятків тисяч.

Таким чином, ні структурне, ні функціональне тестування не може бути вичерпним.

Тестування повинне включати перевірку всіх гілок програми і має включати мінімальний набір прикладів. При тестуванні не можна передбачити усі можливі ситуації, але треба охопити якомога більше випадків чи сценаріїв.

Намагаючись вирішити, як розподілити зусилля щодо тестування та скільки тестів написати для кожного типу, важливо врахувати:

– покриття тесту: скільки коду має тест, який перевіряє його поведінку;

– час виконання тесту: скільки часу потрібно для запуску тесту та повернення результату;

– результат про тестування: Наскільки корисним для тестованої системи є той результат, який отримуємо від тесту;

– технічне обслуговування: Як часто потрібно буде оновлювати тест.

Одиничний тест пропонує найменшу кількість охоплення, оскільки він зосереджений на одному методі. Для отримання високого охоплення потрібно багато модульних тестів. Системні тести або UAT охоплюють велику область коду і, таким чином, забезпечують більш високе охоплення

Юніт-тест дуже швидкий, оскільки вимагає мінімальних налаштувань, щоб зосередитись на одному методі. Виконання тесту відбувається в мілісекундах. Системні тести або UAT охоплюють значну частину коду і вимагають кроків налаштування, тому їх запуск триває довше, зазвичай від декількох секунд до хвилин.

Одиничний тест забезпечує конкретний зворотний зв’язок щодо того, що не вдалося, оскільки тест має вузьку направленість. Тож для виправлення коду потрібно мінімальне усунення несправностей. Системні або UAT тести мають широкий обсяг [6]. Невдалий тест повідомляє про те, що десь є помилка, але потрібне подальше усунення несправностей, щоб визначити, яка точка інтеграції, компонент або блок потребує виправлення.

Оскільки модульний тест є вузьким за обсягом, він потребує оновлення лише тоді, коли відразу змінюється код, який охоплює тест. Будь-які інші зміни не вплинуть на тест. Система або UAT мають широкий обсяг, і будь-яка зміна багатьох частин, які виконує тест, призведе до необхідності оновлення тестів.

Для того, щоб мати тести, які можна підтримувати, що забезпечують швидкий зворотний зв'язок та високий рівень охоплення, нам потрібно написати багато модульних тестів, за якими слідує менша кількість інтеграційних тестів, за ними менша кількість компонентних тестів тощо.

Це веде до піраміди тестування. Тестова піраміда (рис. 2.1) – це спосіб зрозуміти те, як різні види тестів повинні використовуватися для створення збалансованої бібліотеки. Її основний момент полягає в тому, що у команди повинно бути набагато більше UnitTests низького рівня, ніж BroadStackTests високого рівня, що проходять через графічний інтерфейс.

Тестова піраміда багато зустрічається в тестових ранах системи Agile, і хоча її основне направлення є створення добре збалансованої бібліотеки тестів, про неї можна сказати набагато більше.

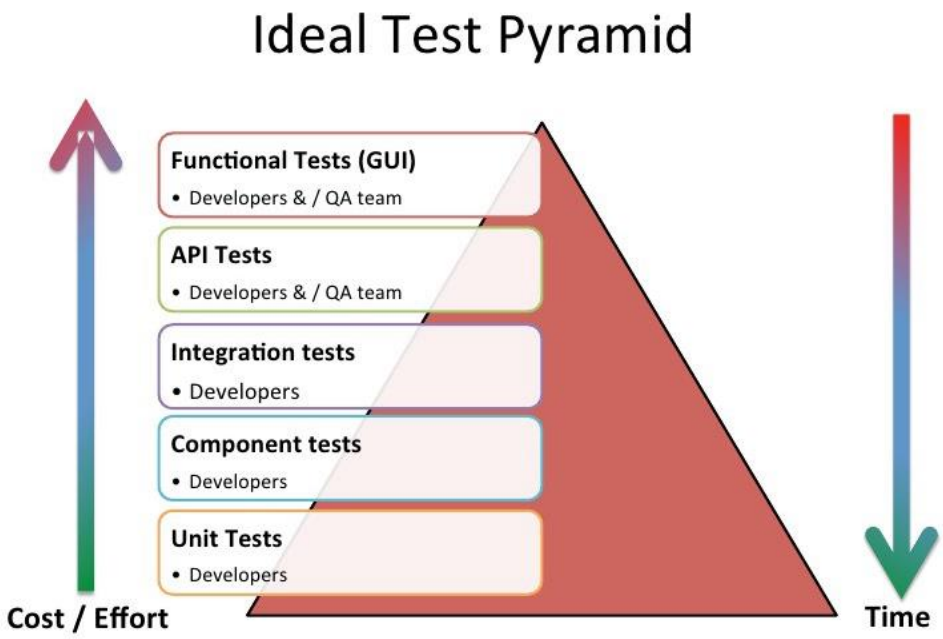


Рисунок 2.1 – Модель тестової піраміди

Типовою проблемою є те, що команди поєднують поняття наскрізних тестів, тестів інтерфейсу користувача та тестів, що стоять перед клієнтами. Це все ортогональні характеристики. Наприклад, розширений інтерфейс Javascript повинен перевіряти більшу частину своєї поведінки інтерфейсу за допомогою модульних тестів Javascript, використовуючи щось на зразок Jasmine. Складний набір бізнес-правил може містити тести, зафіксовані у формі, спрямованій на клієнта, але запускатись лише на відповідному модулі, як і модульні тести.

Тести високого рівня є другою лінією захисту. Якщо в тесті високого рівня спостерігається помилка, то вона спостерігається у функціональному коді, тож у у команди також є відсутність модульного тесту або неправильний модульний тест. Тому перед тим, як виправити помилку, виявлену тестом високого рівня, потрібно відтворити помилку за допомогою модульного тесту. Тоді модульний тест гарантує, що помилка залишається виправленою.

2.3 Життєвий цикл тестування програмного забезпечення

Життєвий цикл тестування програмного забезпечення (STLC) – це послідовність конкретних заходів, що проводяться в процесі тестування для забезпечення досягнення цілей щодо якості програмного забезпечення. STLC передбачає як перевірку, так і валідацію перевірки. На відміну від загальноприйнятої думки, тестування програмного забезпечення – це не просто одинична/ізольована діяльність. Воно складається з ряду заходів, що проводяться методологічно, щоб допомогти сертифікувати програмний продукт. STLC розшифровується як життєвий цикл тестування програмного забезпечення.

У кожній моделі життєвого циклу тестування програмного забезпечення (модель STLC) є шість основних етапів:

- аналіз вимог;
- планування тестів;
- розробка кейсів;
- налаштування тестового середовища;
- виконання тестів;
- закриття тестового циклу.

Перевірка кожного модуля програмного забезпечення або програми є необхідною для забезпечення точності вимог та кінцевого продукту. Оскільки тестування програмного забезпечення саме по собі є складним процесом, проводять його поетапно [7]. Складності можуть з’являтися, якщо тестуванню бракує організації. Складність може включати невирішені помилки, невизначені помилки регресії або в гіршому випадку модуль, який пропустив тестування, оскільки термін випуску проекту наблизився.

Кожен етап STLC має певну мету та результати. Він передбачає ініціювання, виконання та припинення циклу тестування.

Аналіз вимог. Тестувальники програмного забезпечення повинні переглядати, вивчати та аналізувати доступні технічні характеристики та вимоги. Деякі вимоги дають результати, забезпечуючи їх вхідними даними. Ці вимоги є випробувальними вимогами [8]. Команда тестування вивчає як функціональні, так і нефункціональні вимоги. Після цього вони повинні вибрати вимоги, що перевіряються. Діяльність на цьому етапі включає мозковий штурм для аналізу потреб та визначення та визначення пріоритетів вимог до тесту. Вони також включають вибір вимог як для автоматизованого, так і для ручного тестування.

Діяльність у цій фазі:

– визначення типів тестів, які потрібно виконати;

– збір деталей про пріоритети тестування та фокус;

– підготовка матриці відстеження вимог (RTM);

– визначення деталей тестового середовища, де передбачається проведення тестування;

– аналіз доцільності автоматизації (якщо потрібно).

Планування тестів. Другим етапом є планування тестування, і команда з контролю якості створює план тестування після аналізу всіх необхідних вимог. Плани окреслюють сферу та цілі після розуміння продуктової області. Потім команда аналізує пов'язані з цим ризики та визначає графіки та середовища тестування для створення стратегії [9]. Після цього керівництво допрацьовує інструменти та розподіляє ролі та обов'язки для окремих людей. Також визначається приблизний графік, за яким слід завершити тестування кожного модуля. Підводячи підсумок, два основні результати цього етапу включають документацію плану тестування та оцінку часу та зусиль.

Діяльність на цьому етапі:

– підготовка плану/документа стратегії для різних видів тестування;

– вибір інструменту для тестування;

– планування ресурсів та визначення ролей та відповідальності.

Розробка тестових кейсів. Етап розробки тестових кейсів передбачає створення, перевірку чи переробку тестових кейсів та сценаріїв тестів після того, як план тесту буде готовий [10]. Спочатку дані тесту ідентифікуються, потім створюються та переглядаються, а потім переробляються на основі передумов. Потім команда контролю якості розпочинає процес розробки тестових кейсів для окремих підрозділів.

Діяльність на цьому етапі:

– створення тестових кейсів, сценаріїв автоматизації;

– огляд базових тестових кейсів та сценаріїв;

– створення тестового рану (якщо доступне тестове середовище).

Налаштування тестового середовища. Вирішує програмні та апаратні умови, в яких тестується робочий продукт. Це один з найважливіших аспектів процесу тестування, і його можна проводити паралельно з етапом розробки тестових кейсів. Тестова група може не брати участь у цій діяльності, якщо команда розробників надає тестове середовище. Тестова група повинна провести перевірку готовності (smoke тестування) даного середовища.

Передбачено таку діяльність:

– ознайомлення з необхідною архітектурою, налаштування середовища, підготовка списку вимог до обладнання, ПЗ;

– налаштування тестового середовища та даних тесту;

– виконання смоук тестування на даному білді.

Фаза виконання тесту. Проводиться тестувальниками на основі планів тестування та підготовлених тестових кейсів. Процес складається з виконання тестового сценарію та звітування про помилки. Якщо повідомляється про помилки, продукт тестування повертається назад до команди розробників для виправлення і після виправлення буде проведено повторне тестування [11].

Заходи на цьому етапі:

– виконання тестів відповідно до плану;

– перевірка виправлення дефектів;

– відстеження дефектів до закриття.

Фаза закриття тестового циклу. Це завершення виконання тесту, яке включає декілька заходів, таких як звітування про завершення тесту, збір матриць про завершення тесту та виявлення результатів тестів. Члени групи тестування зустрічаються, обговорюють та аналізують артефакти тестування, щоб визначити стратегії, які мають бути реалізовані в майбутньому, беручи уроки з поточного тестового циклу. Ідея полягає в тому, щоб усунути слабкі місця у процесі для майбутніх тестових циклів [12].

2.4 Актуальність етапу тестування

Тестування важливо, оскільки помилки програмного забезпечення можуть бути дорогими або навіть небезпечними. Помилки програмного забезпечення потенційно можуть спричинити грошові та людські втрати, і історія наповнена такими прикладами [13].

– у квітні 2015 року термінал Bloomberg у Лондоні зазнав аварії через збій програмного забезпечення, який торкнувся понад 300 000 трейдерів на фінансових ринках. Це змусило уряд відкласти продаж боргу вартістю 3 млрд фунтів;

– автомобілі Nissan відкликали з ринку понад 1 мільйон автомобілів через збій програмного забезпечення в сенсорних детекторах подушок безпеки. Повідомлялося про дві аварії через несправність цього програмного забезпечення;

– Starbucks був змушений закрити близько 60 відсотків магазинів у США та Канаді через збій програмного забезпечення в його POS-системі. Одного разу в магазині подавали каву безкоштовно, оскільки вони не змогли обробити операцію;

– деякі сторонні роздрібні продавці Amazon побачили, що ціна їх продукції знижується до 1 п. Через збій програмного забезпечення. Їм залишились великі втрати;

– уразливість у Windows 10. Ця помилка дозволяє користувачам вирватися із пісочниць безпеки через недолік системи win32k;

– у 2015 році винищувач F-35 став жертвою програмної помилки, через що він не зміг правильно виявити цілі;

– Airbus A300 China Airlines зазнав аварії через помилку програмного забезпечення 26 квітня 1994 року, внаслідок чого 264 невинних загинули;

– у 1985 р. Канадська машина радіаційної терапії Therac-25 вийшла з ладу через помилку програмного забезпечення та доставила смертельні дози опромінення пацієнтам, внаслідок чого 3 людини загинули, а 3 інших отримали важкі травми;

– у квітні 1999 року помилка програмного забезпечення спричинила невдачу запуску військового супутника на 1,2 мільярда доларів - найдорожчої аварії в історії;

– у травні 1996 року помилка програмного забезпечення призвела до зарахування на банківські рахунки 823 клієнтів великого американського банку 920 мільйонів доларів США.

