ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Комп'ютерних наук

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: « **Розробка програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи»**

Виконав: студент 4 курсу, групи КНД– 42
спеціальності
122 Комп'ютерні науки
(шифр і назва спеціальності)

Устюжанін-Ламалеті О.
(прізвище та ініціали)

Керівник Сєрих С.О.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормоконтроль _____
(прізвище та ініціали)

Київ – 2021

ВСТУП

Актуальність теми дослідження. Розуміння важливості процесу тестування призводить до виникнення тенденцій, спрямованих на застосування технічних прийомів перевірки якості програмного забезпечення. Найбільш важливим напрямком тут є впровадження різних систем автоматизованого тестування. Основна роль в здійсненні якісного процесу тестування належить способам організації взаємодії всіх учасників розробки і вибору правильної методології. Для автоматизації цих процесів застосовується спеціальне програмне забезпечення. Для автоматизації тестування існує безліч інструментів, засобів і рішень. Автоматизація в цілому не тільки дозволяє заощадити час на розробку, але збільшує надійність і безпеку створюваних продуктів.

Використання фреймворків для автоматизованого тестування додатків є ефективним способом для економії часу та матеріальних ресурсів на контроль за якістю додатку, тому в даний час практично жоден з веб-додатків не обходиться без використання спеціальної тестової бібліотеки.

Використання автоматизованих тестів дасть можливість: автоматично надавати розробникам звіт про стан продукту; автоматично надавати звіт про стан додатку; отримати можливість неодноразового запуску тестів; мінімізувати вплив людського фактора на процес тестування.

Таким чином, в сучасних умовах актуальними є застосування інструментів автоматизованого тестування, і розробка програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи, що дозволяє здійснити автоматизоване тестування різних веб-додатків.

Мета та завдання дослідження. Метою даної бакалаврської роботи виступає розробка програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи.

Для досягнення поставленої мети у роботі необхідно виконати низку завдань:

- розкрити поняття Веб-орієнтована інформаційна система;
- проаналізувати необхідності і можливості тестування веб-орієнтованої інформаційної системи;
- навести характеристику сучасних систем автоматизованого тестування веб-орієнтованої інформаційної системи;
- здійснити цільовий аналіз системи автоматизованого тестування веб-орієнтованої інформаційної системи;
- дослідити алгоритми системи автоматизованого тестування веб-орієнтованої інформаційної системи;
- дослідити інструментарій для розробки та удосконалення системи автоматизованого тестування веб-орієнтованої інформаційної системи;
- спроектувати систему автоматизованого тестування веб-орієнтованої інформаційної системи;
- розробити систему автоматизованого тестування веб-орієнтованої інформаційної системи;
- провести тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи;
- здійснити опис методології дослідження ефективності розробленої системи;
- навести перелік показників ефективності системи автоматизованого тестування;
- провести тестування ефективності системи автоматизованого тестування веб-орієнтованої інформаційної системи;
- проаналізувати отримані результати.

Об'єкт та предмет дослідження. Об'єктом дослідження є програмний модуль автоматизованого тестування веб-орієнтованої інформаційної системи.

Предметом виступає процес розробки програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи.

Наукова новизна і значимість бакалаврської роботи полягає в розробці програмного модулю, що дозволяє здійснювати автоматизоване тестування різних веб-додатків.

Теоретична значущість бакалаврської роботи полягає в описі методів інструментальних засобів автоматизації тестування.

Практичне значення одержаних результатів дослідження полягає в розробленому програмному модулю, який дозволить скоротити час на проведення тестування програмного забезпечення; автоматично надавати звіт про стан додатку; отримати можливість багаторазового запуску тестів; мінімізувати вплив людського фактора на процес тестування.

Структура роботи. Структуру роботи складають вступ, чотири розділи, висновки та список використаної літератури.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Розгляд поняття Веб-орієнтована інформаційна система

В даний час актуальними є питання забезпечення менеджменту інформаційних систем. Це досягається в рамках корпоративної політики, перш за все в рамках забезпечення заданих критеріїв і цілей. Їх вибір є актуальним проблемним питанням для будь-якої системи. Вирішення цього питання може бути досягнуто шляхом використання спеціальних архівів (репозиторіїв) і класифікаторів, в яких зібрані типові цілі і характеристики розглянутих класів інформаційних систем. Це дозволяє також побудувати інформаційну систему, не гіршу з наявних, і більш якісно оцінити ефективність існуючої системи [2].

Розглядаючи web-орієнтовані інформаційні системи, можна відзначити, що вони сильно звужують класи інформаційних систем і спрощують вирішення питань вибору ключових характеристик для проведення порівняльного аналізу. Але виявляється, що не дивлячись на величезне розмаїття web-систем, в Інтернеті відсутні їх класифікатори, що дозволяють допомогти розробнику провести порівняння і знайти оптимальне рішення. Питання розробки класифікаторів і класифікації web-сайтів з точки зору розробника системи розглядається як проблемне.

Аналіз відомих результатів досліджень в області класифікації Web-орієнтованих інформаційних систем показав, що багато в чому такі класифікації дуже тенденційні, мають суто комерційну орієнтацію.

Розглядаючи проблему класифікації, необхідно чітко формалізувати цілі її проведення [6]. До них можна віднести бажання проведення декомпозиції системи для порівняння систем. Таке порівняння проводиться для проектування, дослідження і для оцінки ефективності використання існуючих систем. Декомпозиція систем передбачає, що реалізується певний

рівень (глибина) розбиття, що дозволяє виділити закінчені функціональні блоки. Подання системи у вигляді таких блоків є архітектурою системи. Отже, класифікація дозволяє виявити приналежність досліджуваної системи до функціональних блоків певної архітектури.

Аналіз існуючих підходів показав, що їх зазвичай розглядають у вигляді дворівневої архітектури (рис.1.1).

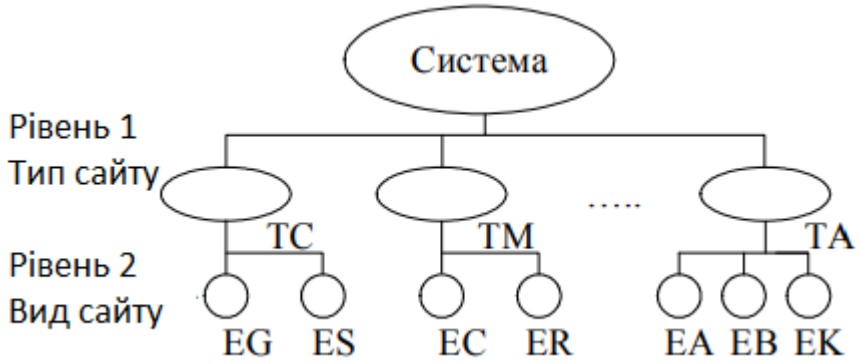

Рисунок 1.1 – Архітектура систем з двома рівнями

При цьому автори акцентують увагу на типах елементів для рівня 1 [1, 3, 4, 8] (тип сайту на рис.1.1), або рівня 2 [2, 5, 7] (вид сайту). У даній роботі пропонується визначити метод класифікації, що дозволяє сформулювати архітектуру систем з великим числом рівнів деталізації (рис.1.2).

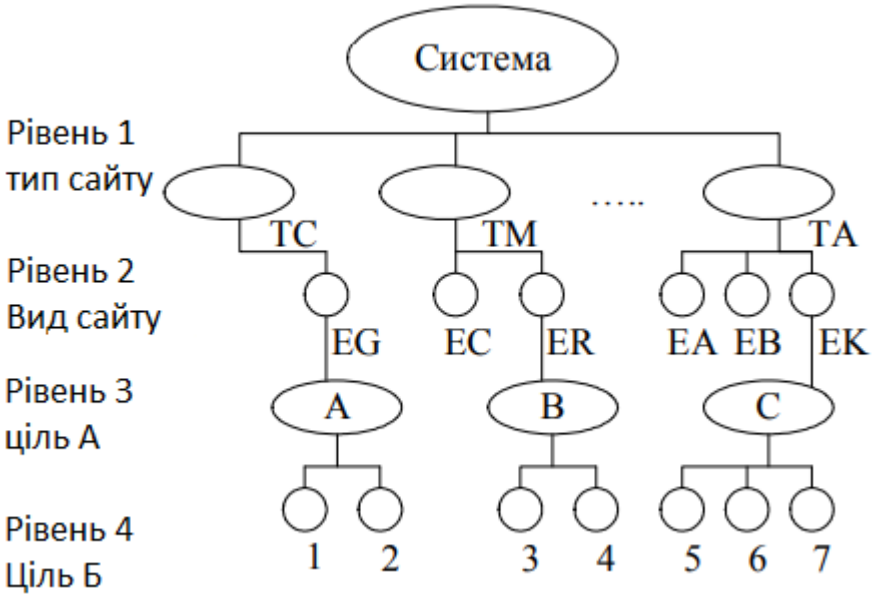

Рисунок 1.2 – Архітектура системи з великим числом рівнів

Робіт, в яких обговорювалися б питання в такому варіанті концептуального уявлення, немає.

В [1-3] була представлена спрощена класифікація типів сайтів. Наприклад, згідно з [1] маємо:

- сайт візиток;
- сайт для просування продукту або послуги;
- сайт корпоративного інтернет-представництва;
- інформаційний, сайт новин;
- інтернет магазин.

У табл.1 2 пропонуються інші приклади для типів і видів систем.

Таблиця 1.1 – Класифікатор типів систем

T_w	Тип системи
TC	Комерційні
TI	Інформаційні
TM	Інформаційно-довідкові
TP	Презентаційні
TA	Рекламні
TN	Корпоративні

$$T_w = \{TC, TI, TM, TP, TA, TN\}$$

$$E_w = \{EG, ES, ED, EW, EC, ER, EI, EP, EE, EA, EB, EK, EN, EX\}$$

У табл. 1.2 наведено приклад класифікатора видів сайтів. Ці класифікатори не претендують на повноту опису і розглядаються тільки як приклад або початковий варіант. Класифікатор може бути розширено.

Таблиця 1.2 – Класифікатор видів системи

E_w	Вид системи
EG	Електронний магазин товарів або послуг
ES	Сайт з продажу мережевих сервісів (B2C)
ED	Дилерські системи
EW	Сайти новин
EC	Каталоги продукції
ER	Каталоги ресурсів
EI	Каталог інформації
EP	Презентаційний сайт організації (необмежений час життя)
EE	Презентаційний сайт події (Обмежений час життя)
EA	Реклама торгової марки
EB	Реклама послуг
EK	Реклама товарів
EN	Сайти комерційних організацій
EX	Сайти некомерційних корпорацій

Як видно з табл. 1.1 і 1.2, не дивлячись на величезне розмаїття сайтів, їх класифікація вийшла досить стрункою і компактною. Кожен з видів сайтів можна розглядати з двох точок зору: з точки зору розробника і з точки зору користувача. Часто буває, що сайт з точки зору розробника дуже хороший, але користувачі не хочуть працювати з ним. І навпаки, є прості сайти, з якими із задоволенням працюють користувачі [8]. З нашої точки зору одна з причин такої картини полягає в тому, що розробники спочатку ставили неправильні цілі. Таким чином, для будь-якого сайту можна сформувати цілі, які переслідує власник сайту (або розробник), і цілі, які можуть бути досягнуті користувачем.

В результаті детального аналізу великої кількості сайтів нами запропоновані приклади класифікаторів для цілей власників сайтів (цілі web-систем) і цілей користувачів. Ці цілі були розглянуті для кожної з груп сайтів, з урахуванням значень за ознаками. Звичайно, можна було б розглядати ще і цілі розробників сайтів, розділяючи їх з цілями власника сайту, але ми не будемо робити цей поділ [9]. Формально для кожного сайту

можна було б не тільки виділити певні цілі, але і визначити деякі рейтингові показники ступеня їх досяжності.

Розмірковуючи далі, можна поставити питання про те, як же забезпечується досягнення зазначених цілей. Відповідь можна шукати серед функцій, які забезпечує сайт, і методів їх реалізації. Для досліджень, що проводяться в рамках реальних проектів, доцільно ці функції деталізувати. Так як кожна з них реалізована в сайті, то їй можна поставити у відповідність деякий метод. У підсумку ми бачимо, що для абсолютно різних видів сайтів використовуються одні й ті ж функціональні методи [11].