2.5 Тестування мобільних додатків

Сьогодні дуже мало бізнес-підприємств можуть подумати про функціонування без мобільного додатка. Згідно з дослідженням ринку прозорості (TMR), впровадження автоматизації тестів буде стимулюватися зростанням мобільних додатків.

Тестування мобільних додатків дозволяє підприємствам створювати програми, які масштабовані та доступні на багатьох платформах. Це процес

створення прикладного програмного забезпечення шляхом його тестування на його функціональність, зручність та послідовність. Це можна зробити за допомогою автоматизації, а також за допомогою ручного тестування. Мобільні програми ускладняються, і існує потреба у наскрізному тестуванні. Від того з’являються питання, чи додаток можна завантажити ефективно, чи він працює безперебійно та забезпечує однаковий досвід роботи на різних пристроях [14].

Використання мобільних додатків продовжує зростати, оскільки мобільні пристрої продовжують ставати все більш і більш поширеними. Згідно з дослідженнями, про які повідомляє MarketWatch, на покупки мобільних пристроїв припадає 14% покупок в Інтернеті, і це, як очікується, суттєво зросте найближчим часом. У заяві, опублікованій MarketWatch, старший директор глобальних ініціатив PayPal Анудж Наяр стверджує, що «ми спостерігали зростання мобільного зв'язку з менш ніж одного відсотка обсягу платежів у 2010 році до більш ніж 20 відсотків у 2020 році». Наявність такого суттєвого зросту повідомляє, що мобільні додатки – це сфера, яку все більше підприємств наживають.

Само собою зрозуміло, переконатися, що додаток працює належним чином, дуже важливо. Така ж обережність, яку замовник вкладає у свою концепцію продукту та розбудову свого бізнесу, має бути спрямована на тестування та контроль якості для мобільних додатків, і таке тестування не можна зробити самостійно. Використовуючи професійне тестування, можна виявити проблеми, перш ніж потенційні клієнти матимуть можливість розчаруватися у додатку, а також розробити способи їх усунення до випуску продукту чи програми.

Тестування визначає, чи можна програму успішно завантажити, виконати та взаємодіяти з підтримкою внутрішньої інфраструктури вмісту. Це життєво важливий фактор у процесі розробки, який виводить на ринок високоякісний продукт. Ще тестування допомагає гарантувати поглиблений аналіз функціональності. Тестування на реальних мобільних пристроях завжди дає

точний результат для програми, і воно може повністю гарантувати, що дана функція працює на відповідному гаджеті. Це також сприяє чудовому користувацькому досвіду.

Симулятори та емулятори не можуть вирішувати такі проблеми, як переривання телефонних дзвінків, ефект зарядного пристрою та споживання батареї. Тестування на реальних пристроях дозволяє отримати конструктивні відгуки щодо цих питань. Окрім цього, реальні пристрої забезпечують точні результати з такими функціями, як акселерометри, геолокація та push-сповіщення. Зовнішній вигляд програми та її зручність також перевіряються, включаючи особливості різних гаджетів. Тестування на реальних мобільних пристроях допомагає забезпечити зручний інтерфейс.

Мобільне тестування вимагає інвестицій спочатку, але це потрібно зробити, щоб отримати більший прибуток у майбутньому. У наш час навіть невеликі стартапи можуть дозволити собі тестування, співпрацюючи з постачальниками послуг.

2.6 Створення тест-кейсів

Тестовий кейс описує умови та змінні, за яких тестувальник перевіряє, чи працює цифровий виріб правильно за невеликі, зрозумілі етапи тестування. Це набір дій, що виконуються для перевірки певної функції або функціональності програмного додатка.

Різні тести, такі як тестування забезпечення якості, функціональне тестування та тестування в різних веб-браузерах та веб-додатках, вимагають різних методологій тестування. Ось чому система тестування повинна бути абсолютно чіткою, включаючи сценарій тестування та кейси.

Створюючи тестові сценарії, важливо дотримуватися кількох правил – незалежно від того, призначені вони для ручного тестування або засобів

автоматизації. Вони повинні бути описані точно, а також бути простими та прозорими [15]. Таким чином є можливість переконатися, що тестувальники мають кришталево чітке розуміння того, що від них очікують.

Важливо визначити послідовні та неупереджені випадки використання та збагатити їх якісними та кількісними питаннями. Не менш важливо пояснити, що саме мають робити тестувальники та які кроки потрібно зробити (також можна залишити цей пункт відкритим, залежно від обсягу тесту). Тестові кейси слід завжди писати з точки зору кінцевого користувача і не повторювати. Під час написання також важливо охоплювати всі вимоги до програмного забезпечення – те, що не записано, тестуватися не буде, специфікація є ключовою.

Найпоширенішими проблемами створення тестових сценаріїв є брак часу, ресурсів та знань експертів. Оскільки все більше і більше цифрових проектів мають справу із циклами швидкого розвитку та гнучким тестуванням, час і робоча сила стають актуальними проблемами. Не маючи часу та ресурсів, можна почати повторно використовувати старі тестові кейси або просто трохи їх адаптувати. Просто неможливо продумати кожен окремий аспект тесту, не приділяючи достатньо часу, і як наслідок, кінцевий користувач забувається або, принаймні, не знаходиться в центрі обговорення проекту.

Необхідно, щоб тестові кейси із забезпечення якості програмного забезпечення були надзвичайно детальними. Це особливо вірно при розгляді таких аспектів, як передумови, які повинні бути виконані при створенні, або кожен крок, який тестувальники виконають, та очікувані результати. Однак це можна досягти лише в тому випадку, якщо компанії мають можливість і знання щодо написання конкретних тестових кейсів.

Тест-кейс, як і будь-який елемент тестування, складається з декількох частин.

Ідентифікатор тестового кейсу. Це унікальний номер тесту в системі управління тестами або в документі. Як правило, всі сучасні системи управління тестами, такі як Jira, TestRail та Zephyr, автоматично призначають ідентифікатор

новоствореному тестовому випадку. Отже, немає можливості помилитися з цим компонентом. Але, слід зазначити, що на деяких проектах все ще використовується Excel для тестування.

Назва тестового кейсу. Назва – обов'язковий атрибут ефективного тесту. Ключові риси сильного заголовка: легкий для розуміння та лаконічний. Окрім цього, Test Case Title повинен представляти назву модуля або функціональну область, яку потрібно перевірити.

Опис тестового кейсу. Перед початком тестування слід згадати всі подробиці про те, що потрібно тестувати. А саме: дані тесту, які слід використовувати, передумови (попередні кроки), деталі середовища тесту, інструменти тестування.

Передумови (Presteps) описують різні види залежностей виконання тесту:

– будь-яке спеціальне налаштування, яке потрібно зробити;

– залежності від будь-яких інших тестових випадків – чи потрібно запускати тестування до/після якогось іншого тестування;

– залежність даних користувача – з якої сторінки користувач повинен розпочати подорож; користувач повинен увійти в систему.

Тестове середовище – це налаштування програмного та апаратного забезпечення для тестових груп для виконання тестових випадків. Іншими словами, середовище підтримує виконання тесту з налаштованим обладнанням, програмним забезпеченням та мережею.

Коли люди згадують про інструменти тестування, часто пропонують автоматизоване тестування. Але є також прості інструменти для ручного тестування. Інструменти картографування штучного розуму, такі як Xmind, менеджери скріншотів, такі як Jing. У будь-якому випадку, якщо якийсь спеціальний інструмент є обов'язковим для тестування, це повинно бути вказано в описі тестового кейсу.

Як вже було згадано раніше, тестові кроки – це шлях до очікуваного результату. Кроки в ефективному тестовому випадку добре продумані та зрозумілі. Ці два моменти є основними для розуміння того, як спланувати кроки для тестування.

Основні атрибути добре продуманих кроків:

– оптимальна кількість кроків;

– один тестовий кейс охоплює лише одну незалежну функціональність;

– відсутність перевірки різної функціональності в одному тестовому випадку;

– кроки легко виконувати;

– етапи повинні охоплювати не лише функціональний потік, а й кожну точку перевірки.

Ефективність програми після виконання вищевказаних етапів тестування відображається в очікуваному результаті. Отже, перед тим, як писати тестові сценарії, треба повністю усвідомити, яка сторінка/екран повинна з’явитись після тесту, а також будь-які оновлення, які очікуються як результат, що будуть зроблені у фонових системах або базі даних.

Один тестовий кейс охоплює одну незалежну функціональність. Ось чому було б помилкою писати тест-кейс із кількома очікуваними результатами.

Пост-умови чи коментарі не є обов’язковими компонентами тесту, але це насправді підвищує ефективність роботи тесту. Тут можна додати додаткову корисну інформацію, наприклад, знімки екрана та описи, щоб надати розробникам інформацію, необхідну для виправлення виявлених дефектів.

2.7 Інструменти для створення та збереження тестових кейсів

Інструменти управління тестами або бібліотеки – це засоби автоматизації, які допомагають керувати тестовими кейсами та підтримувати їх.

Термін «тестовий менеджмент» охоплює все, що роблять тестувальники, і для виконання цього завдання використовується найкраще та ефективне програмне забезпечення для управління тестами.

Щоденні заходи тестувальника включають:

– створення та підтримка інформації про випуск/цикл проекту/компонент;

– створення та підтримка тестових артефактів, характерних для кожного випуску/циклу, для якого маються вимоги, тестові приклади;

– встановлення простежуваності та охоплення тестових активів;

– підтримка виконання тесту;

– збір метрик/створення звітів-графіків для аналізу;

– відстеження помилок / управління дефектами.

Процес управління тестами включає сукупність завдань / заходів, які були згадані вище. Цей процес є критичним, детально орієнтованим та інструментальним для того, щоб переконатися, що всі зусилля з тестування є успішними.

Засоби управління тестами використовуються для зберігання інформації про те, як проводити тестування, планування тестувальних заходів та звітування про стан заходів із забезпечення якості. Як правило, вони використовуються для ведення та планування ручного тестування, запуску або збору даних виконання з автоматизованих тестів, управління кількома середовищами та введення інформації про знайдені дефекти. Інструменти управління тестами пропонують перспективу впорядкування процесу тестування та дозволяють швидкий доступ до аналізу даних, інструментів спільної роботи та простого спілкування між різними командами проектів. Багато інструментів управління тестами включають можливості управління вимогами для впорядкування дизайну тестових кейсів із вимог. Відстеження дефектів та проектних завдань виконується в межах однієї програми для подальшого спрощення тестування.

Список найкращих інструментів управління тестами на 2020 рік:

– Kualitee;
– TestMonitor;
– TestRail;
– Microfocus Silk Central;
– TestLink;
– qTest Manager;
– Testpad;
– Practitest;
– IBM Rational Quality Manager.

3 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА ДОСЛІДЖЕННЯ

Практична частина атестаційної роботи являє собою детальний аналіз всіх вихідних даних, під час якого необхідно виявити суть проблеми обраного предмета дослідження, запропонувати ефективні, науково обґрунтовані шляхи її вирішення, докладно аргументувати висунуту гіпотезу. На практиці вирішити поставлені у вступі завдання (задачі), визначаючи оптимальні для цього методи. Здійснюється аналіз всіх отриманих даних, використовуючи для цього різні методи дослідження.

Описується практична реалізація запропонованих математичних моделей, методів і алгоритмів. Для цього необхідно розробити план експериментальних досліджень, описати процес виконання дослідження та привести його результати.

3.1 План експериментальних досліджень

Планування експериментального дослідження – це один з найважливіших етапів організації дослідження, на якому необхідно скласти оптимальний план для втілення на практиці експерименту.

Грамотно складений план дослідження, дозволяє домогтися оптимальних значень міри відповідності методик і результатів дослідження поставленим завданням, надійності та точності в дослідженні, передбачити всі нюанси. Найчастіше, щоб скорегувати план, експериментатори проводять пробне дослідження майбутнього наукового експерименту.