Отже, пропонується наступний набір ознак:

- тип сайту TW (визначає основне функціональне призначення);
- вид сайту EW (визначає форму реалізації сайту відповідно до деякого типу);
- цілі системи Gw (цілі, переслідувані розробниками або власниками системи);
- цілі користувача Gu (цілі, які може досягти користувач, працюючи з системою);
- функції системи F (функції, що забезпечують рішення будь-яких приватних завдань);
- методи 0, реалізують функції системи.

Отже, структура класифікації

$$\langle T_w, E_w, G_w, G_u, F, M \rangle \qquad (1.2)$$

може характеризувати систему з точки зору певної функціональності. Фактично (1.2) задає архітектуру системи. Приклад такої архітектури приведений на рис. 1.3.

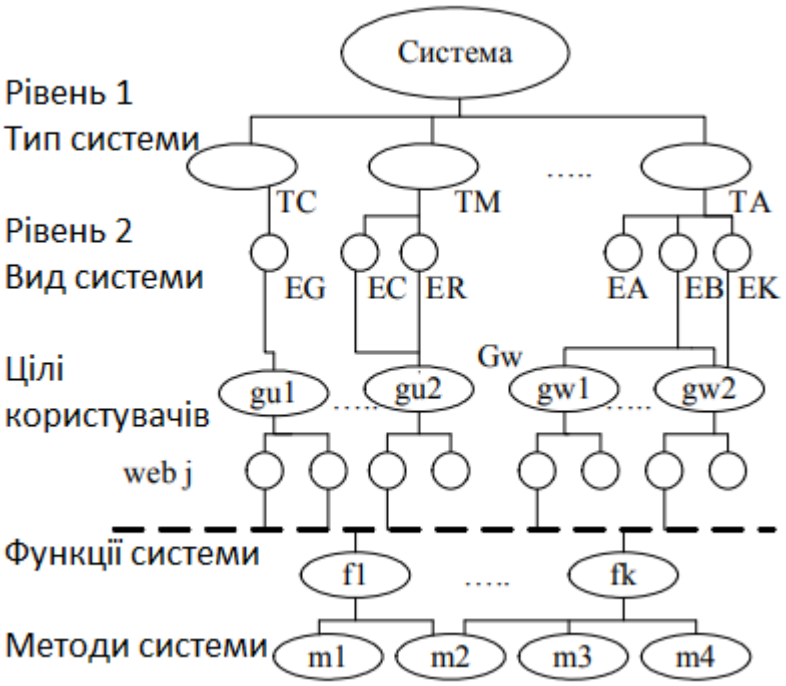

Рисунок 1.3 – Архітектура web-системи

Веб-орієнтована система являє собою клієнт-серверний додаток, в якому клієнтську частину реалізує браузер, який здійснює діалог з користувачем і відображення інформації, а серверну частину - веб-сервер і сервер додатків, які реалізують основну логіку системи. Через обмеженої функціональності клієнта, подібну реалізацію системи часто ще називають тонким клієнтом [20].

Безперечними перевагами веб-орієнтованої системи є наступні факти:

Кросплатформеність. У якості клієнта, як уже було відзначено вище, виступає веб-браузер, а це значить, що для роботи з системою необхідний тільки інтернет-браузер, який входить до складу будь-якої операційної системи. Оновлення та супровід браузера лежить на його розробнику. З огляду на те, що в даний час браузери не залежать від конкретної операційної системи користувача, веб-орієнтовані системи можна сміливо назвати кросплатформеними (або міжплатформеними) [9].

Мобільність. Працювати з системою ви можете, перебуваючи в будь-якому місці, де є Інтернет і з будь-якого пристрою, в якому є інтернет-

браузер. Таким чином, користувач (клієнт) не обмежений вимогами до апаратної частини. Можна працювати і в офісі, і вдома, і у відрядженні, як зі стаціонарного ПК, так і з ноутбука або планшета.

Низька сукупна вартість володіння. Вартість володіння веб-орієнтованою системою фактично укладена в розробці, підтримці та розвитку серверної частини (веб-сервера і сервера додатків) і володінні сервером бази даних системи. Вартість системи управління базами даних залежить від вибору технології розробки веб-додатку: використання відкритої повністю кросплатформеної або конкретної платформи для створення веб-додатків, наприклад, Microsoft ASP.NET, що в свою чергу може накласти вимоги до СУБД, але, в той же час, має і ряд своїх переваг.

Як і будь-які інші, веб-орієнтовані системи в той же час мають свої недоліки:

Для роботи з системою необхідно підключення до мережі Інтернет. Не можна не відзначити той факт, що, незважаючи на розвиток Інтернет технологій і можливості «виходу» в мережу Інтернет з використанням стільникового зв'язку, в багатьох регіонах України вартість трафіку і ширина каналу передачі даних залишають бажати кращого. Таким чином, реальна мобільність може істотно відрізнятися від заявленої з незалежних від користувача причин [30].

Не всі СЕД можуть бути замінені на веб-орієнтовані через обмеженість функціональних можливостей інтернет-браузерів. Якщо ми говоримо про просту роботу з документами, то тут труднощів не виникає, але, наприклад, робота зі специфічною документацією або необхідність відображення тривимірних моделей може накласти обмеження на використання тонкого клієнта.

Вся інформація, включаючи особисту інформацію користувача, зберігається на сервері. Цей факт накладає серйозні вимоги до захисту інформації на сервері і захисту каналів передачі даних. Не кожен клієнт готовий передавати корпоративну і особисту інформацію по мережі Інтернет.

Якій системі віддати перевагу, залежить від потреб і вимог конкретної організації. Але варто зауважити, що ринок веб-орієнтованих систем останнім часом стрімко розвивається через популярність таких аспектів як мобільність і платформна незалежність. А це означає, що йде безперервна робота над вдосконаленням веб-орієнтованих систем і усуненням їх недоліків. Цей висновок потребує більш детального аналізу необхідності і можливості тестування веб-орієнтованої інформаційної системи.

1.2 Аналіз необхідності і можливості тестування веб-орієнтованої інформаційної системи

Тестуванню додатків останнім часом приділяється дуже багато уваги. Існує величезна кількість літературних джерел [10 - 17], де наводяться рекомендації з проведення заходів з тестування та верифікації створюваного програмного забезпечення (ПЗ). У деяких навчальних закладах навіть відкриваються окремі спеціальності, які готують фахівців у цій галузі – тестерів. Однак не існує універсального підходу або досить загальних рекомендацій для проведення тестування – в кожному конкретному випадку тестер спирається на конкретне ПЗ, яке вирішує конкретну задачу.

Багаторазово збільшується складність тестування при створенні систем з розподіленою архітектурою - «база даних + сервер додатків + web-інтерфейс». Процес тестування цих додатків вимагає додаткових витрат на «Інтеграційне тестування» (тестування системного інтерфейсу) і «системне тестування» (тестування інтерфейсу користувача). Тому розробка нових підходів, що полегшують цей процес, як і раніше є актуальним завданням перед «виходом» інформаційних систем.

Загальна ідеологія тестування орієнтована на використання двох основних підходів – тестування в режимі «білого ящика» або «чорного ящика». У першому випадку процес тестування проводить розробник системи, використовуючи вихідний код. У другому – тестується вже готовий

додаток або (його модулі) з опорою тільки на виконуваний код системи. Існує проміжний підхід, який використовує ідеї перших двох – тестування «сірого ящика», де тестеру частково або повністю надано вихідний текст системи, а також її виконуваний код. Такий підхід дозволяє найбільш якісно провести всі процедури тестування ПЗ. Однак при тестуванні реальних додатків для визначення і реалізації повного набору тестових прикладів може виявитися недостатньо часу або інших ресурсів. У такій ситуації потрібно визначити, які тести найбільш важливі, а які функції аналізованого ПЗ не будуть тестуватися. Як правило, тестер в реальних умовах дотримується підходу тестування в режимі «чорного ящика».

Додатки, засновані на технологіях WWW, несуть в собі нові проблеми як для розробки, так і для тестування [14, 18, 19]. Тести для web-вузла повинні бути орієнтовані на передбачувану поведінку вузла. Необхідно оцінювати такі проблемні моменти:

- функціональні можливості;
- практичність;
- навігацію;
- форму, вміст сторінки.

Детально принципи тестування web-додатків викладені в [14]. Тестування баз даних часто є дуже важливою частиною тестування програми [10]. При тестуванні баз даних потрібні всебічні знання тестованої програми. До ключових проблем, що виникають при тестуванні баз даних, відносяться

- цілісність даних;
- достовірність даних (відповідна форма при введенні в бази даних);
- маніпуляції з даними і оновлення.

Загальне резюме: Тестування розподіленого додатка досить складне завдання. Якщо додаток використовує багаторівневу архітектуру «клієнт - сервер», то описані вище підходи повинні застосовуватися для кожного рівня. Але при цьому потрібно провести інтеграційне тестування для

перевірки системних інтерфейсів і системне тестування для перевірки зовнішніх інтерфейсів (як правило, інтерфейсів користувача або зовнішньої системи).

Зазвичай для великих програм тестування розпадається на

1) інкрементальне – тестування окремих компонент, окремих модулів, окремих підсистем, окремих частин програми;
2) інтеграційне – тестування взаємодії всіх частин програми, інтерфейсу користувача, інтерфейсу системи в цілому.

Загальною методики не існує, оскільки потрібно використовувати «особливості» додатка. Ефективного тестування неможливо досягти, не знаючи вихідного коду (структури) всіх компонент системи. Особливо завдання ускладнюється в тому випадку, якщо система розробляється окремими частинами або в систему інтегруються «чужі» модулі. Наступним кроком необхідно здійснити характеристику сучасних систем автоматизованого тестування веб-орієнтованої інформаційної системи.

1.3 Характеристика сучасних систем автоматизованого тестування веб-орієнтованої інформаційної системи

У сучасному світі існує безліч інструментів, які допомагають прискорити досягнення мети. Сучасні системи автоматизованого тестування веб-орієнтованої інформаційної системи використовуються в таких областях, як автоматичне / ручне тестування, модульне тестування, тестування продуктивності, веб, мобільне тестування та ін.

Selenium

Selenium використовує Web Driver для Chrome, щоб тестувати команди і обробляти веб-сторінки для отримання потрібних даних.

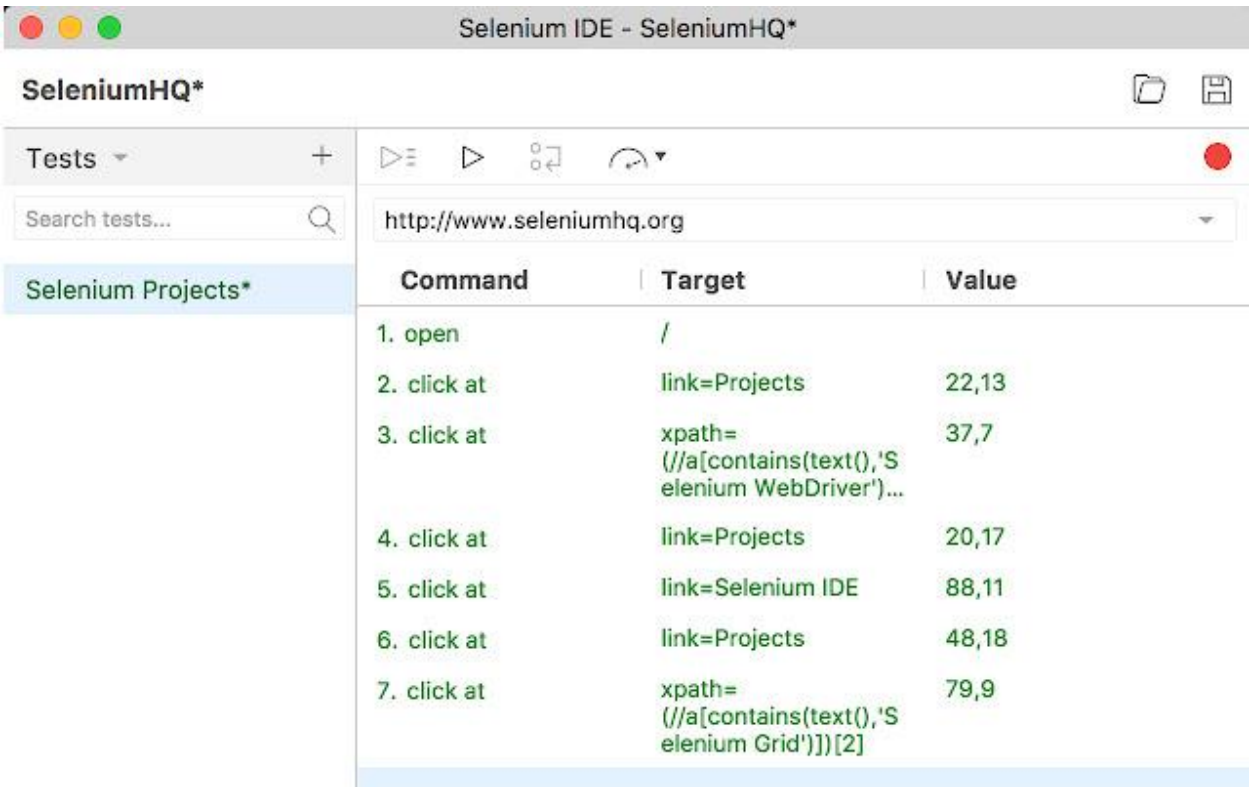

Рисунок 1.4 – Selenium

Він сумісний практично з усіма мовами програмування, пропонуючи при цьому широкий набір команд і опцій для управління.

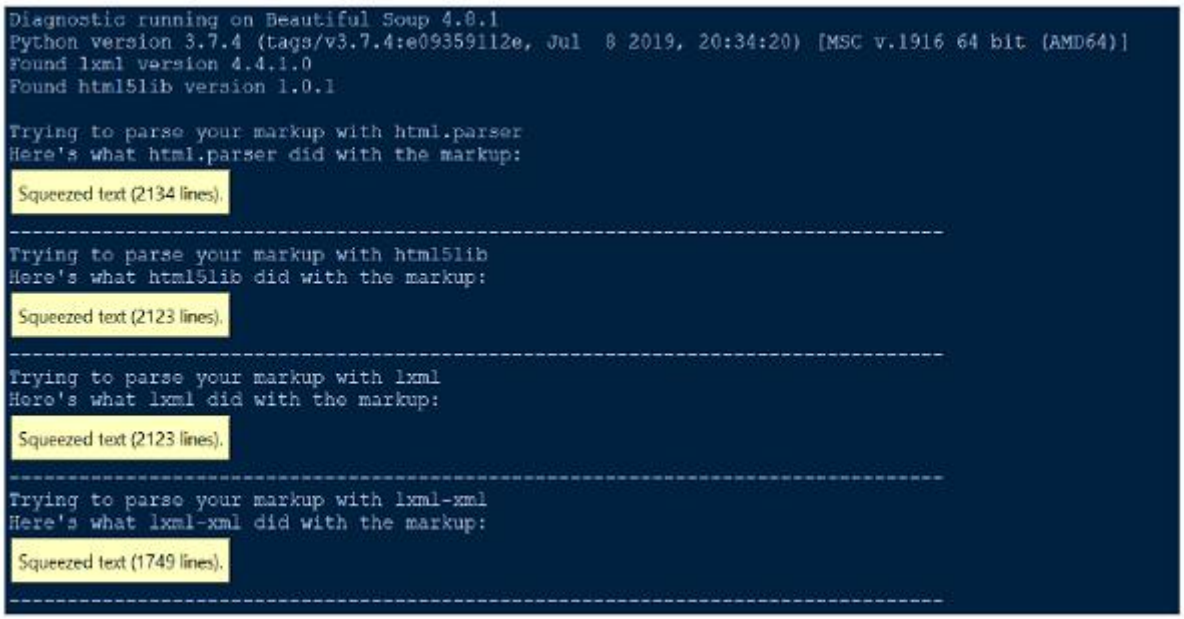

Рисунок 1.5 – Beautiful Soup

Beautiful Soup

Beautiful Soup – це бібліотека Python для отримання даних з файлів HTML і XML. Вона створює дерева зчитування даних, що дозволяють з легкістю ці дані отримувати.

Одні тільки потужні можливості цього інструменту і простота використання ставлять його в число пріоритетних інструментів.

Robotium

Robotium – це безкоштовний фреймворк для автоматизованого тестування додатків Android. Він підтримує безліч областей тестування, включаючи тестування сірого ящика UI, системне тестування і призначене для користувача приймальне тестування, як для нативних, так і для гібридних додатків Android.

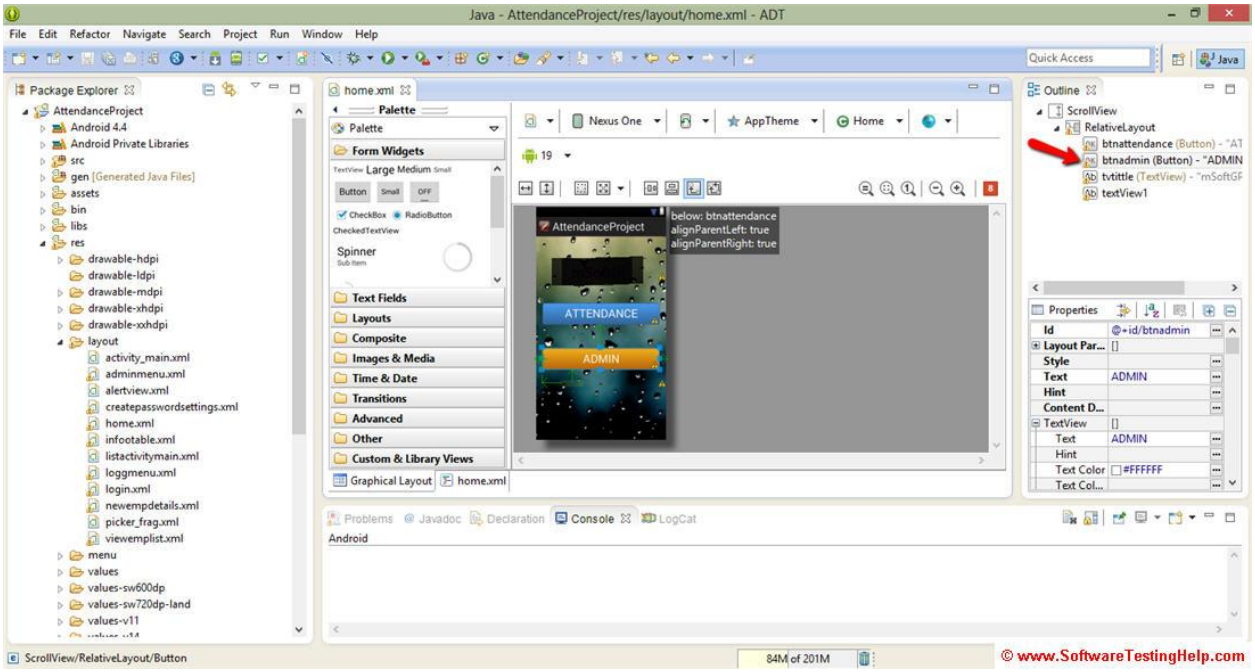

Рисунок 1.6 – Robotium

Robotium – це фреймворк для автоматичного тестування, який має повну підтримку нативних і гібридних додатків. Він полегшує написання потужних і надійних автоматичних тестів чорного ящика UI в додатках Android. З його підтримкою розробники тестових випадків можуть писати

сценарії тестування функцій, системи і призначеного для користувача приймального тестування, що охоплюють кілька видів активності.

Watir

Watir – безкоштовна бібліотека Ruby, що дозволяє виконувати автоматизоване тестування у вигляді кліків, заповнення форм та ін.

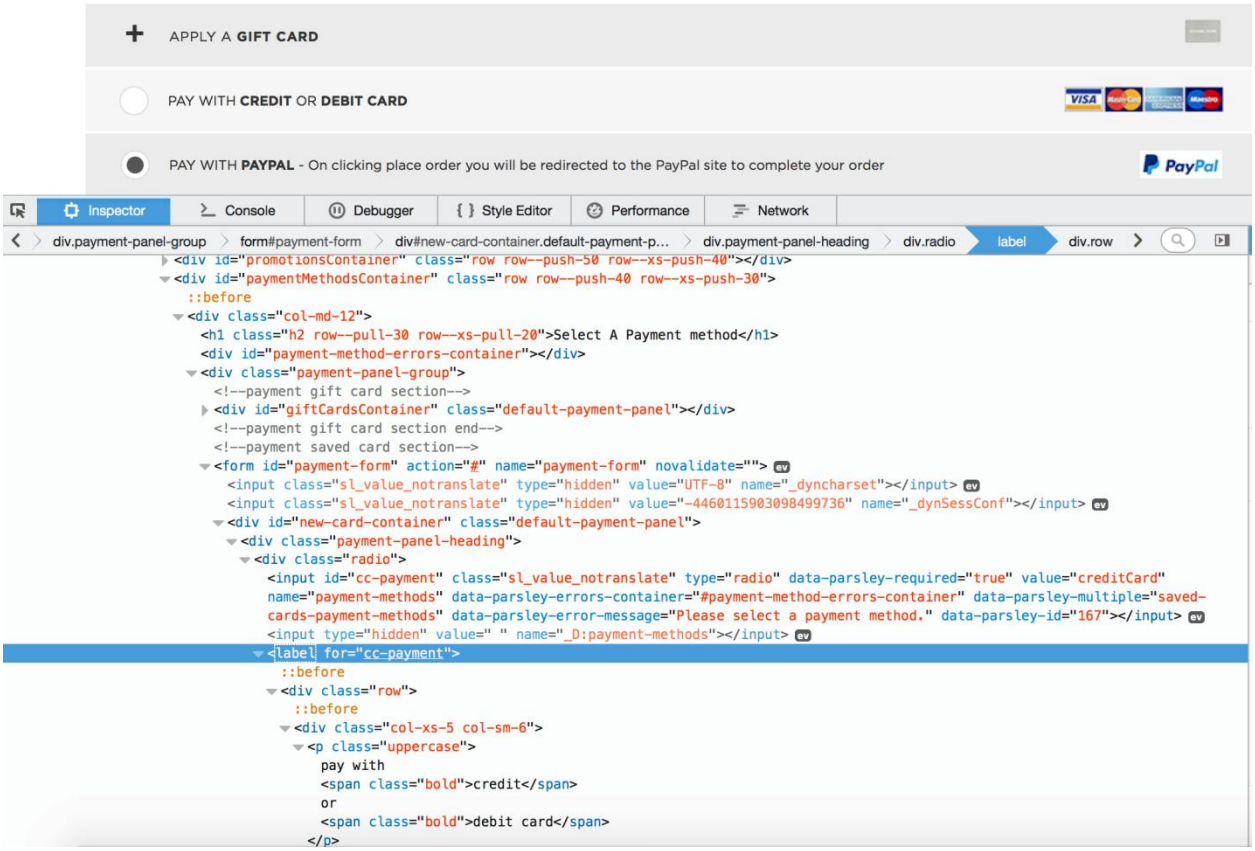

Рисунок 1.7 – Watir

Будучи відкритою бібліотекою Ruby для автоматичних тестів, Watir взаємодіє з браузером, симулюючи поведінку користувачів: відкриває посилання, заповнює форми і перевіряє текст.

Apache JMeter

Apache JMeter – це безкоштовний десктопний Java-додаток, який в основному використовується для навантажувального тестування веб-додатків. При цьому функціональне і модульне тестування він підтримує в обмеженій формі.