Проведено дослідження одиничного випадку. Це один з видів доекспериментального плану, в якому лише один раз тестується одна група, піддана впливу. Контроль залежної та незалежної змінної повністю відсутній. Такі дослідження проводяться на перших етапах наукової діяльності для

зіставлення їх результатів зі звичайними уявленнями про реальність. Відповідно дослідження за цим планом можна вважати скоріше пошуковими, ніж причинними. На цьому ж етапі застосовано метод порівняння для отриманих результатів.

Експеримент проводився у соціальній мережі. Сторінка опитування має наступну статистику: охоплення аудиторії на момент проведення експерименту, з 24 по 30 жовтня, складає 286 чоловік, покази – 1257.

Для визначення цільової аудиторії опитуваним було запропоновано відповісти на два легеньких питання, адже саме ці запитання можуть виділити окремі групи, у яких і буде проходити дослідження. Результатами опитування виявлено (рис. 3.1), що цільовою аудиторією для даної роботи є люди від 25 до 34 років, які вже працюють (чи працювали) у галузі тестування та користувалися різними програмами-бібліотеками для збереження та використання тестових кейсів.

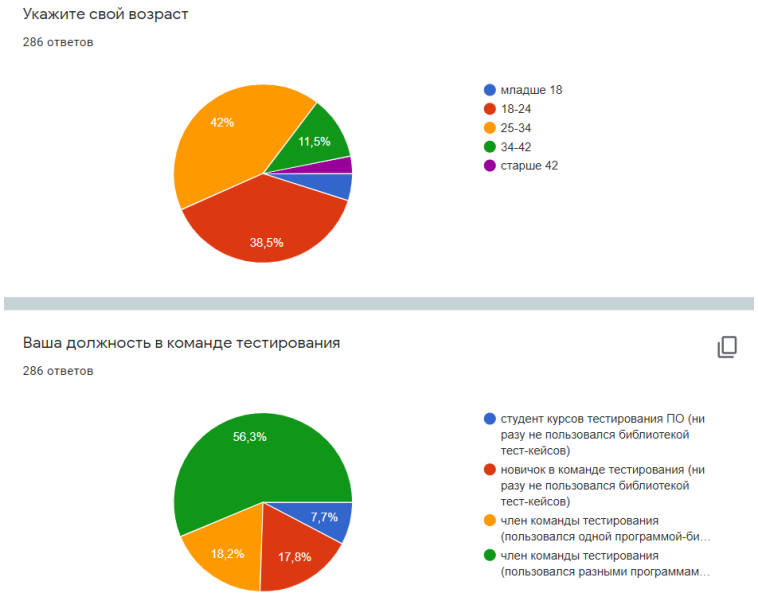


Рисунок 3.1 – Результати опитування, спрямованого на визначення цільової аудиторії

Після визначення цільової аудиторії, можна переходити до планування експерименту. План експерименту – це тактика експериментального дослідження, втілена в конкретній системі операцій планування експерименту.

Отже, план проведення експерименту наступний:

– визначення мети і завдань експерименту;
– аналіз існуючих бібліотек для збереження тест-кейсів;
– підбір досліджуваних та умов проведення експерименту;
– вибір методів дослідження;
– опис додаткових змінних, що можуть вплинути на результати;
– проведення дослідження;
– обробка та аналіз результатів дослідження;
– формування висновків.

Наукові експерименти, в цілому, можна ділити на три етапи – а саме підготовчий, організаційний та підсумковий.

3.2 Проведення дослідження

На початку експерименту було визначено мету та задачі атестаційної роботи. Оскільки, мета дослідження – це запланований результат, на досягнення якого спрямоване дослідження, то відповідно до теми роботи метою дослідження є розробка рекомендацій щодо технологій створення бібліотек для збереження тест-кейсів.

Після цього, треба проаналізувати найпоширеніші існуючи програми для управління тест-кейсами. Було розглянуто п’ять програм-бібліотек: Kualitee, TestRail, TestMonitor, Microfocus Silk Central та TestLink.

Kualitee (рис. 3.2) – найкраща програма для управління дефектами. Kualitee – це хмарний інструмент управління тестами, який підтримує як ручне, так і автоматизоване тестування. Тестувальники можуть створювати та

імпортувати тестові кейси та пов'язувати їх зі збірками та вимогами, щоб отримати повну наскрізну простежуваність.

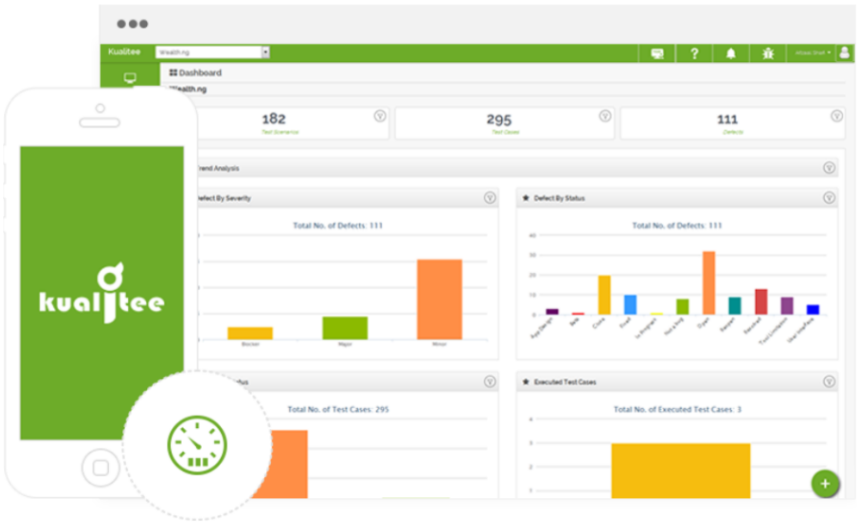


Рисунок 3.2 – Бібліотека Kualitee

Завдяки своїй функції управління дефектами, Kualitee проводить повний потік тестування від планування тесту до виконання та відстеження проблем. Це універсальний інструмент, де тестувальники, розробники та менеджери проектів можуть співпрацювати.

Інструмент дозволяє користувачам запускати власні звіти про тестування, помилки та тестування, а також включає в себе функції для планування вимог. Їх інтерактивні інформаційні панелі інтуїтивно зрозумілі та зручні.

Kualitee інтегрується з Jira, Selenium, Jenkins та Bitbucket. Інструмент також пропонує мобільний додаток для більшої зручності використання. Ціни починаються від $7 за місяць. Kualitee також пропонує безкоштовну 15-денну пробну версію.

Переваги: повний потік тестування від планування тесту до виконання та відстеження проблем, організація даних та простота використання. Інтеграція з Jira дуже корисна.

Недоліки: Деякі функції графічного інтерфейсу можна покращити. Крім того, деякі функції (мердж, створення запитів та завантаження бібліотек) повільно реагують. Налаштування набору тестових кейсів трохи трудомістке. Крім того, тестові випадки не можуть бути просто пронумеровані 1-2-3. Якщо помилки знаходяться в межах сторінки, вони перекривають кнопку «Далі»

TestRail. TestRail (рис. 3.3) – це джерело для веб-управління тестовими кейсами. Програма пропонує хмарне рішення або її можна встановити на своєму власному сервері. Завдяки цій програмі можна ефективно керувати ручними та автоматизованими тестовими кейсами, планами та прогонами та отримувати інформацію в реальному часі про тестування за допомогою інформаційних панелей, показників та звітів про діяльність.

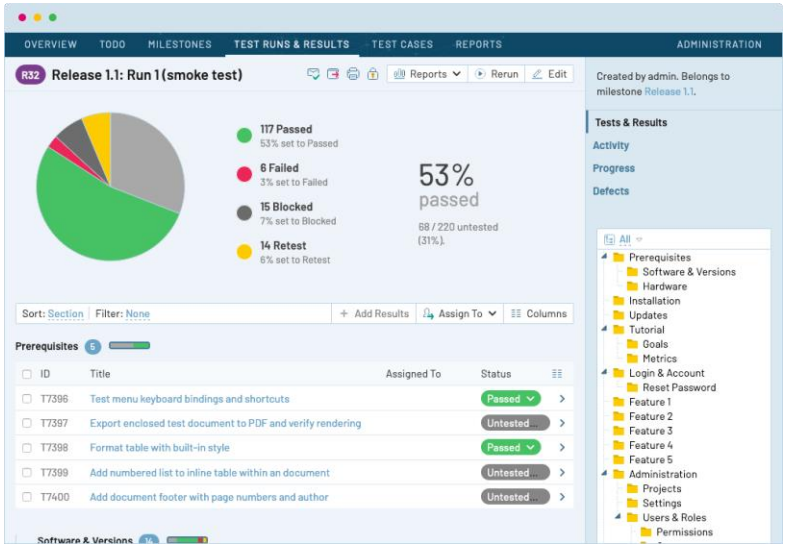


Рисунок 3.3 – бібліотека TestRail

TestRail дозволяє користувачам документувати тестові випадки за допомогою знімків екрана та очікуваних результатів. Можна використовувати гнучкі вбудовані шаблони або створювати власні шаблони. TestRail також включає деякі чудові функції підвищення продуктивності, такі як етапи, особисті списки справ та сповіщення електронною поштою.

Програму можна інтегрувати з інструментами JIRA, Bugzilla, Jenkins, TFS та багато іншого. TestRail також пропонує підтримку контейнерів Docker. Ціна TestRail починається з $34 на користувача на місяць. Вони також пропонують корпоративне видання, призначене для великих команд та критично важливих проектів.

Переваги: простий інтерфейс і функціональність. Користувачам легко зайти і використовувати інструмент, припускаючи, що інструмент відповідає робочому процесу команди Dev/QA та способу використання. Адміністратору (головному користувачу) надана можливість обмеження доступу та визначення ролей користувачів у організації. Можливості оплати – адміністратор щомісяця автоматично визначає, скільки у компанії активних користувачів, і адміністратор оплачує це, замість того, щоб закріплювати за собою встановлену кількість ліцензій – це є надзвичайно корисно, оскільки компанія проходить найми, звільнення та залучення офшорних робітників. Сценарії інтерфейсу користувача, де можна налаштувати шаблон зовнішнього виду, додаючи або видаляючи функції та функціональність.

Недоліки: неможливо відновити видалені тест-кейси. Також неможливий масовий (балковий) варіант позначення тестових випадків як протестовані (Pass/Failed/Blocked). Немає детального перегляду однієї сторінки на всіх тестових випадках. Не вдається скопіювати тестові кейси між проектами. Кожного разу, коли користувач закриває набір тест кейсів (тест ран), ідентифікаційні номери тестових випадків мають величезну область значень (наприклад ідентифікаційний номер тестового кейсу, який було створено сьогодні перевищував 11 мільйонів, хоча бібліотеці менше 50 000 кейсів). Спосіб додавання та видалення тестів із циклу – важко зрозуміти, які саме тести користувач додає та видаляє.

TestMonitor (рис. 3.4) – це інструмент управління тестами для кожної організації. Незалежно від того, чи впроваджує організація корпоративне

програмне забезпечення, створює якісний додаток, чи їй просто потрібна якісна допомога або допомога для тестового проекту, TestMonitor надає інформацію.

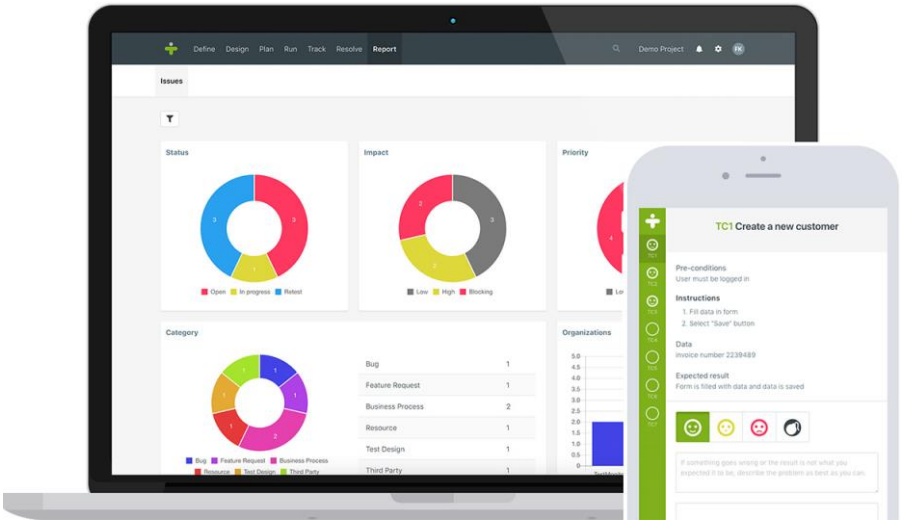


Рисунок 3.4 – бібліотека TestMonitor

TestMonitor пропонує вдосконалений дизайн тестових кейсів, який може підтримувати тисячі справ. Надійні інструменти планування заходів мають багатотестерні пробіги та клонування.