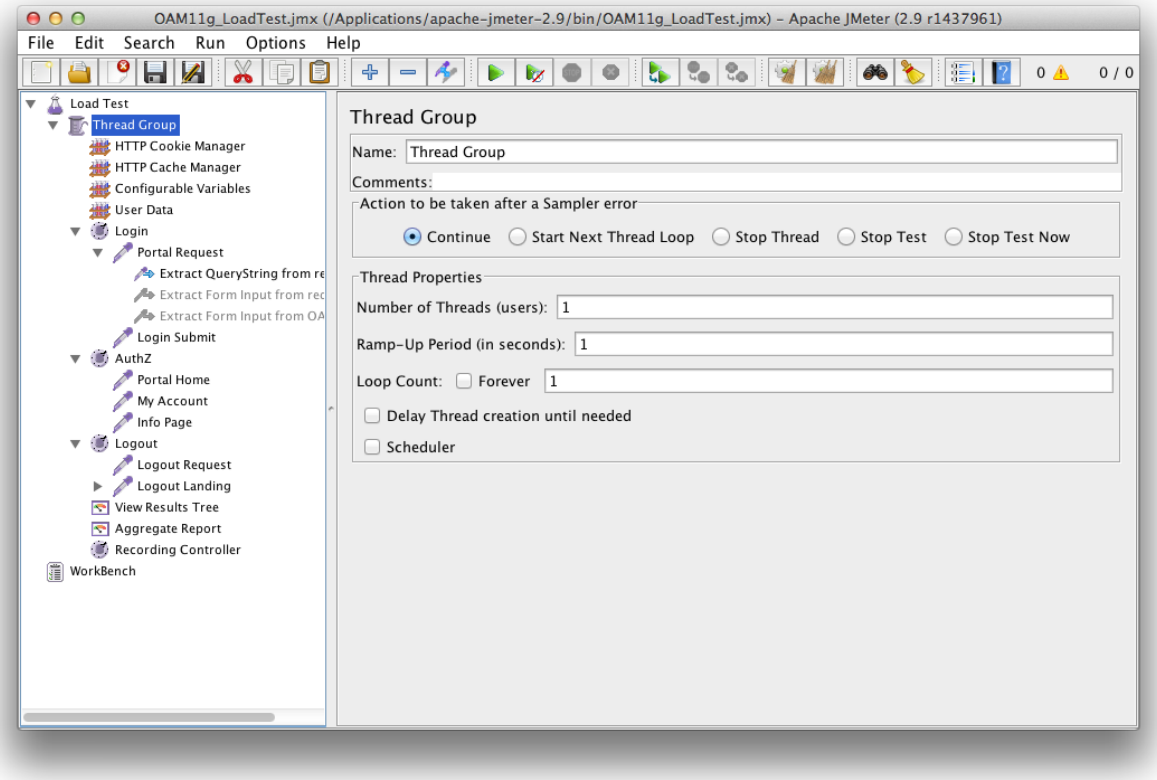

Рисунок 1.8 – Apache JMeter

У Apache JMeter є безліч опцій, на зразок динамічного звіту, переносимості та потужного IDE тестування. Крім цього, він підтримує різні типи додатків, скриптів оболонок, Java об'єктів і баз даних.

Додаток Apache JMeter є відкритим ПЗ, 100% чистим Java додатком, спроектованим для навантажувального тестування функціональної поведінки і вимірювання продуктивності. Спочатку його було створено для тестування веб-додатків, але з тих пір було розширено іншими функціями тестування.

Apache JMeter може використовуватися для випробування продуктивності як на статичних, так і на динамічних ресурсах, веб-динамічних додатках.

Його можна використовувати для симуляції сильного навантаження на сервер, групу серверів, мережу або об'єкт, щоб протестувати їх витривалість або проаналізувати загальну продуктивність під різними видами навантаження.

Katalon

Katalon – це відкрите ПЗ, призначене для автоматизованого тестування в веб та на мобільних пристроях. Це дуже простий багатоплатформовий інструмент, який має, крім іншого, дивовижну реалізацію JIRA.

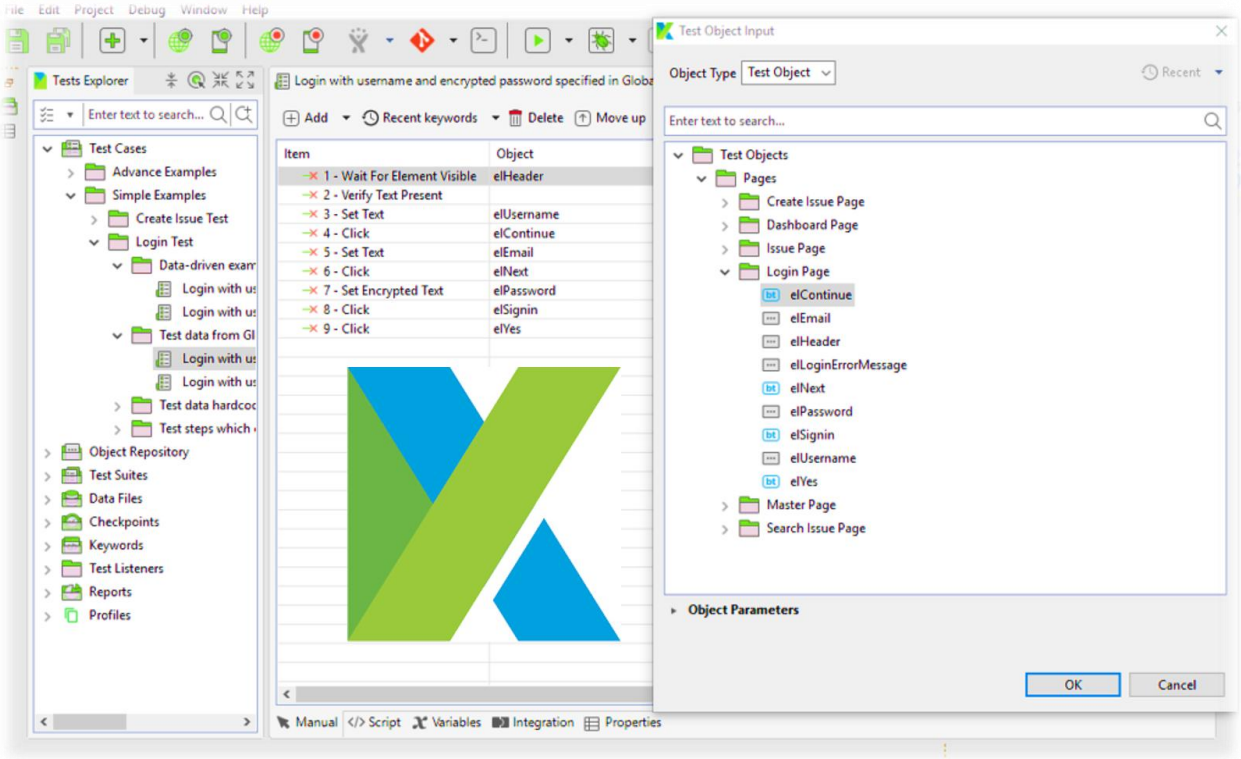

Рисунок 1.9 – Katalon

Katalon допомагає вам швидко генерувати автоматизовані крос-платформні тести, а також без зусиль інтегрувати ці тести в CI / CD лінію збірки. Крім цього, він надає централізовані звіти і якісні аналітичні дані, що формуються Katalon TestOps.

Maven

Maven – це безкоштовний інструмент для автоматизованого тестування проектів Java.

Maven – це інструмент з відкритим вихідним кодом, призначений для автоматизації збирання і використовується найчастіше для проектів Java. Є доступні плагіни для тестування. Мета "surefire: test", переслідувана однойменним плагіном Surefire, пов'язана з фазою тестування життєвого циклу управління ПЗ.

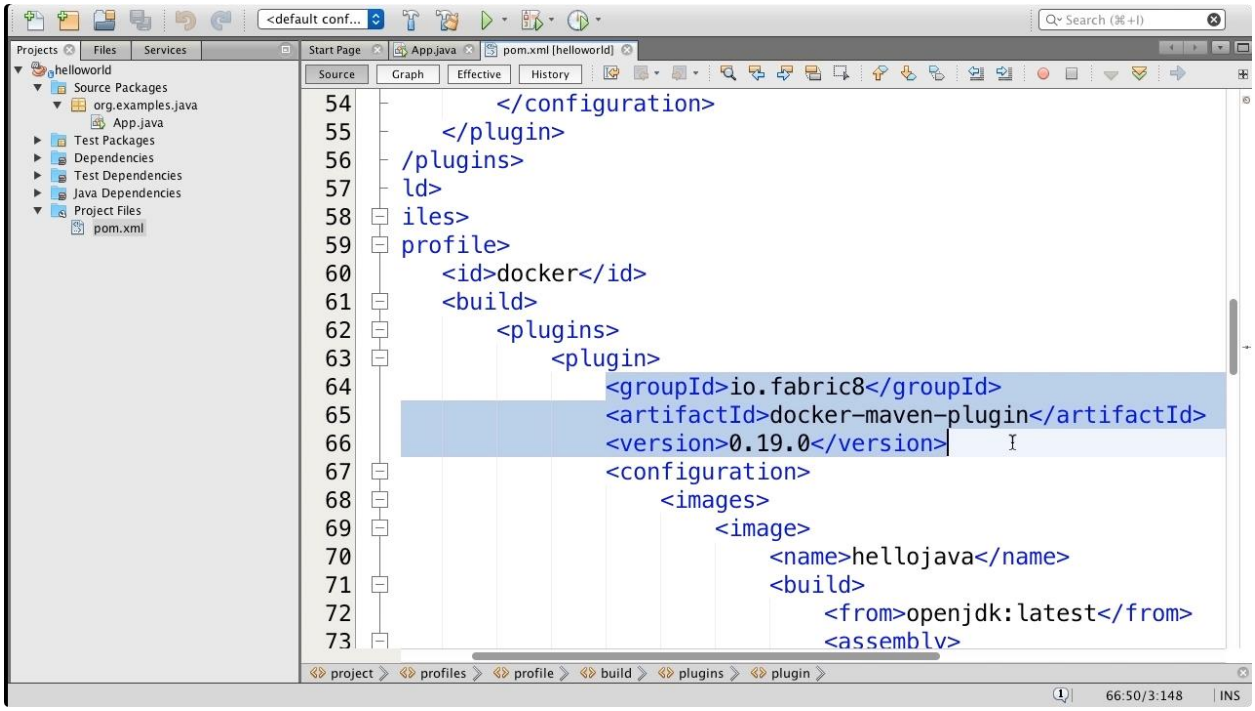

Рисунок 1.10 – Maven

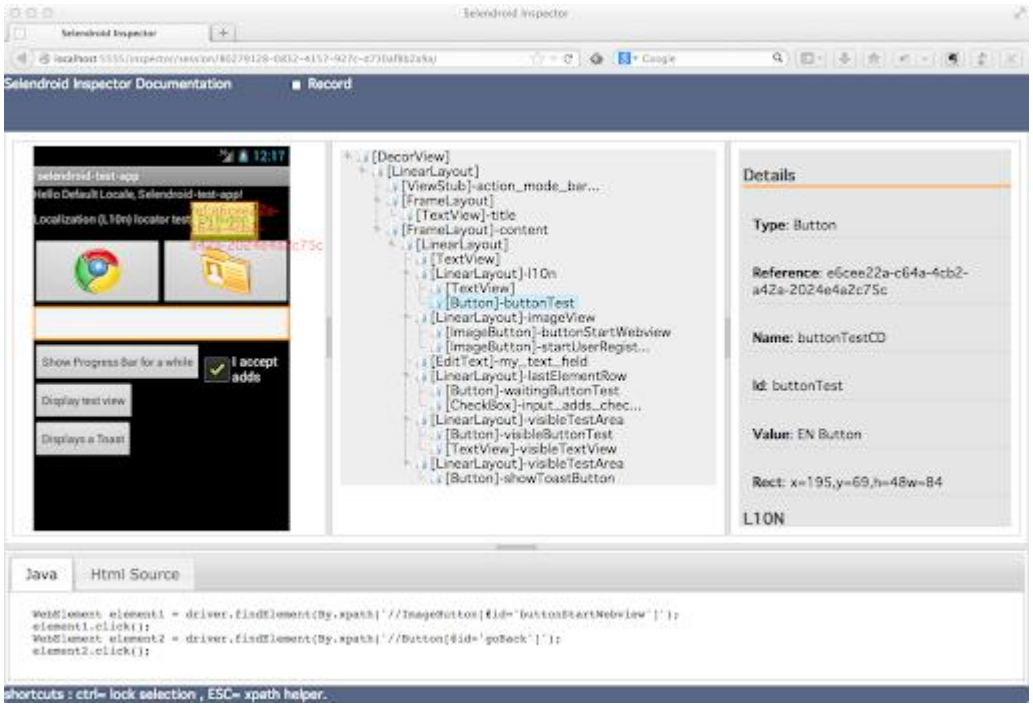

Рисунок 1.11 – Selendroid

Selendroid

Це безкоштовний фреймворк для автоматизації додатків Android і мобільних мереж. Його головна особливість – підтримка масштабування і паралельного тестування.

Selendroid є фреймворком для автоматизації тестування, який працює з UI нативних і гібридних додатків, а також бездротової локальної мережі. Тести написані з використанням клієнтського API Selenium 2.

Linux Desktop Testing Project

LDTP – це безкоштовний інструмент, головним чином націлений на тестування GUI з широким спектром мов.

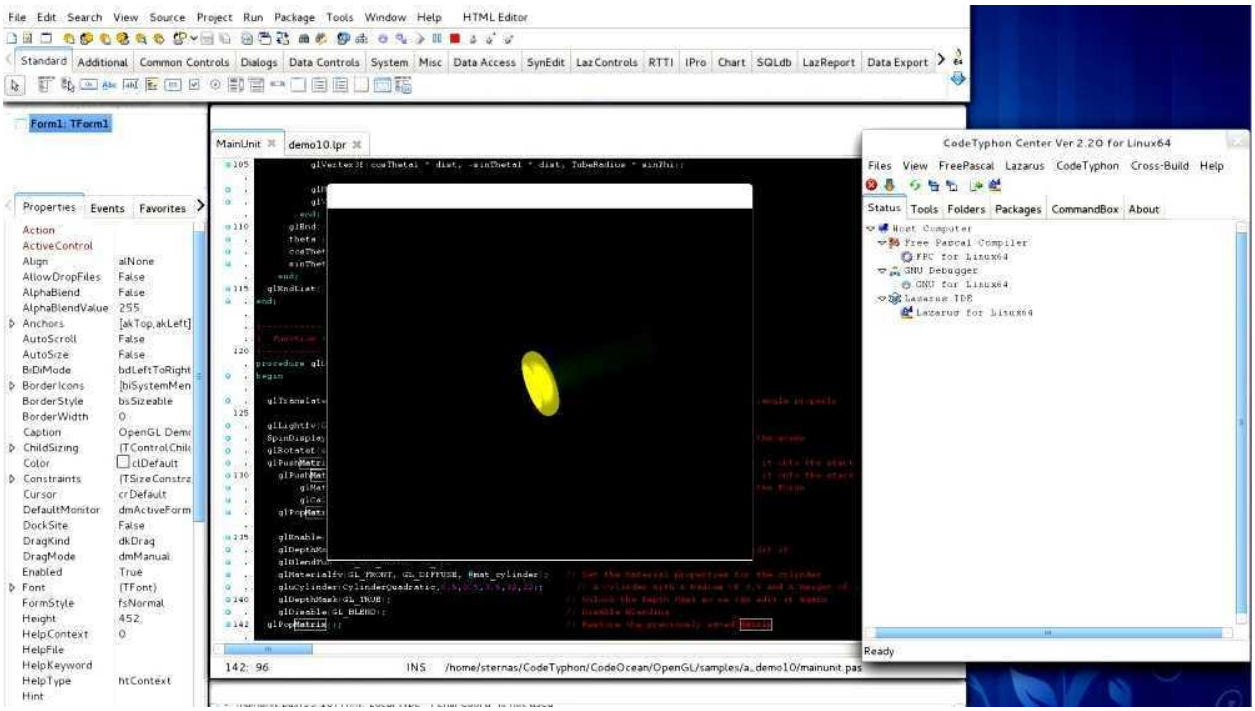

Рисунок 1.12 – Linux Desktop Testing Project

Проект тестування GNU LDTP спрямований на створення високоякісного фреймворка для автоматизації тестування, спорядженого новітніми інструментами, які можуть використовуватися для тестування і поліпшення робочих столів GNU / Linux або Solaris. Він використовує бібліотеки доступності , щоб зробити перевірку UI додатку.

OpenTest

Це безкоштовний інструмент для тестування веб, мобільних додатків і API. OpenTest є відкритим фреймворком для автоматизації функціонального тестування веб та мобільних додатків, а також API. Він розроблений для

масштабування і розширення з акцентом на включення основних методів автоматизації процесу тестування.

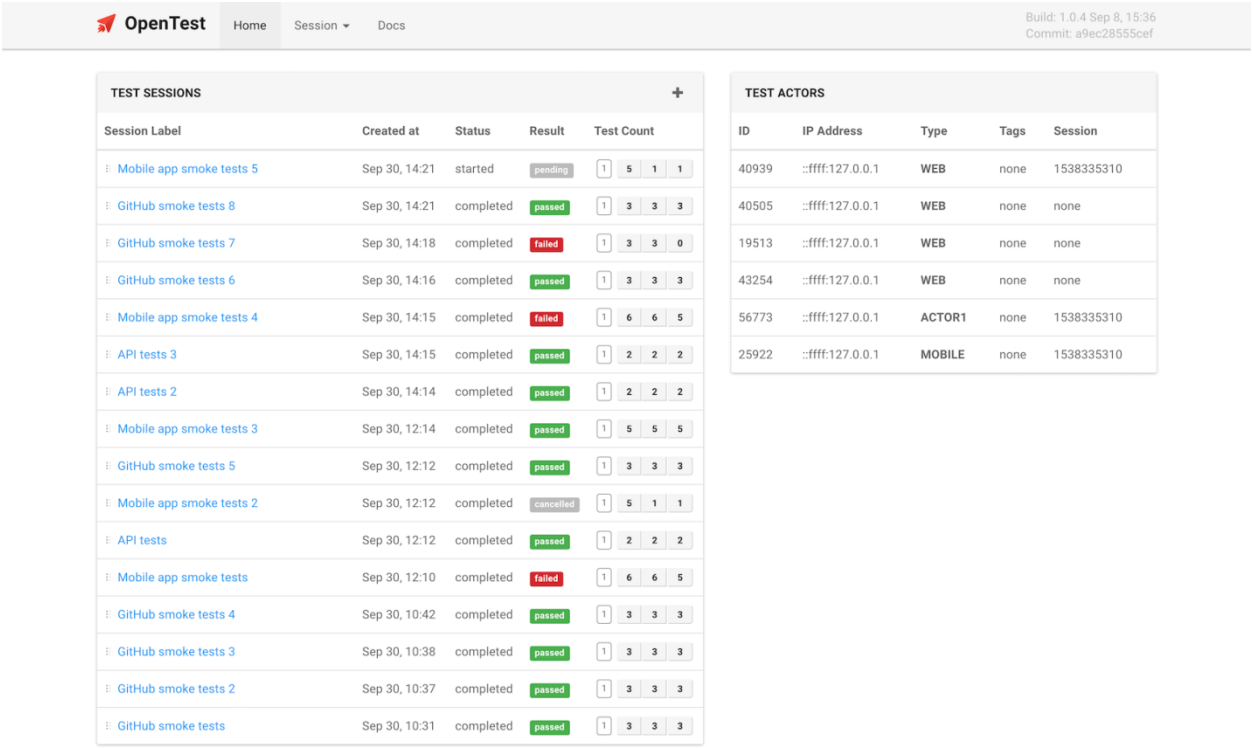

Рисунок 1.13 – OpenTest

OpenTest має багатий арсенал інструментів, вимагає мінімум навичок написання коду і може обробляти майже будь-який проект по автоматизації тестування.

2 ДОСЛІДЖЕННЯ ПРИНЦИПІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Цільовий аналіз системи автоматизованого тестування веб-орієнтованої інформаційної системи

За останні 15 років автоматизація тестування пройшла довгий шлях. Змінилися цілі і роль сервісу в IT-процесах: колись вона впроваджувалася тільки для скорочення часу тестування, а зараз до цього додається забезпечення оптимального тестового покриття і більш ефективне використання тест-кейсів.

Таким чином, впроваджуючи даний сервіс в цикл розробки сьогодні, компанії переслідують комплексну і важливу мету – отримати високоякісний програмний продукт, і як можна швидше.

Однак далеко не всі організації застосовують автоматизацію тестування ПЗ. Згідно з даними Світового звіту за якістю (World Quality Report) 2018-2020, 61% респондентів підкреслили, що зіткнулися з труднощами при її впровадженні на проектах [11].

Причина в тому, що зміни в логіці вже розробленої функціональності, а також випуск нових версій ПЗ вимагають проведення додаткових перевірок нової функціональності, регресійного тестування, а також регулярної підтримки Автотесту і їх зміни згідно актуальним вимогам до системи.

Останнім часом в автоматизації часто використовуються ІС, машинне навчання і аналітика, які дають їй нові переваги і роблять її «розумніше» і ефективніше.

Для досягнення значних результатів компанії повинні враховувати, що автоматизоване тестування – це не просто сервіс для заміни ручних перевірок або додатковий спосіб знизити витрати. У найзагальнішому сенсі воно націлене на те, щоб забезпечити високу якість ПЗ швидше, вивести процес QA на новий рівень, а також ефективно реалізовувати DevOps і Agile.

Щоб отримати максимальний ефект від сервісу, потрібно звертати увагу як на якість системи автоматизованого тестування, так і на якість процесів по впровадженню автоматизації, а також розуміти, коли потрібно автоматизувати, а коли цього робити не варто.

Таблиця 2.1 – Необхідність впровадження автоматизації

Автоматизація корисна в наступних випадках	Автоматизація не буде корисною, якщо
Першочергова задача - заощадити час проектної команди.	Для виконання тестування потрібен людський інтелект і інтуїція.
Тести повинні виконуватися для кожної збірки додатку.	Процес тестування обмежений інтуїтивними або дослідницькими перевірками.
Проект тривалий або комплексний (складається з різних ітерацій).	Вимоги, що ставляться до існуючої функціональності, часто змінюються.
На виконання тест-кейсів витрачається багато часу і ресурсів.	Потрібно провести тестування лише одного разу.
Проводиться навантажувальне або стрес-тестування.	
Потрібно скоротити поточний обсяг тестування з метою встигнути до певного строку.	

Ручне тестування все ще відіграє важливу роль в процесі забезпечення якості ПЗ. Однак в умовах процесу безперервного тестування (continuous testing) воно може бути вкрай ресурсовитратним [22].

Тим часом автоматизація збільшує швидкість виконання перевірок і економить ресурси команди QA-інженерів. Автоматизовані перевірки зменшують кількість тестів, які повинні бути виконані вручну (наприклад, під час тестування багатомовних сайтів), але, в той же час, не виключають їх.

У деяких випадках автоматизація застосовуватися не може. Наведемо приклади:

Тестування для користувача інтерфейсу. Ручне тестування перевірить загальний вигляд програми (чіткість зображень, розташування і відображення елементів графічного інтерфейсу при різних дозволах екрану і

ін.), а також окремо взяті компоненти (наприклад, колір шрифту - чи зможе кінцевий користувач його легко сприймати). Ручні перевірки покажуть, чи відповідає графічний інтерфейс перевагам споживачів.

Тестування зручності використання. Воно відповість з точки зору користувача на питання, є додаток простим у використанні чи ні.

Інтуїтивне тестування. Це тип перевірок, при якому тест-кейси не створюються заздалегідь, а QA-інженери тестують додаток і досліджують його «на ходу».

В ідеалі потрібно навчитися поєднувати ручне тестування і автоматизацію так, щоб вони примножували цінність один одного, а не применшували [17].