Ще однією чудовою функціональністю є комплексне відстеження результатів та інтелектуальне звітування з безліччю опцій фільтрації та візуалізації. Комплексне управління проблемами, а також тестування на основі вимог та ризику також включено. TestMonitor має простий інтерфейс та професійну підтримку з швидким часом відгуку.

Доступні сторонні інтеграції, такі як Jira, DevOps та Slack, а ще TestMonitor включає REST API для подальшої інтеграції. Ціна TestMonitor починається з $19 за користувача на місяць.

Переваги: інтуїтивно зрозумілий інтерфейс для споживачів. Управління масою результатів тестів дуже просте, а самі незадовільні результати легко перетворити на проблеми (баги) та інтегрувати з баг-трекінговою системою.

Логічна структура надає необхідний огляд на проблеми, знайдені на етапі тестування, а загальні звіти показують кожну деталь тестів. Дає хороший погляд на те, що робити та що робиться в даний момент на проекті.

Недоліки: програма призначена для ручного тестування, а не автоматизованного тестування. Для додавання скріншотів потрібні плагіни Java. Коли Java не встановлена, функція додавання скріншотів не працює. Потрібно створювати власні сценарії з нуля – неможливо використати попередні бібліотеки чи шаблони, що пропонує програма.

Microfocus Silk Central – це програма, що найкраще підходить для відтворення багів та помилок. Silk Central дозволяє об'єднати всі тестові матеріали в один простий у використанні центр планування, відстеження, звітування та виконання. За допомогою Silk Central легко відтворити помилки та проблеми в програмному забезпеченні.

Silk Central (рис. 3.5) надає уявлення про тестування за допомогою панелі TestBook для ручного тестування та дізнається, хто над чим працював, щоб відстежувати прогрес. Тестувальники можуть ділитися коментарями та ставити запитання, публікуючи коментарі безпосередньо в TestBook під час тестування, що спрощує процес співпраці між різними членами команди тестування.

Сторонні інтеграції Silk Central включають IBM DOORS, IBM Rational ClearQuest, Bugzilla, Jira та Git. Silk Central пропонує подальшу інтеграцію через їх API. Ціни на Silk Central доступні за запитом, також компанія пропонує демонстративну безкоштовну пробну версію.

Переваги: тестові цикли, що створюються автоматично програмою, це найкращий спосіб керувати та проводити ручне тестування для перевірки будь-яких вимог до продукту, виявлення помилок та регресій для їх виправлення перед тим, як доставити будь-яку версію кінцевому користувачу. Також, корисна функція для користувачів – поєднання вимог до тесту, кейсів, планів виконання та звітів про тести. Кожен об'єкт (тестовий випадок, вимога тощо)

надає посилання для переходу користувачів до інших об’єктів одним кліком, і користувачі можуть переходити між двома об’єктами.

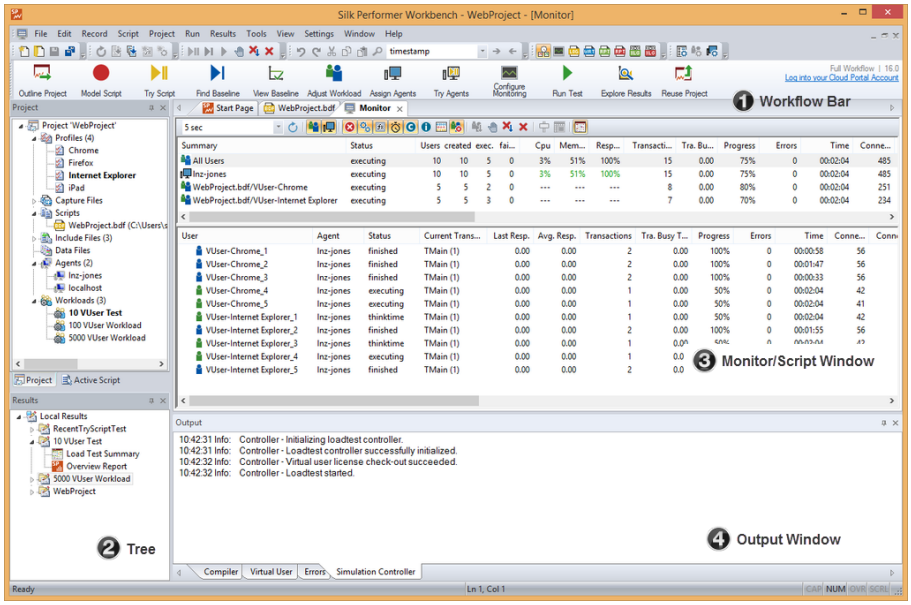


Рисунок 3.5 – бібліотека Microfocus Silk Central

Недоліки: може стати повільним, якщо проект стає величезним – можна розділити цей величезний проект на кілька, щоб запобігти цим проблемам з продуктивністю. Немає можливості легко налаштувати чи змінити тестові кейси протягом активного циклу тестування. Інтеграція зі сторонніми інструментами запрограмовані погано (плагіни для JIRA мають багато обмежень та не можуть бути використані ефективно).

TestLink. TestLink (рис. 3.6) дозволяє користувачам створювати тестові кейси та тест-плани, керувати користувачами, а також візуалізувати результати тестування та прогрес за допомогою звітування та статистики. Експорт та імпорт тестових кейсів є простим та інтуїтивно зрозумілим, а ще TestLink також пропонує простий метод призначення тестових кейсів для кількох користувачів. TestLink – це програмне забезпечення з відкритим кодом, тому користувачі можуть налаштувати його відповідно до своїх потреб.

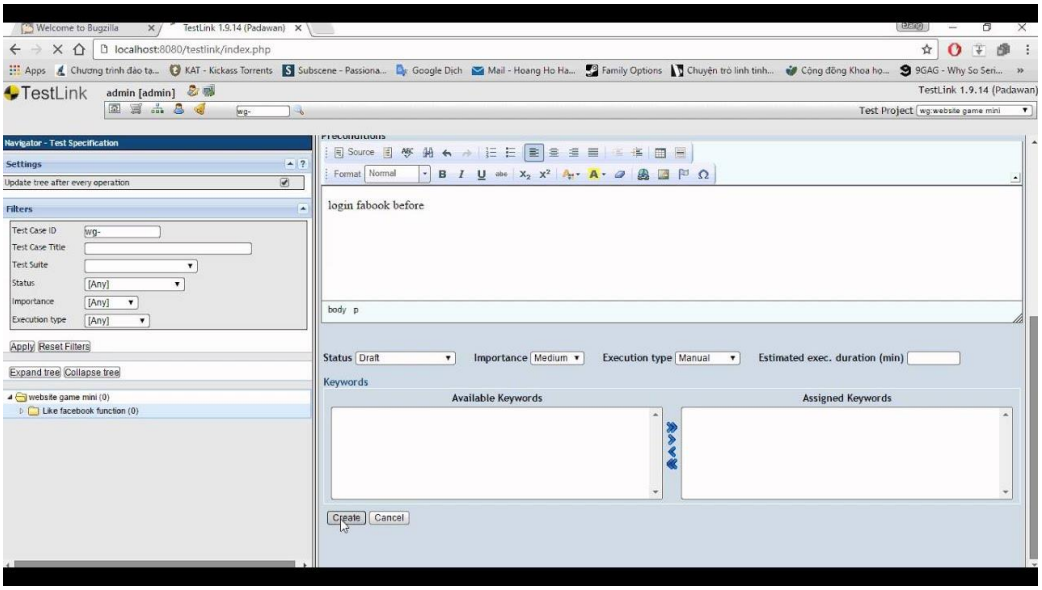


Рисунок 3.6 – бібліотека TestLink

TestLink може інтегруватися з багатьма інструментами управління дефектами, а також такими інструментами, як Selenium, Katalon Studio, TeamForge, Jira та іншими. Як інструмент з відкритим кодом, TestLink можна використовувати безкоштовно.

Переваги: підтримує декілька проектів, разом з цим легко експортувати та імпортувати тестові кейси. Є можливість легко виконати фільтрацію тестових кейсів за версією, ключовими словами, ідентифікатором тестового кейсу. Легко призначити тестові кейси для кількох користувачів. Присутнє табличне відображення результатів для кількох тестових циклів. Інформація розглядається у контексті значень, розділених комами, і надалі уточняється у вхідній частині.

Недоліки: Користувальницький інтерфейс справді старий, і користувальницький інтерфейс повинен бути більш реалістичним, а ознайомлення з інтерфейсом повинно бути корисним для майбутніх користувачів та тільки. Є ускладнення з роботою програми (повільна робота та наявність помилок, що призводять до збою програми), коли користувач прив'язує вимоги до окремого тесту та коли в кейса є багато кроків у сценарії.

Test Link не має сповіщення електронною поштою, наприклад, коли тестові випадки надсилаються на експертну перевірку, потрібно вручну надіслати посилання для рецензентів. При завантаженні на машину тестовий ран (набір тест-кейсів) самі кейси виглядають хаотично та деякі ділянки файлу є пошкоджені.

Таким чином, після аналізу програм-бібліотек, що використовуються для збереження тестових кейсів, потрібно сформувати критерії, за якими бібліотеки будуть оцінюватися експертами.

Для оцінки були виділені такі критерії:

– функціональність (чи виконує програма свої функції без збоїв?);
– user interface (UI);
– інтеграція зі сторонніми сервісами;
– автоматизація;
– up-time monitoring (постійний моніторинг) / підтримка.

Також, було обрано альтернативи для оцінювання, якими стали такі бібліотеки:

– бібліотека на основі Kualitee;
– бібліотека на основі TestRail;
– бібліотека на основі TestMonitor;
– бібліотека на основі Microfocus Silk Central;
– бібліотека на основі TestLink.

Щоб вирішити проблему прийняття рішень, потрібно скласти ієрархічну схему, у якій розподілити елементи по рівням ієрархії (рис. 3.7) та зв’язати всі критерії.

Метод аналізу ієрархій (MAI) складається в ієрархічній декомпозиції проблеми на все більш прості складові частини і подальшій обробці послідовності суджень експерта по парним порівнянням. У загальному випадку ієрархічна модель може бути представлена наступним чином: на самому

верхньому рівні знаходиться глобальна мета (фокус ієрархії), далі показані критерії, а самому нижньому рівні знаходяться альтернативи.

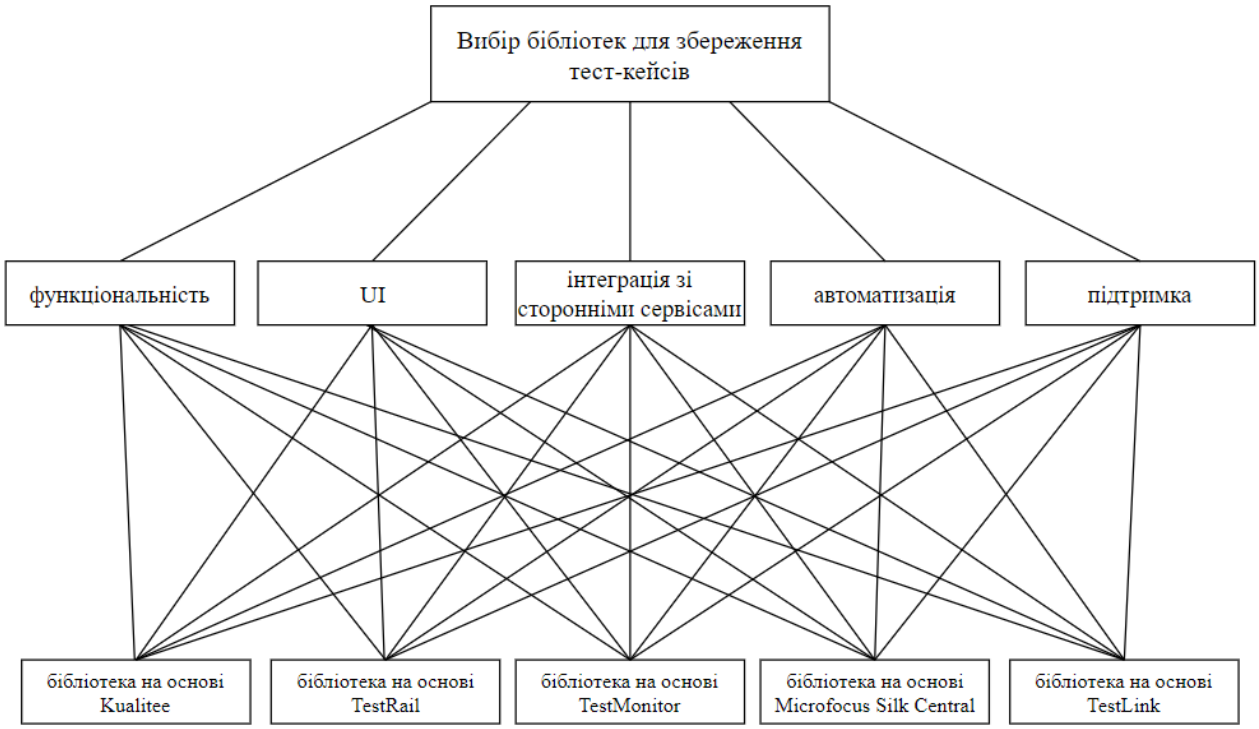


Рисунок 3.7 – ієрархічна схема вибору бібліотек для збереження тест-кейсів

Після формування ієрархії критеріїв оцінки встановлюються пріоритети (вага) критеріїв і відповідно до них проводиться оцінка альтернатив за методом лінійної згортки. В результаті визначається відносна значимість досліджуваних альтернатив для всіх критеріїв, які перебувають в ієрархії.