Розглянемо 5 випадків, коли така «командна робота» буде особливо вдалою:

У порівнянні з результатами, отриманими в 2018 році, кількість респондентів, які впроваджують автоматизоване тестування ПЗ на етапі розробки, зросла на 12%. Кількість опитаних IT-фахівців, які почали автоматизацію на етапі забезпечення якості, скоротилося на 9%.

Що стосується ручного тестування, то, в порівнянні з попереднім роком, модель «shift left» простежується слабо. На контрасті з 2018 роком, відсоток опитаних респондентів, які почали тестування програмного забезпечення вручну на стадії розробки, знизився на 2%. Кількість фахівців, які впроваджують ручне тестування на етапі стейджінгу, збільшилася на 4%.

Автоматизація може бути вельми прибутковою, якщо побудувати грамотний процес щодо її впровадження.

Впровадження автоматизації – це не одноразова дія, так як потрібно витрачати час на підтримку актуальності автоматизованого тестування.

Кількість нових регресійних дефектів завжди буде трохи менше в порівнянні з тими помилками, що будуть виявлені в новій функціональності.

Іноді результати тестування потрібні протягом декількох годин, особливо якщо говорити про перевірки недавно впровадженої

функціональності. В цьому випадку ефективніше перейти до ручного тестування і отримати результати якомога швидше.

Проте, плюси автоматизації для будь-якого проекту можуть бути безмежними [29].

Таблиця 2.2 – Переваги впровадження автоматизованого тестування

Переваги автоматизації тестування	Ефект
Вплив на ROI	Незважаючи на те що для написання тест-кейсів потрібні певні ресурси, окупність у даного підходу для великих і довгострокових проектів може бути величезною. Це можна пояснити тим, що тести легко налаштовуються і можуть багаторазово використовуватися. Впроваджуючи автоматизацію, можна значно знизити вартість кожної години, що витрачається на перевірки, а також знайти недоліки, що найбільш важко виявляються.
Швидкий зворотній зв'язок про якість додатку	Автоматизація може дати можливість швидко зібрати зворотний зв'язок щодо стану програмного продукту і забезпечити більш високу ефективність роботи команди розробників. Це покращує інтеграцію між інженерами по забезпеченню якості, програмістами, менеджерами та іншими учасниками процесу роботи над ПЗ.
Більш раннє виявлення дефектів програмного забезпечення	Оскільки автоматизація тестування допомагає збільшити швидкість розробки, вона стає більш рентабельною і значно спрощує виявлення помилки і її усунення на ранній стадії життєвого циклу системи.
Економія часу	Автоматизація регресійних тестів економить час QA-інженерів і звільняє більше ресурсів для проведення інтуїтивних перевірок та інших активностей по тестуванню. Крім того, при коректному використанні, автоматизовані тести можуть виконуватися з мінімальною залученістю людини або взагалі без неї.
Безпека тестових даних	Ефективність перевірок багато в чому залежить від якості використовуваних тестових даних. Створювати їх вручну досить ресурсомістко, тому тестування часто проводиться на копіях «живих» БД.

	Автоматизовані тести можуть бути спроектовані таким чином, щоб вони створювали і отримували необхідні дані, змінювали їх (не зачіпаючи інші) і поверталися у початковий стан.
Максимальна швидкість виконання перевірок	Автоматизація дозволяє проходити етапи проведення тестів швидше, ніж це робить людина. Це особливо актуально для DDT (тестів, керованих даними), оскільки одні й ті ж перевірки проводяться багато разів, але з різними наборами даних.

Автоматизація тестування – це оптимальний спосіб досягти багатьох QA-цілей. Вона відмінно підійде тим організаціям, які прагнуть представити на ринок видатні програмні продукти і хочуть залишатися конкурентоспроможними в своїй галузі дуже довго. Для більш детального розуміння необхідно детальне дослідження алгоритмів системи автоматизованого тестування веб-орієнтованої інформаційної системи.

2.2 Дослідження алгоритмів системи автоматизованого тестування веб-орієнтованої інформаційної системи

В автоматизованому тестуванні виділяють такі підходи:

1) TDD (англ. Test Driven Development);

2) BDD (англ. Behaviour Driven Development);

3) KDT (англ. Keyword Driven Testing);

4) DDT (англ. Data-driven testing).

Data-Driven Testing – це тестування, кероване даними. При такому підході тестові дані зберігаються окремо від тест-кейсів, припустимо, в файлі або в базі даних. Такий поділ логічно спрощує тести [7].

Data-Driven Testing використовується в тих проектах, де потрібно виконати тестування окремих додатків в декількох середовищах з великими наборами даних і стабільними test cases.

Зазвичай при DDT виконуються наступні операції:

– витяг частини тестових даних зі сховища;
– введення даних в форму додатка;
– перевірка результатів;
– продовження тестування з наступним набором вхідних даних.

Підхід Data-Driven Testing:

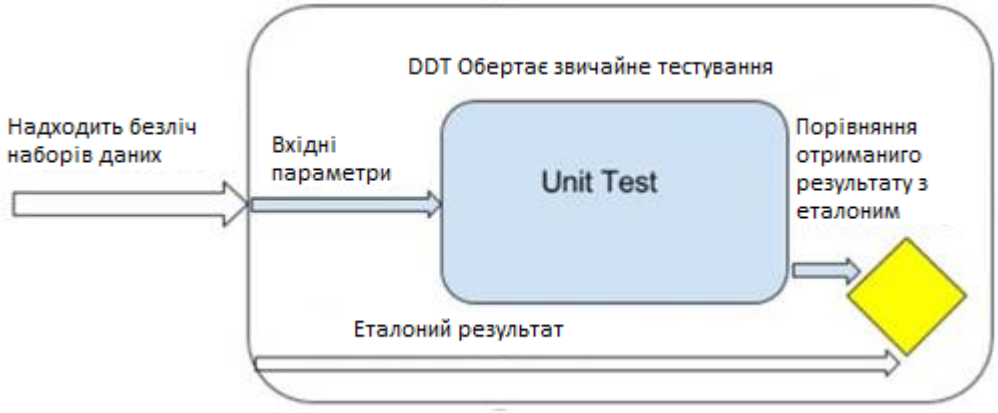

Рисунок 2.1 – Підхід Data-Driven Testing

Щоб перевірка додатка була успішна, будуть потрібні різні комбінації даних.

Keyword Driven Testing – мова йде про тести, керовані ключовими словами. Даний підхід передбачає використання ключових слів, що описують набір дій, потрібних для виконання конкретного кроку тестового сценарію.

При такому підході в першу чергу визначається набір ключових слів, а тільки після цього асоціюється функція дія якої пов’язана з даним ключовим словом. Наприклад, кожні кроки тесту, такі як клацання мишею, натискання клавіші, відкриття або закриття браузера описуються певними ключовими словами («відкрити» - openbrowser, «натиснути» - click і т. п.) [31].

При KDT-підході ви можете створювати прості функціональні тести на самих ранніх етапах розробки і тестувати додаток частинами.

Етапи розробки KDT-тестів:

1. Визначаємо ключові слова.

2. Реалізуємо ключові слова як виконувані файли.

3. Створюємо тест-кейси.

4. Створюємо скрипти.

5. Виконуємо автоматизовані сценарії.

Плюси підходу:

1) функціональні тестувальники можуть планувати автоматизацію тестування до того, як додаток буде готовий;

2) тести можна розробити без знань програмування;

3) підхід не залежить від вибраної мови програмування.

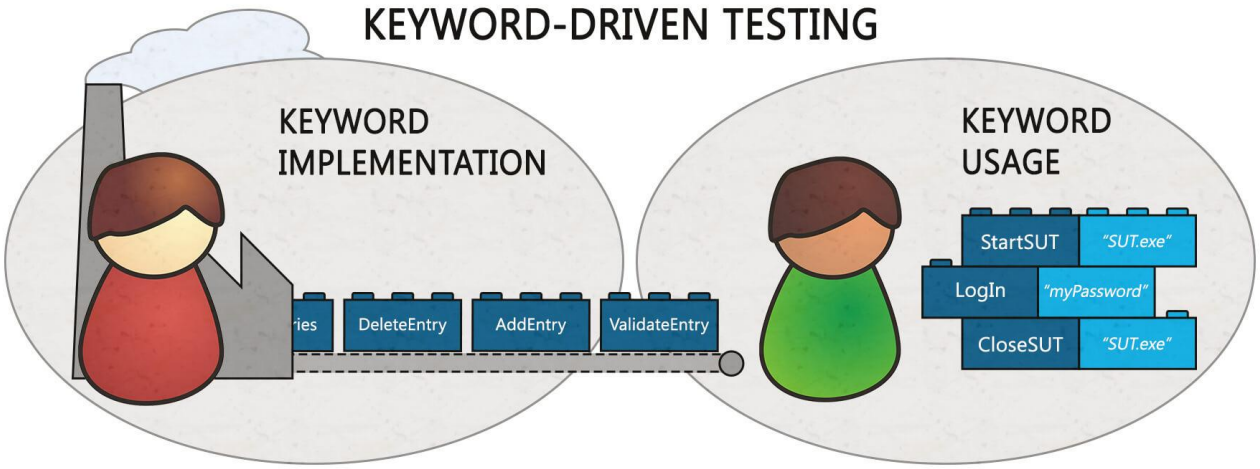

Рисунок 2.2 – Keyword Driven Testing

Test Driven Development

Підхід розробки через тестування (TDD) передбачає організацію автоматичного тестування за допомогою написання модульних, функціональних і інтеграційних тестів, що визначають вимоги до коду перед написанням коду . Тобто в першу чергу пишеться тест, який перевіряє коректність роботи ще ненаписаного коду. Тест, само собою, не проходить. Далі програміст пише код, де виконуються дії, необхідні для проходження тесту. Коли тест буде успішно пройдено, можлива доробка наявного коду.

Підхід Test Driven Development:

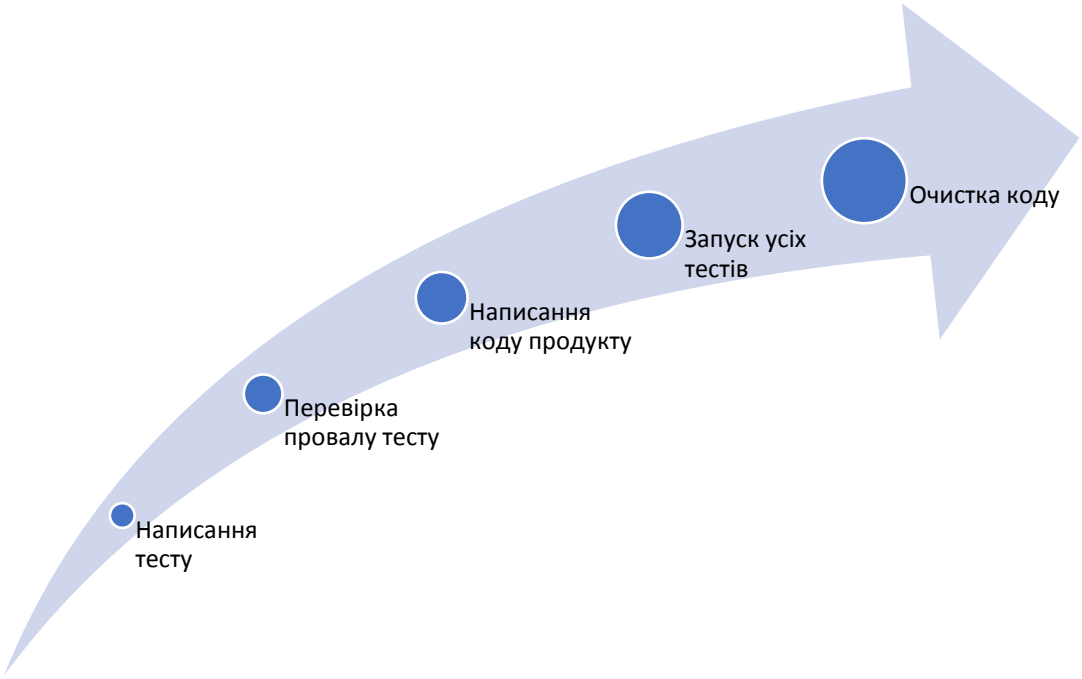

Рисунок 2.3 – Підхід Test Driven Development

Розробка через тестування – це більше, ніж просто перевірка коректності, так як вона впливає і на дизайн програми. Якщо користувач спочатку сфокусований на тестах, йому простіше уявити, яка саме функціональність потрібна користувачеві. В результаті розробник продумає деталі інтерфейсу до його реалізації. Це, в свою чергу, скоротить час на розробку і налагодження [29].

Крім того, розробка через TDD зосереджується на тестуванні окремо взятих модулів, при цьому використовуються заглушки (mock-об'єкти) для подання зовнішнього світу [17].

Behavior-driven development

Підхід BDD – це розробка, заснована на поведінці. По суті, BDD є різновидом (розширенням) TDD з тією лише різницею, що BDD-підхід орієнтований на поведінку суті, яку ви тестуєте (в TDD основний фокус йде безпосередньо на сам код).

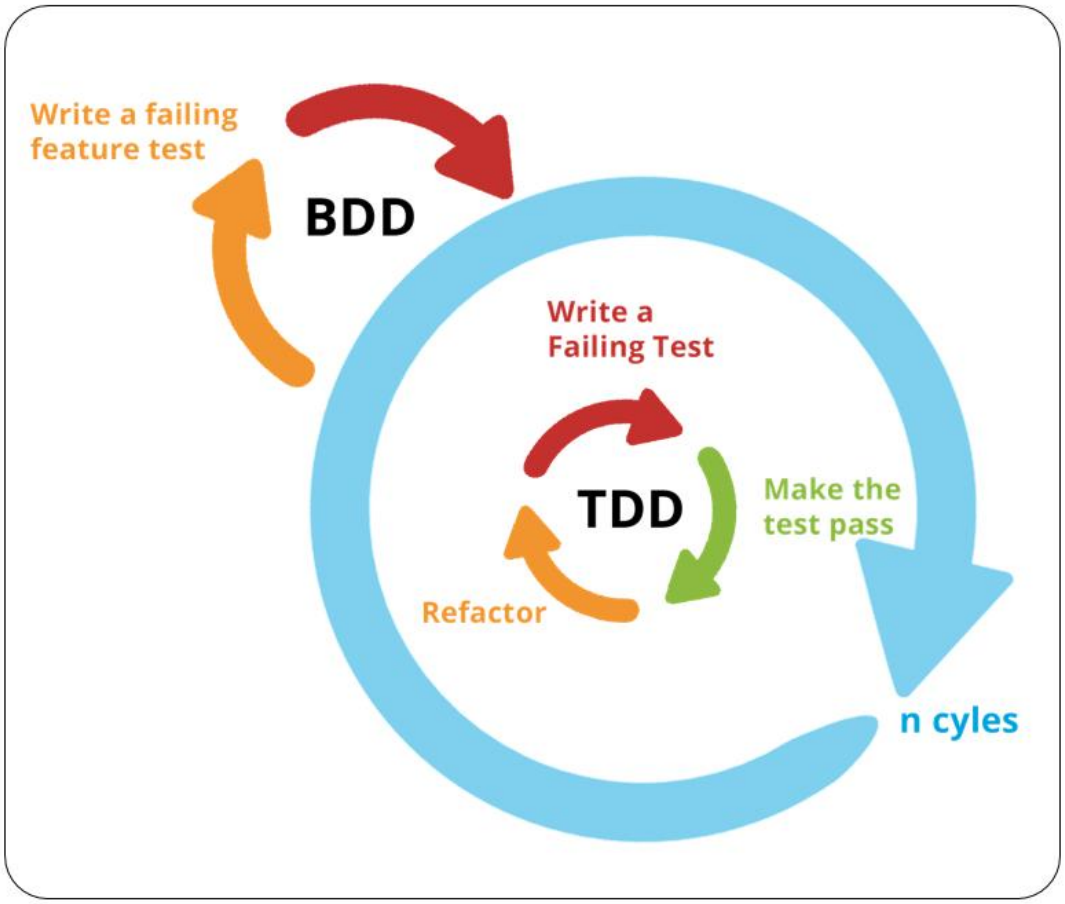

Рисунок 2.4 – Підхід Behavior-driven development

Суть BDD полягає в описі системи архітектури додатку в термінах, зрозумілих неспеціалісту. Це дає можливість прискорити процес отримання зворотного зв'язку, прибравши традиційні бар'єри. Тобто опис користувальницьких сценаріїв відбувається на природній мові – грубо кажучи, на мові бізнесу [7]. Для подальшого проектування необхідно здійснити дослідження інструментарію для розробки та удосконалення системи автоматизованого тестування веб-орієнтованої інформаційної системи.

2.3 Дослідження інструментарію для розробки та удосконалення системи автоматизованого тестування веб-орієнтованої інформаційної системи

У якості базового інструментарію обрано HTML, Java та CSS. На сьогодні, ринок програмного забезпечення для реалізації ідей програміста є доволі широким. Низка мов об’єктно-орієнтованого програмування та засобів для розробки веб-сервісів є масштабною.

У рамках даної роботи прийнято рішення використовувати HTML - стандартизовану мову розмітки веб-сторінок на базі Windows 10.

Кожен вибирає свій інструмент для створення Web-сторінок. Це може бути MS FrontPage або Macromedia Dreamweaver, Namo Web Editor, Allaire HomeSite або 1st Page 2000. А хтось користується простим текстовим редактором, наприклад Блокнотом (Notepad) [13].

Microsoft FrontPage

Для використання Microsoft FrontPage не вимагається знання мови HTML. У ході редагування сторінок як у текстовому редакторі – при введенні і форматуванні тексту, додаванні малюнків, таблиць та інших елементів сторінок – теги мови HTML, автоматично додаються у фоновому режимі. Можна просто редагувати сторінки в режимі конструктора. Щоб познайомитися з мовою HTML або прямо редагувати код HTML, можна використовувати режим Код, при якому відображається код HTML веб-сторінки, або режим з поділом, при якому сторінка одночасно відображається в режимі Код і в режимі Конструктор. При наявності навичок роботи з HTML в режимі Код можна відображати HTML-теги і безпосередньо писати і редагувати їх. За допомогою параметрів створення і оптимізації коду, доступних в Microsoft FrontPage можна створювати чистий код HTML і легко видаляти непотрібний код. Недоліком є те, що виникають проблеми при редагуванні і збереженні сторінок з фреймами [14].

Основним недоліком Microsoft FrontPage є те, що він генерує надлишковий HTML-код (занадто багато зайвого), тому сторінки виходять великими, що позначається на швидкості завантаження. Web Editor

Містить стандартний набір функцій для роботи з HTML-сторінками, можливість роботи з різними елементами на сторінках, існує великий вибір готових графічних шаблонів для оформлення сторінок; основний недолік – даний редактор не створює цілісний вузол для збереження окремих частин сайту, виникають проблеми при збереженні фреймів і зображень.

Створення та оптимізація графіки. Безумовно, можливе створення Web-сторінки і без використання графіки – за допомогою шрифтів, скриптів і таблиць стилів (CSS) – і це буде красиво і стильно. Але ж остаточний вид документа залежить від великої кількості різних факторів, таких як: ширина вікна браузера, попередні налаштування браузера, прийняті за замовчуванням розмір шрифту, його ім'я і колір. До того ж не всі скрипти та стилі підтримуються всіма браузерами. Якщо ж буде використана графіка, то відвідувач сторінки побачить її точно такою, якою її зробив і бачить саме розробник [24].

Основна складність роботи з Web- графікою полягає в тому, що пропускна здатність каналів Інтернету, в більшості випадків, дуже низька і перед вами відразу встануть проблеми – як зробити графічний файл невеликий за обсягом, але хорошої якості, які програми і прийоми використовувати при його оптимізації.

При створенні Web додатків застосовується безліч технологій і різних засобів розробки, з якими багато знайомі [35].

Коротко технології можна розділити так :

Мова гіпертекстової розмітки HTML. Мова широко використовується для створення сторінок в Web. Ця мова є фундаментальною в мережі Інтернет та знання її необхідно будь-якому розробнику. Динамічний мова гіпертекстової розмітки DHTML. Ця мова дозволяє створювати динамічні інтерактивні сторінки.

Мова сценаріїв JavaScript і VB Script. Сценарії, написані на цих мовах, застосовуються як на стороні клієнта, так і на стороні Web сервера.

Мова розробки скриптів JavaScript

Мова програмування JavaScript розроблена фірмою Netscape для створення інтерактивних HTML-документів. Це об'єктно-орієнтована мова розробки вбудованих додатків, що виконують як на стороні клієнта, так і на стороні сервера. Синтаксис мови дуже схожий на синтаксис мови Java – тому її часто називають Java-подібною. Клієнтські програми виконуються браузером перегляду Web-документів на машині користувача, серверні додатки виконуються на сервері [4].

При розробці обох типів додатків використовується загальний компонент мови, званий ядром і такий, що включає визначення стандартних об'єктів і конструкцій (змінні, функції, основні об'єкти і засіб LiveConnect взаємодії з Java-апплетами), і відповідні компоненти доповнень мови, що містять специфічні для кожного типу додатків визначення об'єктів.

Клієнтські додатки безпосередньо вбудовуються в HTML-сторінки і інтерпретуються браузером у міру відображення частин документа в його вікні.

Серверні додатки для збільшення продуктивності попередньо компілюються в проміжний байт-код.

Основні галузі використання мови JavaScript при створенні інтерактивних HTML-сторінок:

Динамічне створення документа за допомогою сценарію;

Оперативна перевірка достовірності заповнених користувачем полів форм HTML до передачі їх на сервер;

Створення динамічних HTML-сторінок спільно з каскадними таблицями стилів і об'єктної моделлю документа;

Взаємодія з користувачем при вирішенні локальних завдань, що вирішуються додатком JavaScript, вбудованому в HTML-сторінку.

3 ПРОЕКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1.1 Проектування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Для практичної демонстрації пропонованого підходу використовуємо реальну інформаційну систему, що має наступну розподілену архітектуру (рис. 3.1):

- клієнт через web-браузер звертається до додатка;
- запит обробляє сервер додатків, при необхідності вставляє дані, одержувані за запитом з сервера баз даних, формує вихідний HTML-файл;
- сформована сторінка відправляється клієнтові.

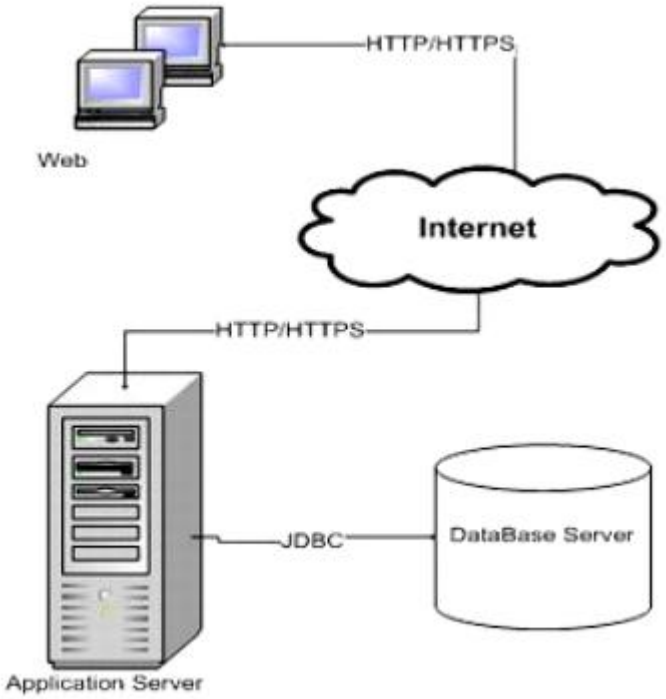

Рисунок 3.1 – Архітектура розподіленої системи

На сервері додатків обробкою даних займається пакет KemsuWEB, структура якого показана на рис. 3.2. Основним носієм інформації є XML-файл шаблону майбутньої сторінки з даними. У шаблоні присутні спеціальні конструкції, призначені для розширення можливостей обробки даних мови XML:

– виклик процедур або виконання запитів для отримання даних з сервера бази даних;
– управління логічними інструкціями;
– обробка локальних змінних і змінних сесії (глобальних для даного сеансу користувача).