На підставі суджень експерта будуються матриці парних порівнянь на кожному рівні по відношенню до кожного критерію вищого рівня. До складу експертної групи увійшли люди віком від 25 до 34 років (які є цільовою групою, що було визначено раніше). Також, члени експертної групи є тестувальниками, які користувалися різними бібліотеками, що було визначено при складанні портрета цільової аудиторії. Сам експеримент проводився у закритому офісному приміщені.

Опитування експертів представляє собою заслуховування та фіксацію суджень експертів стосовно розв'язуваної проблеми. Головним в організації опитування є створення умов, при яких експертів необхідно максимально забезпечити всією необхідною інформацією. Для опитування експертів було використано анкетування, а саме дві анкети – вибір критеріїв для бібліотек, призначений для збереження тест-кейсів, а також вибір програмного забезепечення (аналогів бібліотек), яким раніше користувались (чи досі користуються) експерти.

Суть методу експертних оцінок полягає в проведенні експертами інтуїтивно-логічного аналізу проблеми з кількісною оцінкою суджень і формальною обробкою результатів, що дозволяє отримати ефективне рішення проблеми. Характерними особливостями цього методу, як наукового інструменту вирішення складних неформалізованих проблем є, по-перше, науково обґрунтована організація проведення всіх етапів експерименту, що забезпечує найбільшу ефективність роботи на кожному з етапів, і, по-друге, застосування кількісних методів як при організації експерименту, так і при оцінці суджень експертів і формальній груповій обробці результатів..

Після того як опитувані відповіли на питання дослідження необхідно обробити отримані відповіді, привести їх до зрозумілого для дослідника вигляду, придатного для аналізу і подальшого прийняття рішень. В результаті аналізу ієрархій визначається відносна значимість досліджуваних альтернатив для всіх критеріїв, які перебувають в ієрархії. Спиратися можна на загальноприйняту шкалу (табл. 3.1).

Результати експерименту можуть бути неточними завдяки впливу додаткових факторів – внутрішніх та зовнішніх. Фактори, що впливають на прийняття управлінського рішення, поділяються на об'єктивні і суб'єктивні.

Об'єктивні чинники – це чинники, які залежать від ОПР (особа, приймаюча рішення) . Уявлення про них дає зібрана інформація про стан

людини, зовнішніх факторах, що впливають на неї. Суб'єктивні чинники пов'язані з ОПР і впливають на адекватність його сприйняття об'єктивного положення системи. Зовнішнє оточення особи відрізняється високою динамічністю і невизначеністю, воно значно ускладнює умови прийняття ефективних рішень.

Таблиця 3.1 – Шкала оцінки

Значення	Важливість параметрів оцінки
1	однакова
3	незначна
5	значна
7	явна
9	абсолютна

Інформаційне середовище, в якій приймається управлінське рішення, рівень інформаційної забезпеченості, визначають об'єктивність факторів. Інформаційне середовище характеризується трьома станами: визначеністю, ризиком, невизначеністю.

Також, на результати дослідження може вплинути ефект Хоторна. Ефект Хоторна полягає в тому, що умови новизни та інтересу до експерименту, підвищена увага до самого дослідження призводять до надто позитивних результатів, що є спотворенням та відходом від реального стану речей. Відповідно до дій цього ефекту учасники дослідження, схвильовані своєю причетністю до нього, «занадто сумлінні», тож діють інакше, ніж звичайно. Найбільшою мірою проявляється цей артефакт у соціально-психологічних дослідженнях. Досліджувані особи розглядають свою участь у дослідженні як прояв уваги до себе.

Насамперед, щоб уникнути цього ефекту, експериментатор поводився спокійно та вживав заходів, щоб учасники не дізналися гіпотезу, яка перевіряється.

Перш за все, були підготовлені анкети, які кожен з експертів повинен був заповнити, опираючись на свою експертну думку. Далі, в ході дослідження було підготовлене приміщення та запрошені усі експерти. На заповнення анкет було виділено 30 хвилин загалом, тобто по 15 хвилин на кожну анкету. За експертами велось спостереження, оцінювалась їх поведінка та чесність у ході опитування, а також спостерігалось, чи були якісь відхилення у процесі. Заповнення анкети проводилось, опираючись на досвід експертів та/або їх уподобання.

Кількість відповідей експертної групи мала значний вплив на розрахування важливості параметрів оцінки. Підсумком опитування стала матриця, що відповідає кількості переваг критеріїв у парах (табл. 3.2). Елементами цієї матриці є число разів, коли в парі критеріїв, які мають найважливіше значення для експерта при виборі програми-бібліотеки для збереження тест-кейсів *j, i* критерію дали більш переважаючу оцінку ніж критерію *j*.

Таблиця 3.2 – Таблиця попарного порівняння критеріїв

	Функціонал	UI	Інтеграція	Автоматизація	Підтримка
Функціонал	1/7	1/5	1/5	1/9	1
UI	1/3	3	5	1	9
Інтеграція	1/7	5	1	1/5	5
Автоматизація	1/7	1	1/5	1/3	5
Підтримка	1	7	7	3	7
Σ	1,72	16,2	13,4	6,4	27

Нормалізація матриці. Отримана матриця перетворюється в матрицю, що містить ймовірності вибору стимулу і в парі і, j діленням оцінки з табл. 3.2 на суму (∑), отриману у кожному стовпчику таблиці та вирахувати середнє значення у кожному рядку. Отримані результати занесені у табл. 3.3.

Таблиця 3.3 – Нормалізація матриці

						Середнє значення
Функціонал	0,25	0,47	0,05	0,43	0,58	0,36
UI	0,18	0,05	0,01	0,06	0,08	0,08
Інтеграція	0,18	0,03	0,07	0,3	0,08	0,13
Автоматизація	0,33	0,16	0,37	0,19	0,19	0,25
Підтримка	0,03	0,02	0,01	0,01	0,08	0,03

Для подальшого дослідження вирішено провести попарне порівняння бібліотек, на основі яких можна в подальшому створювати свою, за функціональним критерієм (табл. 3.4).

Таблиця 3.4 – Порівняння за критерієм функціональності

	K	TM	TR	MSC	TL
K	1	1/5	1	1/7	1/9
TM	5	1	7	1/5	1/5
TR	1	1/7	1	1/5	1/7
MSC	7	5	5	1	1
TL	9	5	7	1	1
Σ	23	11,34	21	3,54	2,44

Нормалізація матриці попарного порівняння за критерієм відповідність функціоналу (табл. 3.5).

Таблиця 3.5 – Нормалізація матриці

						Середнє значення
K	0,04	0,02	0,04	0,04	0,04	0,18
TM	0,22	0,09	0,33	0,06	0,08	0,16
TR	0,04	0,01	0,04	0,06	0,05	0,4
MSC	0,3	0,44	0,23	0,28	0,4	0,33
TL	0,39	0,44	0,33	0,28	0,4	0,37

Далі попарно порівняно за критерієм UI та нормалізація матриці (табл. 3.6-3.7)

Таблиця 3.6 – Порівняння за критерієм UI

	K	TM	TR	MSC	TL
K	1	1/5	1/3	1/7	1/7
TM	5	1	3	1/5	1/7
TR	3	1/3	1	1/7	1/5
MSC	7	5	7	1	1/3
TL	7	7	5	3	1
Σ	23	13,53	16,33	4,48	1,8

Таблиця 3.7 – Нормалізація матриці

						Середнє значення
K	0,04	0,01	0,02	0,03	0,08	0,04
TM	0,21	0,07	0,18	0,04	0,08	0,12
TR	0,13	0,02	0,06	0,03	0,11	0,42
MSC	0,3	0,37	0,43	0,22	0,18	0,3
TL	0,3	0,52	0,3	0,67	0,55	0,47

Далі попарно порівняно варіанти за інтеграцією зі сторонніми сервісами – таблиця 3.8.

Таблиця 3.8 – Порівняння за інтеграцією зі сторонніми сервісами

	K	TM	TR	MSC	TL
K	1	1/5	7	1/7	1/9
TM	5	1	3	1/5	1/7
TR	1/7	1/3	1	1/7	1/7
MSC	7	5	7	1	3
TL	9	7	7	1/3	1
Σ	22,14	13,53	25	1,81	4,38

Нормалізація матриці попарного порівняння за критерієм наявності інтеграцій зі сторонніми ресурсами (табл. 3.9).

Таблиця 3.9 – Нормалізація матриці

						Середнє значення
K	0,05	0,01	0,28	0,08	0,02	0,09
TM	0,23	0,07	0,12	0,11	0,03	0,11
TR	0,01	0,02	0,04	0,08	0,03	0,17
MSC	0,32	0,37	0,28	0,55	0,68	0,44
TL	0,41	0,51	0,28	0,18	0,23	0,32

Далі попарно порівняно варіанти за критерієм наявності автоматизації – таблиця 3.10. Нормалізація матриці попарного порівняння за критерієм наявності автоматизації (табл. 3.11).

Таблиця 3.10 – Порівняння за критерієм наявності автоматизації

	K	TM	TR	MSC	TL
K	1	3	7	1/5	1/5
TM	1/3	1	3	1/7	1/9
TR	1/7	1/3	1	1/5	1/3
MSC	5	7	5	1	1/3
TL	5	9	3	3	1
Σ	10,47	20,33	19	4,54	1,96

Таблиця 3.11 – Нормалізація матриці

						Середнє значення
K	0,1	0,15	0,38	0,04	0,1	0,15
TM	0,03	0,05	0,16	0,03	0,05	0,06
TR	0,01	0,02	0,05	0,04	0,17	0,12
MSC	0,48	0,34	0,26	0,22	0,17	0,29
TL	0,48	0,44	0,16	0,7	0,51	0,46

Далі попарно порівняно варіанти за критерієм наявності підтримки (таблиця 3.12) та зроблено матрицю нормалізації за критерієм (таблиця 3.13)

Таблиця 3.12 – Порівняння за критерієм наявності підтримки

	K	TM	TR	MSC	TL
K	1	3	5	1/3	7
TM	1/3	1	3	1/5	5
TR	1/5	1/3	1	1/7	3
MSC	3	5	7	1	1/3
TL	1/7	1/5	1/3	3	1
Σ	4,67	9,53	16,33	4,67	16,33

Таблиця 3.13 – Нормалізація матриці

						Середнє значення
K	0,21	0,31	0,31	0,07	0,02	0,18
TM	0,07	0,1	0,18	0,04	0,31	0,14
TR	0,04	0,03	0,06	0,03	0,18	0,19
MSC	0,64	0,52	0,43	0,21	0,43	0,45
TL	0,03	0,02	0,02	0,64	0,06	0,15

Наступним кроком є розрахунок зведеного коефіцієнта для кожного варіанту. Зведений коефіцієнт потрібен для того, щоб визначити підсумковий результат за всіма розрахунками.

Розраховано зведений коефіцієнт для бібліотеки на основі Kualitee:

$$П(K) = 0,36\times0,18+0,08\times0,04+0,13\times0,09+0,25\times0,15+0,03\times0,18= 0,11.$$

Розраховано зведений коефіцієнт для бібліотеки на основі TestMonitor:

$$П(TM) = 0,36\times0,16+0,08\times0,12+0,13\times0,11+0,25\times0,06+0,03\times0,14=0,1.$$

Розраховано зведений коефіцієнт для бібліотеки на основі TestRail:

$$П(TR) = 0,36\times0,4+0,08\times0,42+0,13\times0,17+0,25\times0,12+0,03\times0,19=0,54.$$

Розраховано зведений коефіцієнт для бібліотеки на основі Microfocus Silk Central:

$$П(MSC) = 0,36\times0,33+0,08\times0,3+0,13\times0,44+0,25\times0,23+0,03\times0,45=0,32.$$

Розраховано зведений коефіцієнт для бібліотеки на основі TestLink:

$$П(ТС) = 0,36×0,37+0,08×0,47+0,13×0,32+0,25×0,46+0,03×0,15=0,26.$$

Отже, після проведення експерименту та необхідних розрахунків, отримано наступні результати:

Таблиця 3.14 – Результати експерименту

Kualitee	0,11	Microfocus Silk Central	0,32
TestMonitor	0,1	TestLink	0,26
TestRail	0,54		

Результати розрахунків представлені у вигляді діаграми (рис. 3.8).

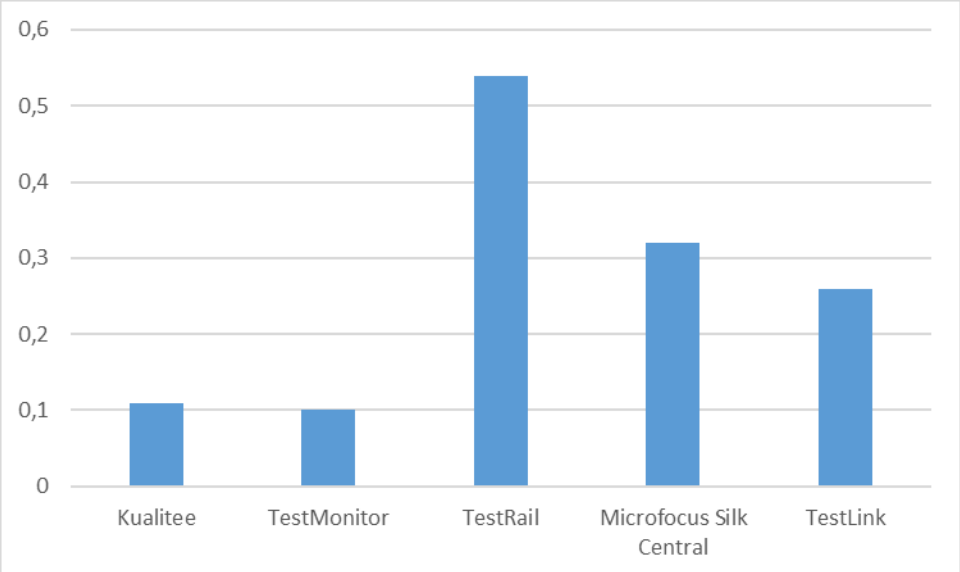


Рисунок 3.8 – Діаграма результатів

3.3 Розрахунок узгодженості експертної оцінки

Визначення узгодженості оцінок експертів необхідно для підтвердження правильності гіпотези про те, що експерти є досить точними вимірювачами, і виявлення можливих угруповань в експертній групі. Оцінка узгодженості думок експертів характеризує ступінь близькості індивідуальних думок.

Розраховано узгодженість думок експертів щодо важливості обраних критеріїв за допомогою значення коефіцієнту конкордації Кенделла:

$$W = \frac{(n+m)S}{n^2(m^3-m)}. \quad (3.1)$$

де S – сума квадратів відхилень всіх оцінок рангів кожного критерія від середнього значення; n – кількість експертів; m – кількість критеріїв.

Коефіцієнт конкордації змінюється у діапазоні 0<W<1, звідси 0 – повна неузгодженість, а 1 – повна узгодженість (табл. 3.15).

Таблиця 3.15 – Розрахунок узгодженності думок

Експерти (5)	Критерії (5)					Σ
	1	2	3	4	5	
Сума рангів	27	6,4	13,4	16,2	1,72	64,72
Відхилення від середньої суми рангів	14,06	6,54	0,46	3,26	-11,2	13,12
Квардрат відхилення	197,68	42,77	0,21	10,63	125,9	340,2

Визначення величини коефіцієнта конкордації:

$$W=10\times340,2/25\times(125-5) = 0,89.$$

Рівень узгодженості думок експертів впливає на коректність результатів виконуваного дослідження. Як показано у розрахунках, що наведено вище, у даному випадку узгодженість експертної оцінки висока.

3.4 Рекомендації щодо вибору програм-бібліотек

По-перше, потрібно відфільтрувати деякі інструменти зі списку всіх доступних інструментів на основі бюджету проекту. У великій організації, якщо це дозволяє бюджет, хороший варіант – це використання комерційного інструменту, оскільки його легко впроваджувати та підтримувати. Тоді потрібно вирішити, скільки вартості ліцензування компанія готова заплатити. Це буде залежати від часових рамок, протягом яких потрібно використовувати інструмент. Багато комерційних бібліотек також постачаються з індивідуальною ліцензією, де компанія оплачує рахунок відповідно до вибраних функцій та тривалості використання.

Обране програмне забезпечення для управління тестами повинно підвищити продуктивність – воно обов’язково повинно охоплювати багато сфер його постійного та неперервного використання.

Деталізація інформації про тестування: Інструмент повинен мати можливість надавати детальну та корисну інформацію на різних етапах процесу тестування. Звіт про тестування, сформований інструментом, повинен мати можливість вказати точний крок, коли тест не вдався, щоб це заощадило час на відстеження та тиражування проблеми. Крім того, інструмент повинен мати можливість документувати стратегії тестування, підтримувати версії тестових кейсів, журнал дефектів, посилання на історії користувачів, що працювали над кейсом, планувати виконання тесту та завантажувати відео/зображення.

Відстеження управління випуском: Інструмент повинен мати можливість надійно відстежувати та оновлювати список модулів програми, що

перевіряються та випускаються. Крім того, він повинен мати можливість надати сховище, яке представляє єдину версію історії програми для всіх зацікавлених сторін, щоб підтвердити відсутність конфлікту з вимогами.

Підтримка Agile: У наші дні більшість проектів застосовують гнучку методологію, тому обрана компанією бібліотека також повинна підтримувати Agile. Вона має включати підтримку таких функцій, як створення історій користувачів, спринти, діаграми «згорання», звіти тощо.

Бібліотека повинна пропонувати безперервну інтеграцію і повинна легко інтегруватися з інструментами автоматизації тестування та інструментами відстеження помилок, щоб зробити загальний процес тестування швидким та найбільш можливо простим.

Серед усіх можливих варіантів платформи, на якій розміститься бібліотека (її база даних, функціонал, тощо), рекомендується обирати «хмарне» сховище. Суть концепції «хмарних» технологій полягає в наданні кінцевим користувачам віддаленого динамічного доступу до послуг, обчислювальних ресурсів і додатків (включаючи операційні та інформаційні системи, серверне ПЗ і інше) через інтернет або по засобам корпоративної мережі. Розвиток сфери хостингу і необхідність масового використання громадськими ресурсами було обумовлено виникла потребою в програмному забезпеченні і цифрових послугах, якими можна було б управляти зсередини, але які були б при цьому більш економічними і ефективними за рахунок економії на масштабі. Технології «хмарних» обчислень мають величезний потенціал, тому що всі сучасні комп'ютерні продукти постійно збільшують свої вимоги до технічного оснащення комп'ютера користувача, що неминуче веде до значних витрат на апгрейд. Так що дана технологія дозволяє вирішити проблему надмірної вимогливості додатків до ресурсів кінцевого користувача [16].

Дуже корисно, якщо інструмент має повнофункціональну версію, доступну і для мобільних пристроїв. Це допомагає синхронізувати команду, яка

розпорошується по різних місцях. Це також допомагає у моніторингу 24 на 7. Обрана бібліотека повинна забезпечити хорошу підтримку в чаті, телефонних дзвінках, спілкування за допомогою електронної пошти, довідкової служби, бази знань, поширених запитань тощо [17].

Таким чином, було виділено основні критерії якості для бібліотек, що призначені для збереження тестових кейсів:

– функціональність;
– юзабіліті;
– ефективність;
– портативність;
– надійність (up-time monitoring);
– ремонтопридатність;
– ліцензування та ціноутворення.

Нижче наведені функції високого рівня, які повинна мати кожна бібліотека для зберігання тест-кейсів:

– сприяє плануванню, контролю;
– збір показників, пов’язаних з тестом;
– інформація про ведення журналу;
– моніторингові можливості;
– управління різними тестовими активами та артефактами;
– зв'язок між тестовими активами;
– підтримка звітності.

Таким чином, оглянувши п’ять найбільш популярних систем управління тестами можна виділити основні критерії та рекомендації для створення бібліотек:

– повинна бути присутня можливість редагувати тест-кейсі у інтерфейсі самої бібліотеки;

– наявність повної підтримки з боку команди розробки, а не через сторонні ресурси;

– можливість створення опитування, завдяки якому уся команда тестування може переглянути оновлення чи зміни, що внесені у той чи інший тестовий сценарій;

– без узгодження більшості членів команди тестовий сценарій неможливо оновити;

– можливість переглянути процес тестування за допомогою сторонніх інтеграцій (Jira, Git, BitBucket) для всіх членів команди;

– наявність контроль кожної дії, збереження змін у історію кейса;

– однаковий доступ до бази тестових кейсів проекту;

– можливість завантажувати зображення чи скріншоти насамперед у тестовий кейс через бібліотеку (без використання сторонніх ресурсів);

– можливість роботи з кейсами за допомогою інтерфейсу користувача (UI бібліотеки) або введенням команд до командного рядка;

– загальний (однаковий) підхід для всіх команд для роботи з документацією тестування.

Також для майбутньої бібліотеки, що буде розроблена за рекомендаціями, зроблено розрахунки методом парного порівняння з урахуванням експертної оцінки. Та розраховано зведений коефіцієнт для бібліотеки, що буде розроблена за рекомендаціями:

$$П (БР) = 0,36×1,19 + 0,08×0,6 + 0,13×0,51 + 0,25×0,37 + 0,03×0,4 = 0,35.$$

Таким чином, можна зробити висновок, що після розроблення реклами, враховуючи рекомендації розроблені у науково-дослідній роботі, показники ефективності використання бібліотек виросли.

3.5 Перевірка результатів дослідження

Задля підтвердження результатів дослідження, вирішено провести додатковий аналіз та вибрати програму-аналог, на підставі якої будуть формуватися критерії іншим методом. Методом перевірки було обрано дерево рішень.

Дерева рішень є одним з найбільш ефективних інструментів інтелектуального аналізу даних і самий корінь аналітики, які дозволяють вирішувати завдання класифікації і регресії. Вони являють собою ієрархічні деревоподібні структури, що складаються з вирішальних правил виду, відповідаючих обставинам «Якщо ..., то ...».

Правила автоматично генеруються в процесі навчання на навчальній множині і, оскільки вони сформулюються практично на природній мові, дерева рішень як аналітичні моделі більш є вербальні і інтерпретовані, ніж, наприклад, нейронні сіті.

Оскільки правила в деревах рішень виходять шляхом узагальнення безлічі окремих спостережень (навчальних прикладів), що описують предметну область, то за аналогією з відповідним методом логічного висновку їх називають індуктивними правилами, а сам процес навчання – індукцією дерев рішень.

У навчальній множині для прикладів має бути задано цільове значення, тому що дерева рішень є моделями, що будуються на основі навчання з учителем. При цьому, якщо цільова змінна дискретна (мітка класу), то модель називають деревом класифікації, а якщо безперервна, то деревом регресії.

Основні ідеї, що послужили поштовхом до появи і розвитку дерев рішень, були закладені в 1950-х роках в області досліджень моделювання людської поведінки за допомогою комп'ютерних систем.

Власне, саме дерево рішень – це спосіб подачі вирішальних правил в ієрархічній структурі, що складається з елементів двох типів – вузлів (node) і листя (leaf). У вузлах знаходяться вирішальні правила і проводиться перевірка відповідності прикладів цього правила з якого-небудь атрибуту навчальної множини.

У найпростішому випадку, в результаті перевірки, безліч прикладів, які потрапили в вузол, розбивається на дві підмножини, в одне з яких потрапляють приклади, що задовольняють правилу, а в інше – ті, що не задовольняють.

Потім до кожної підмножини знову застосовується правило і процедура рекурсивно повторюється поки не буде досягнуто деяка умова зупинки алгоритму. В результаті в останньому вузлі перевірка і розбиття не проводиться і він оголошується листом. Лист визначає рішення для кожного потрапившого в нього приклада. Для дерева класифікації – це клас, що асоціюється з вузлом, а для дерева регресії – це відповідний листу модальний інтервал цільової змінної.

Таким чином, на відміну від вузла, в листі міститься не правило, а підмножина об'єктів, які відповідають всім правилам гілки, яка закінчується даними листом. Очевидно, щоб потрапити в лист, приклад повинен відповідати всім правилам, які лежать на шляху до цього листа. Оскільки шлях в дереві до кожного листу єдиний, то й кожен приклад може потрапити тільки в один лист, що забезпечує єдність розв'язку.

Процес побудови дерев рішень полягає в послідовному, рекурсивному розбитті навчальної множини на підмножини з застосуванням вирішальних правил в вузлах. Процес розбиття триває до тих пір, поки всі вузли в кінці всіх гілок не будуть оголошені листям. Оголошення вузла листом може статися природним чином (коли він буде містити єдиний об'єкт, або об'єкти тільки одного класу), або після досягнення деякого умови зупинки, що задається користувачем (наприклад, мінімально допустиму кількість прикладів в вузлі або максимальну глибину дерева).

Алгоритми побудови дерев рішень відносять до категорії так званих жадібних алгоритмів. Жадібними називаються алгоритми, які допускають, що локально-оптимальні рішення на кожному кроці (розбиття в вузлах), призводять до оптимального підсумкового рішення. У разі дерев рішень це означає, що якщо один раз був обраний атрибут, і по ньому було вироблено розбиття на підмножини, то алгоритм не може повернутися назад і вибрати інший атрибут, який дав би краще підсумкове розбиття. Тому на етапі побудови не можна сказати, чи зможе обраний атрибут забезпечити, в кінцевому підсумку, оптимальне розбиття.

Дотримуючись усіх рекомендацій та алгоритмів побудови дерев рішень, було побудовано дерево для даного дослідження (рис. 3.9).

Аналіз дерева рішень - це графічне зображення різних альтернативних рішень, доступних для вирішення проблеми. Манера ілюстрації часто виявляється вирішальною при виборі.

Цей аналіз особливо корисний у ситуаціях, коли вважається бажаним розробити різні альтернативи рішень структурованим чином, оскільки це дасть чітке обґрунтування.

Таким чином, проаналізувавши дерево рішень, що було побудовано, можна зробити висновок, що розрахунки методом ієрархічного аналізу є правдиві, адже, двома методами було отримано однаковий результат. А саме програму, опираючись на яку, буде створено бібліотеку для збереження тест-кейсів на етапі тестування мобільних додатків.

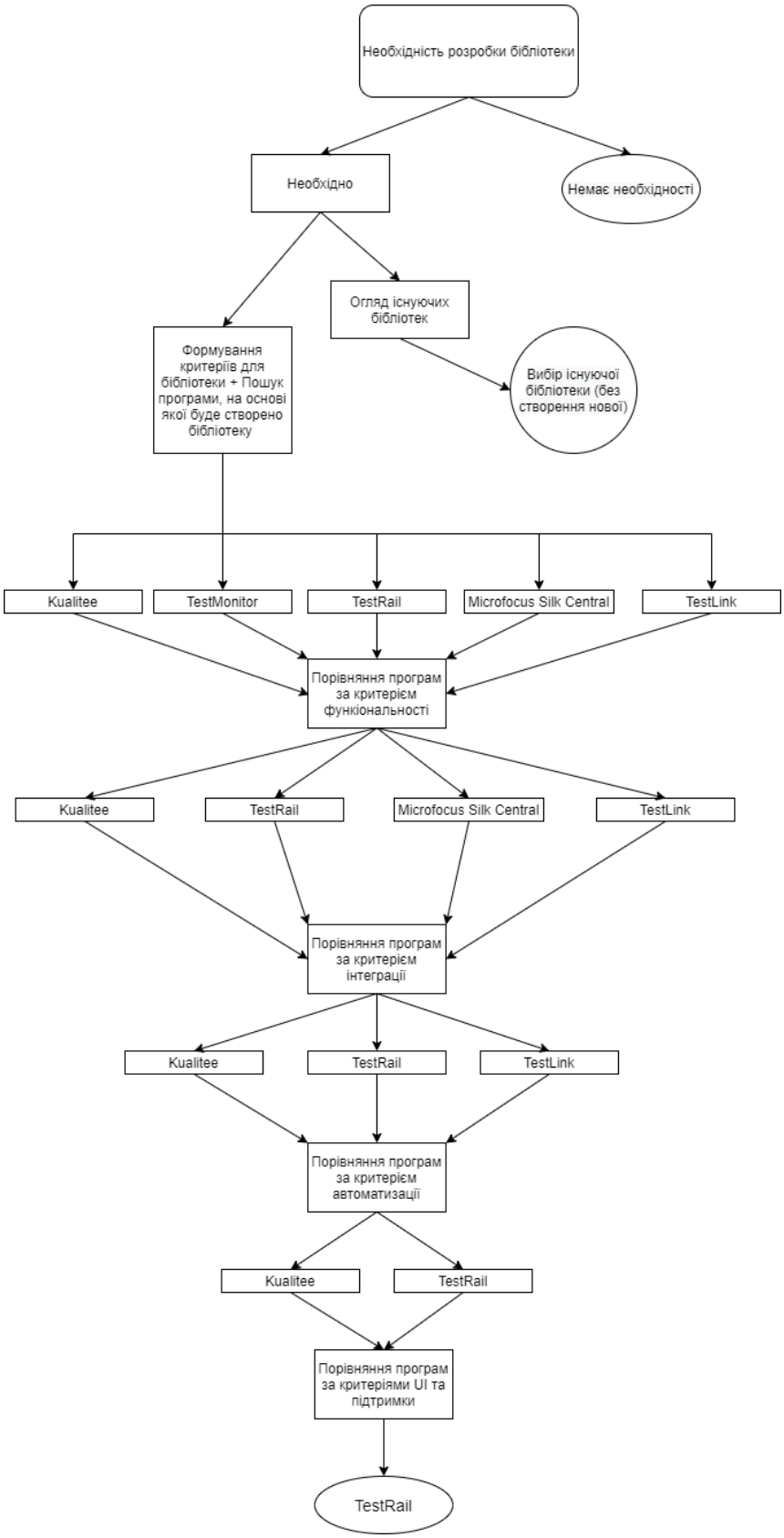


Рисунок 3.9 – Дерево рішень

4 ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ПРОЕКТУ

4.1 Характеристика наукового дослідження

У даному розділі наукового дослідження розраховуються витрати на проведення дослідження технологій створення бібліотек для збереження тест-кейсів на етапі тестування мобільних додатків. У ході обґрунтування необхідно провести розрахунок:

– трудовитрат та заробітної плати працівникам;
– одноразових витрат;
– прибутку;
– оцінки роботи;
– економічної ефективності НДР.

Метою атестаційної роботи є дослідження технологій створення бібліотек для збереження тест-кейсів для тестування мобільних додатків та створення рекомендацій щодо функцій, які повинна мати програма-бібліотека.

Об'єкт дослідження – це те, що буде взято дослідником для вивчення і дослідження. Предмет дослідження – це особлива проблема, окремі сторони об'єкта, його властивості та особливості, які, не виходячи за рамки досліджуваного об'єкта, будуть досліджені в роботі. Об’єктом дослідження даної роботи є процес тестування мобільних додатків. Предмет дослідження – інструменти для тестування мобільних додатків.

В ході дослідження проводився аналіз предметної області, визначення основних критеріїв, які повинні бути присутні у бібліотеках, огляд існуючих програм-бібліотек для збереження тест-кейсів, порівняльний аналіз даних бібліотек, а також було розроблено набір рекомендацій для створення бібліотек для збереження тест-кейсів.

4.2 Етапи виконання НДР, їх трудомісткість та заробітна плата

У процесі виконання науково-дослідної роботи (НДР) було вирішено такі задачі:

– проведено аналіз стану проблеми тестування мобільних додатків;

– проведено аналітичний огляд літератури за темою атестаційної роботи;

– проведено порівняльний аналіз існуючих програм-бібліотек для збереження тест-кейсів;

– проведено аналіз процесу створення та збереження тест-кейсів;

– сформовано критерії для бібліотеки тест-кейсів;

– розроблено рекомендації щодо програм-бібліотек для збереження тест-кейсів.

Умовно науково-дослідну роботу можна розділити на три етапи: підготовчий, основний і заключний.

На стадії виконання підготовчого етапу було проведено аналіз процесу тестування мобільних додатків, аналіз процесу створення та збереження тест-кейсів та проведено аналітичний огляд літератури за темою атестаційної роботи. Пошук інформації проходив у мережі Internet, а також було розглянуто літературні джерела.

Роботами, що було проведені на основній стадії дослідження, були:

– порівняльний аналіз існуючих програм-бібліотек для збереження тест-кейсів;

– формування критеріїв для бібліотеки тест-кейсів;

– розробка рекомендацій щодо програм-бібліотек для збереження тест-кейсів.

На заключному етапі науково-дослідної роботи було проведено аналіз результатів виконання НДР, визначення методики, складання звіту по НДР, а також захист звіту. При плануванні науково-дослідної роботи потрібно

провести розрахунок трудомісткості робіт, що є найбільш відповідальною частиною етапу. Трудові витрати часто становлять основну частину вартості науково-дослідних робіт і безпосередньо впливають на часові строки розробки.

У даній роботі біло задіяно три фахівця: керівник роботи, тестувальник та розробник. Середньомісячна заробітна плата кваліфікованого тестувальника становить 12000 грн/місяць, QA інженера – 15000 грн/місяць, керівник роботи отримує 18000 грн/місяць. Усі дані було взято з веб-ресурсу для IT-сфери – а саме усім відоме dou.ua.

Проведемо розрахунок трудовитрат і заробітної плати виконавця робіт.

Середньоденна заробітна плата виконавця робіт ($З_{ср.дн.}$) розраховується:

$$З_{ср.дн.} = \frac{З_{ср.міс.}}{n}, \quad (4.1)$$

де $З_{ср.міс.}$ – середньомісячна зарплата виконавця роботи;

n – число робочих днів у місяці, (n=22).

Середньоденна заробітна плата тестувальника складає:

$$З_{ср.дн} = \frac{12000}{22} = 545(грн).$$

Середньоденна заробітна плата керівника роботи складає:

$$З_{ср.дн} = \frac{18000}{22} = 820\ (грн).$$

Середньоденна заробітна плата QA інженера складає:

$$З_{ср.дн} = \frac{15000}{22} = 682\ (грн).$$

Етапи виконання НДР, перелік і зміст робіт, трудомісткість їх виконання, заробітна плата виконавців робіт представлені в таблиці 4.1.

Таблиця 4.1 – Етапи виконання НДР

Перелік робіт	Кіль-кість виконавців	Посада виконавця	Трудомісткість робіт, люд. днів	Середньоденна заробітна плата, грн	Сума заробітної плати, грн
1	2	3	4	5	6
1. Підготовчий етап					
1.1. Розробка та затвердження ТЗ	1	керівник роботи	1	820	820
1.2 Підготовка довідкових матеріалів та даних для виконання НДР	1	керівник роботи	1	820	820
2. Основний етап					
2.1 Постановка задачі	1	керівник роботи	1	820	820
2.2 Аналіз аналогів	1	тестуваль-ник	2	545	1090
2.3 Аналіз літератури	1	тестуваль-ник	1	545	545
2.4 Проведення порівняння аналогів програм-бібліотек	1	тестуваль-ник	2	545	1090
2.5 Формування основних критеріїв	1	QA інженер	1	682	682
2.6 Розробка рекомендацій щодо програм-бібліотек для збереження тест-кейсів	1	тестуваль-ник	1	545	545

Продовження таблиці 4.1

1	2	3	4	5	6
3. Заключний етап					
3.1 Аналіз результатів проведення роботи	2	керівник роботи, тестуваль-ник	1	1365	1365
3.2. Формування звіту, висновків та пропозицій за темою дослідження	1	керівник роботи	2	820	1640
Всього			15		9417

4.3 Розрахунок одноразових витрат на НДР

Калькуляція собівартості розраховується відповідно до існуючих нормативних актів України. До складу калькуляції входять такі статті витрат:

– матеріальні витрати;
– витрати на оплату праці;
– єдиний соціальний внесок;
– амортизація основних засобів (вартість машинного часу);
– витрати на електроенергію;
– інші витрати.

До інших витрат відносяться адміністративні витрати (водопостачання, водовідведення, опалення, освітлення) та вартість послуг зв’язку.

Матеріальні витрати визначаються витратами на матеріали та визначені їх потребою для виконання робіт, і цінами, що діють на момент складання калькуляції. Для проведення НДР потрібно: 1 механічний олівець, 3 шт. ручки та 2 шт. блокноти.

Матеріальні витрати розраховуються за формулою:

$$М = \sum_{j=1}^{n} Q_j \cdot Ц_j, \quad (4.2)$$

де М – сумарні витрати на матеріали, в тому числі малоцінні предмети, що швидко зношуються (носії, папір, канцелярське приладдя тощо), або на літературу, яка необхідна для проведення роботи, тощо;

Q_j – кількість використаних одиниць j-го виду матеріалів, j =(1÷n);

$Ц_j$ – ціна одиниці *j*-го виду матеріалів.

Розрахунок матеріальних витрат представлено в табл. 4.2.

Таблиця 4.2 – Розрахунок матеріальних витрат

Найменування	Од. вим.	Кількість, шт	Ціна, грн	Сума, грн.
Олівець механічний	шт.	1	4	4
Ручка	шт.	3	8	24
Блокнот	шт.	2	15	30
Всього				58

Витрати на оплату праці розраховуються виходячи з необхідного для виконання робіт складу й кількості працівників, а також із середньомісячної заробітної плати. Згідно проведеним раніше розрахунком, витрати на оплату праці дорівнюють 9417 грн.

Єдиний внесок на загальнодержавне соціальне страхування (ЄСВ) – консолідований страховий внесок, збір якого здійснюється в систему загальнообов'язкового державного соціального страхування в обов'язковому порядку і на регулярній основі з метою забезпечення захисту у випадках,

передбачених законодавством, прав застрахованих осіб і членів їх сімей на отримання страхових виплат (послуг) за діючими видами загальнообов'язкового державного соціального страхування.

Для об'єкта дослідження ставка єдиного соціального внеску дорівнює 22% від витрат на оплату праці, тобто розмір ЄСВ дорівнює 2071,75 грн.

При виконанні НДР застосовувалось наступне обладнання: 3 комп'ютера вартістю 25000 грн. Вищенаведене устаткування є власністю організації, тому доцільно розрахувати суму амортизаційних відрахувань на період виконання НДР.

Амортизація основних засобів розраховується за формулою:

$$АВ = \sum_{k=1}^{L} \frac{ВО_k}{ТЕ_k} \cdot Т, \qquad (4.3)$$

де АВ – сума амортизаційних відрахувань, нарахованих під час проведення науково-дослідницької роботи;

$ВО_k$ – вартість основних засобів *k*-го виду;

$ТЕ_k$ – термін експлуатації основних засобів *k*-го виду, днів;

Т – термін науково-дослідницької роботи, днів;

L – кількість видів обладнання.

Підставивши відомі значення у формулу, визначимо величину амортизаційних відрахувань:

$$АВ = \frac{25000 \times 4}{560} + \frac{25000 \times 6}{560} + \frac{25000 \times 1}{560} = 179 + 268 + 45 = 492 \text{ (грн)}.$$

Витрати на використану обладнанням електроенергію розраховуються за формулою:

$$З_е = М \cdot t \cdot Т_{кВт}, \quad (4.4)$$

де М – потужність устаткування, тобто кількість енергії, споживаної за одиницю часу (кВт/година);

t – кількість годин використання устаткування за період проведення науково-дослідницької роботи;

$Т_{кВт}$ – тариф, тобто вартість використання 1 кВт електроенергії.

Споживна потужність комп'ютера складає 0,65 кВт за годину. Тариф споживачів за першим класом напруги, тобто 35 кВт та більше), складає 1,68 грн/кВт·годин (без ПДВ). Підставивши значення у формулу, визначимо величину витрат на спожиту електроенергію:

$$З_з = 0{,}65 \times 54 \times 1{,}68 + 0{,}65 \times 24 \times 1{,}68 + 0{,}65 \times 6 \times 1{,}68 =$$
$$= 69{,}15 + 30{,}75 + 7{,}65 = 91{,}73 \text{ (грн)}.$$

До інших статей витрат відносяться такі:

– адміністративні витрати: (водопостачання, водовідведення, освітлення, опалення), які прийнято у розмірі 20% від витрат на оплату праці;

– вартість оплати послуг зв'язку.

Вартість оплати послуг зв'язку становитиме інтернет – із розрахунку 300 грн на місяць (тариф на доступ до мережі інтернет у нежитлових приміщеннях); всього 150 грн за 15 днів виконання НДР.

Витрати на ліцензію ПЗ на 15 днів становлять: Microsoft Word – 250 грн.

За час виконання НДР витрати на відрядження, аутсорсинг, інформаційні послуги не мали місця.

Результати розрахунку кошторису витрат, тобто одноразових витрат, на виконання НДР наведені в таблиці 4.3.

Таблиця 4.3 − Кошторис витрат на розробку та ціна НДР

№ з/п	Показник	Сума, грн
1	Заробітна плата	9417
2	Єдиний соціальний внесок (22,0 % від п.1)	2071,75
3	Матеріальні витрати	58
4	Амортизація основних засобів	492
5	Витрати на електроенергію	91,73
6	Інші витрати, у тому числі:	
6.1	вартість ліцензії	250
6.2	вартість послуг інтернету	150
6.3	адміністративні витрати (20% від п.1)	1883,4
7	Всього	14413,88

Таким чином, кошторис витрат на виконання даної НДР відбиває сумарні витрати за статтями п.1÷п.6 та складає 14413,88 грн.

4.4 Оцінка результатів науково-дослідної роботи

Результат – це завершальний наслідок послідовності дій, виражений якісно або кількісно. В загальному випадку оцінка результатів НДР – це визначення ефективності отриманих рішень порівняно з сучасним науково-технічним рівнем.

Відповідно до теми даної роботи можна зробити висновок про те, що у якості результату впровадження НДР є зменшення часу, що має досить велике значення для розробників, та як наслідок, користувачів.

Результат від впровадження НДР визначається за такою формулою:

$$\Delta P_j = \left| Хб_j - Хн_j \right| , \qquad (4.5)$$

де ΔP_j – покращення j-ої характеристики досліджуваного процесу за рахунок впровадження результатів НДР ($j=1,m$);

m – кількість досліджуваних характеристик;

$Хб_j$ – базове значення j-ої характеристики, тобто до впровадження результатів НДР;

$Хн_j$ – нове значення j-ої характеристики після впровадження пропонованих рішень.

У якості досліджуваної характеристики виступає оцінка експертів обраної бібліотеки для збереження тест-кейсів на етапі тестування мобільних додатків (TestRail) та оцінка після аналізу та формування рекомендацій щодо бібліотек, що в подальшому будуть створені, спираючись на ці рекомендації. Оцінка експертів визначається сумарною бальною оцінкою бібліотек з урахуванням ваги кожного критерію, яка дорівнює до впровадження 10,55 та після формування рекомендацій – 20,65. Підставивши відповідні значення у формулу (4.5), визначимо результат від впровадження НДР у чисельному вигляді:

$$\Delta P_1 = |20{,}65 - 10{,}55| = 10{,}1.$$

Далі проведено оцінку економічної ефективності отриманого результату виконаної науково-дослідної роботи.

4.5 Визначення економічної ефективності результатів НДР

Для визначення економічної ефективності результатів НДР необхідно порівняти витрати на розробку НДР з отриманими результатами.

Основним показником економічної ефективності науково-дослідної роботи є коефіцієнт «ефект-витрати», який розраховується за такою формулою:

$$K_{ев} = \frac{\Delta P_j}{B_p}, \tag{4.6}$$

де B_p – витрати (кошторисна вартість) на виконання НДР, грн.;

$K_{ев}$ – коефіцієнт «ефект-витрати», який відбиває, наскільки кожна гривня витрат НДР змінює *j*-ту характеристику досліджуваного процесу.

Підставивши раніше визначені значення до (4.6), розрахуємо чисельне значення коефіцієнту «ефект-витрати»:

$$K_{ев} = \frac{10,1}{14413,88} = 0,0007 \text{ (\%)}.$$

Таким чином, отриманий результат свідчить про те, що кожна гривня витрат на НДР змінює експертну оцінку щодо бібліотек, які використовуються для збереження тест-кейсів на етапі тестування мобільних додатків до впровадження рекомендацій, виявлених у роботі, та після впровадження на 0,0007 %. Дана науково-дослідна робота має досить високий показник економічної ефективності.

ВИСНОВКИ

В результаті проведення наукової роботи було проведено дослідження технологій створення бібліотек для збереження тест-кейсів для тестування мобільних додатків та створення рекомендацій щодо функцій, які повинна мати програма-бібліотека.

Для досягнення поставленої мети було проведено аналіз стану проблеми тестування мобільних додатків, аналітичний огляд літератури за темою атестаційної роботи, а ще проведено аналіз стану проблеми процесу створення та збереження тест-кейсів.

Також було зроблено порівняльний аналіз існуючих програм-бібліотек для збереження тест-кейсів. Це було досягнуто, використовуючи метод ієрархій, а також метод експертних оцінок та попарного порівняння обраних програм.

Завдяки цьому методу була виявлена найбільш актуальна програма, на базі якої буде будуватися бібліотека. За результатами було сформовано критерії, які повинні бути присутні у бібліотеці тест-кейсів та розроблено рекомендацій щодо програм-бібліотек для збереження тест-кейсів.

Проведено економічне обґрунтування науково дослідної роботи та розраховано економічну ефективність даного дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кеннер К. Що таке добрий тест-кейс? Нью-Йорк: STAR East: 2, 2003. 256 с.

2. Бейзер Б. Тестування методом чорного ящику. Нью-Йорк: Уайли, 1995. 189 с.

3. Кріспін Л., Грегорі Д. Agile тестування: практичний посібник для тестувальників та Agile команд. / Л. Кріспін. Аддісон-Веслі, 2012. 192 с.

4. Рекс Д. Управління процесом тестування: практичні інструменти та методи управління апаратним та програмним тестуванням. Нью-Джерсі: Уайли, 2009. 102 с.

5. Хуан Л. Дослідження процесів тестування програмного забезпечення на основі графічного інтерфейсу. Пекін, 2014. 113 с.

6. Роберт А. Writing Software Security Test Cases - Putting security test cases into your test // QA Guidelines. 2017. С. 2-5.

7. STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria. URL: https://www.guru99.com/software-testing-life-cycle.html (дата звернення 20.10.2020).

8. Куліков С. Тестування програмного забезпечення. Харків, 2014. 357 с.

9. Майерс Г., Барджетт Т., Сандер К. Мистецтво тестування програм. Лондон: Eye Book, 2011. 371 с.

10. Уіттакер Д., Арбон Д. Як тестують у Google? Вашингтон: Уайли, 2018. 254 с.

11. Клейн Т. Щоденник мисливця за багами. Лондон, 2012. 174 с.

12. Software testing theory. URL: https://www.researchgate.net/publication/236031163_The_Theory_of_Software_Testing (дата звернення 20.10.2020).

13. Software and systems engineering. URL: https://www.iso.org/standard/45142.html (дата звернення 20.10.2020).

14. Why is Mobile Application Testing Important? // The segue quality control team. 2015. URL: https://www.seguetech.com/why-mobile-application-testing-important/ (дата звернення 20.10.2020).

15. Test Case Selection and Adequacy // Pre-print for U. Toronto. 2007. URL: http://www.cs.toronto.edu/~chechik/courses18/csc410/Ch9-10AdequacyAndFunctional.pdf (дата звернення 20.10.2020).

16. Колесникова Т., Лисенко В. Хмарні технології в освітньому процесі // PMW-2016: матеріали Школи-Семінару. 2016. URL: https://openarchive.nure.ua/bitstream/document/1650/1/ShS_PMW-2016_v.pdf (дата звернення 20.10.2020).

17. О'Рейлі Н. Практика тестування програмного забезпечення: управління тестами / Ніколас О'Рейлі., 2007. – 154 с.

18. Кулішова Н.Є. Методичні вказівки з виконання магістерської атестаційної роботи для напряму підготовки 6.051501 «Видавничо-поліграфічна справа» (освітньо-кваліфікаційний рівень – магістр). Харків: ХНУРЕ, 2010. 44 с.

19. Методичні рекомендації до виконання економічної частини дипломних проектів, робіт для студентів денної та заочної форми навчання усіх спеціальностей / Л.В. Соколова та ін. Харків: ХНУРЕ, 2015. 49 с.