Іншими словами, сервер додатків розділяє рівні зберігання даних, обробки даних і формування інтерфейсу користувача (рис. 3.3).

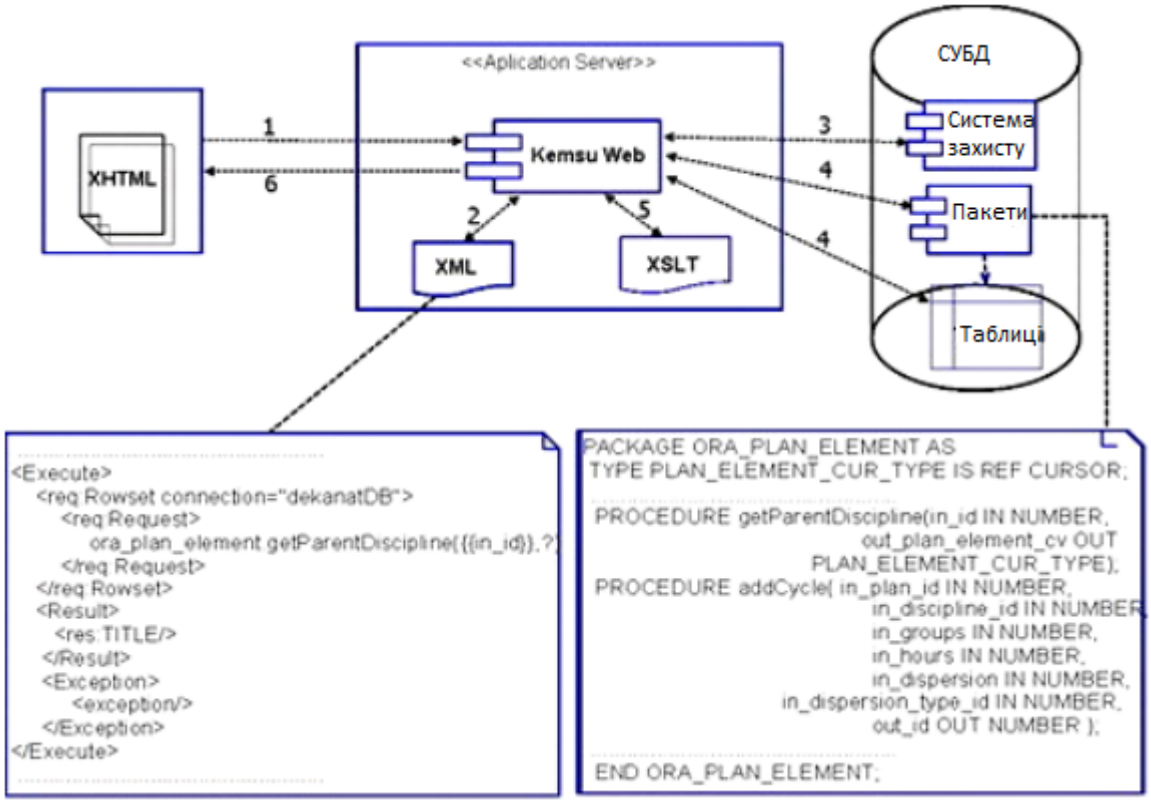

Рисунок 3.2 – Архітектура пакета KemsuWEB

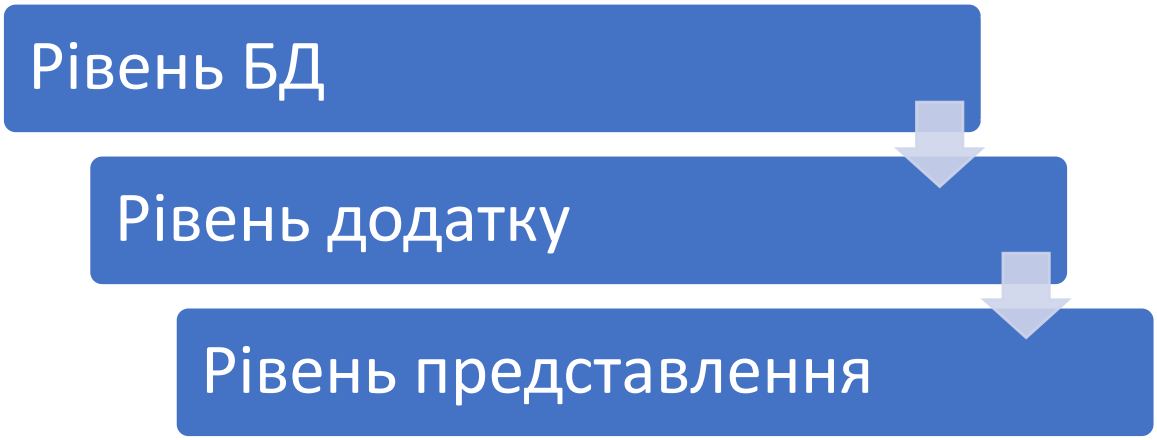

Рисунок 3.3 – Рівні додатку

Процес тестування такого додатка є складним завданням. Для її вирішення мало провести інкрементальне тестування окремих компонент додатка, інтерфейсу користувача, коректності роботи запитів до даних і т.д. Необхідно накопичити статистику виконання тестових сценаріїв і станів додатки, визначити структуру тестованої системи. Крім того, під час виконання тестових сценаріїв бажано мати можливість швидко змінювати структуру програми та визначати гілку структурного графа додатку, яку необхідно перевірити. Безліч значень вхідних параметрів, що задаються через форми web-інтерфейсу, також необхідно накопичувати разом з відповідними повідомленнями системи. Варто виконати розробку системи автоматизованого тестування веб-орієнтованої інформаційної системи.

3.1.2 Розробка системи автоматизованого тестування веб-орієнтованої інформаційної системи

Проведемо розробку системи автоматизованого тестування веб-орієнтованої інформаційної системи, для цього сформуємо діаграму прецедентів (рис. 3.4).

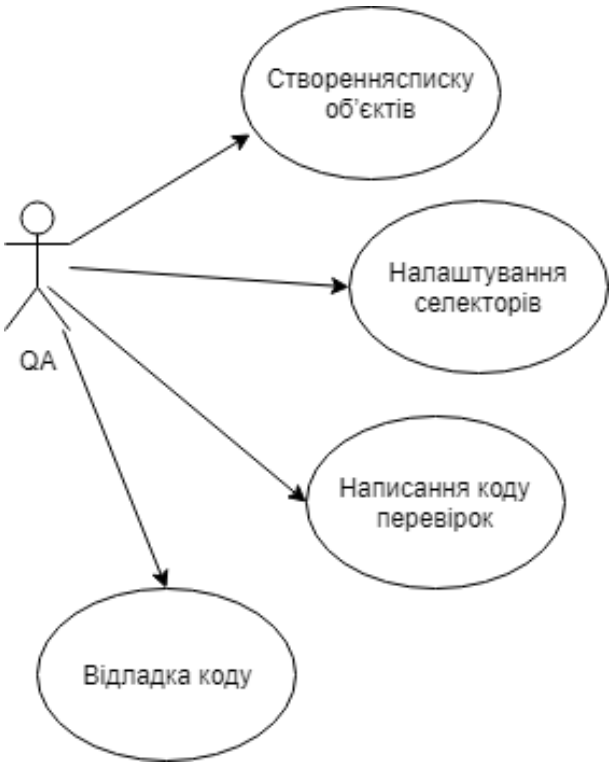

Рисунок 3.4 – Діаграма прецедентів системи автоматизованого тестування веб-орієнтованої інформаційної системи

Головною дієвою одиницею створення системи автоматизованого тестування веб-орієнтованої інформаційної системи є проектувальник, тобто людина що створює автоматизовану систему тестування для проведення незалежного тестування веб-орієнтованої інформаційної системи.

За рахунок виконання процесу програмування проектувальник виконує низку дій, які є обов'язковими для реалізації дієвої системи. Діаграма прецедентів процесу написання системи автоматизованого тестування веб-орієнтованої інформаційної системи наведена на рисунку 3.5.

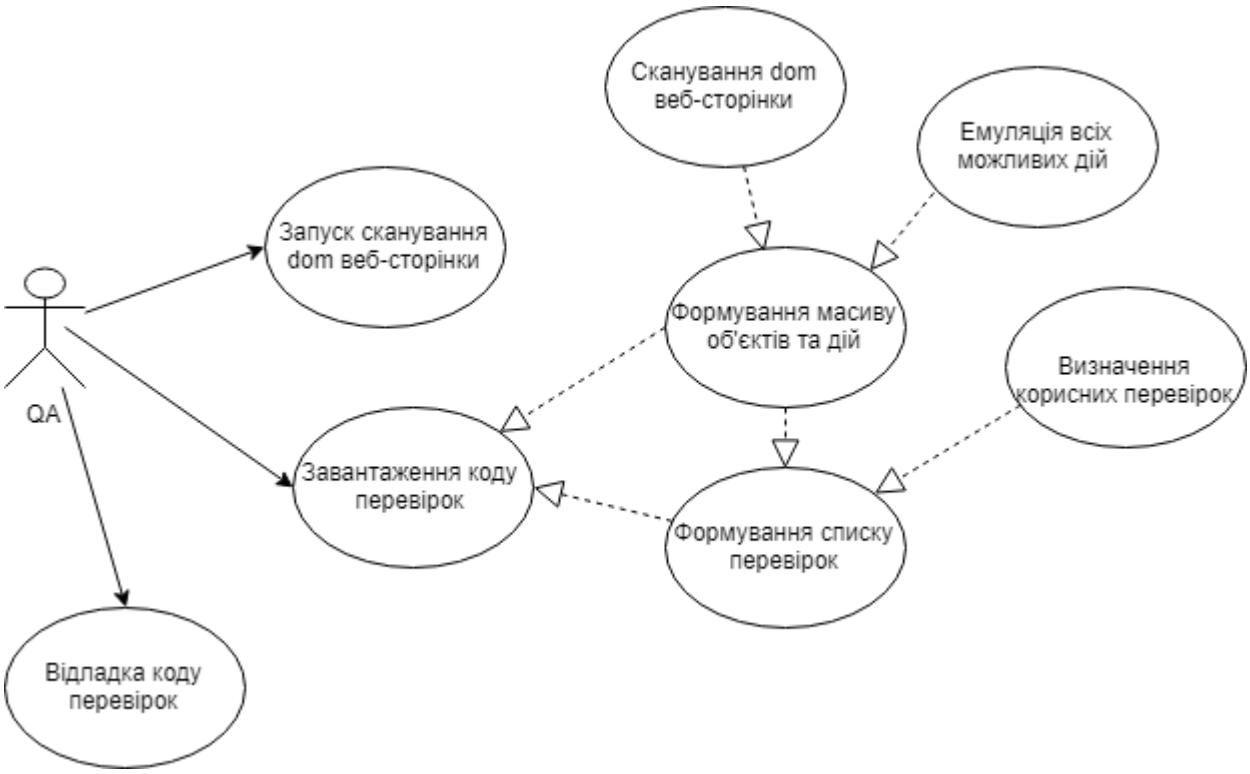

Рисунок 3.5 – Діаграма прецедентів процесу написання системи автоматизованого тестування веб-орієнтованої інформаційної системи

Схема реалізації процесу тестування наведена на рисунку 3.6.

З рисунку видно, що початкова дія включає в себе додавання основної інформації про об’єкт тестування його параметри та інтерфейс, наступною дією є перевірка чи всі об’єкти додано та описано, якщо так то переходимо до написання коду перевірок, якщо ні повертаємося до пункту додавання інформації. Після написання коду перевірок за умови наявності всіх елементів у системі здійснюємо відладку коду, та переходимо безпосередньо до тестування веб-застосувань.

Враховуючи часову залежність від написання коду до його реалізації на практиці доцільним є висвітлення схеми структурної діяльності взаємодії проектувальника з написання коду з тестувальником який безпосередньо втілює систему у реальне життя. На рисунку 3.7 наведено схему структурної діяльності взаємодії проектувальника з написання коду з тестувальником.

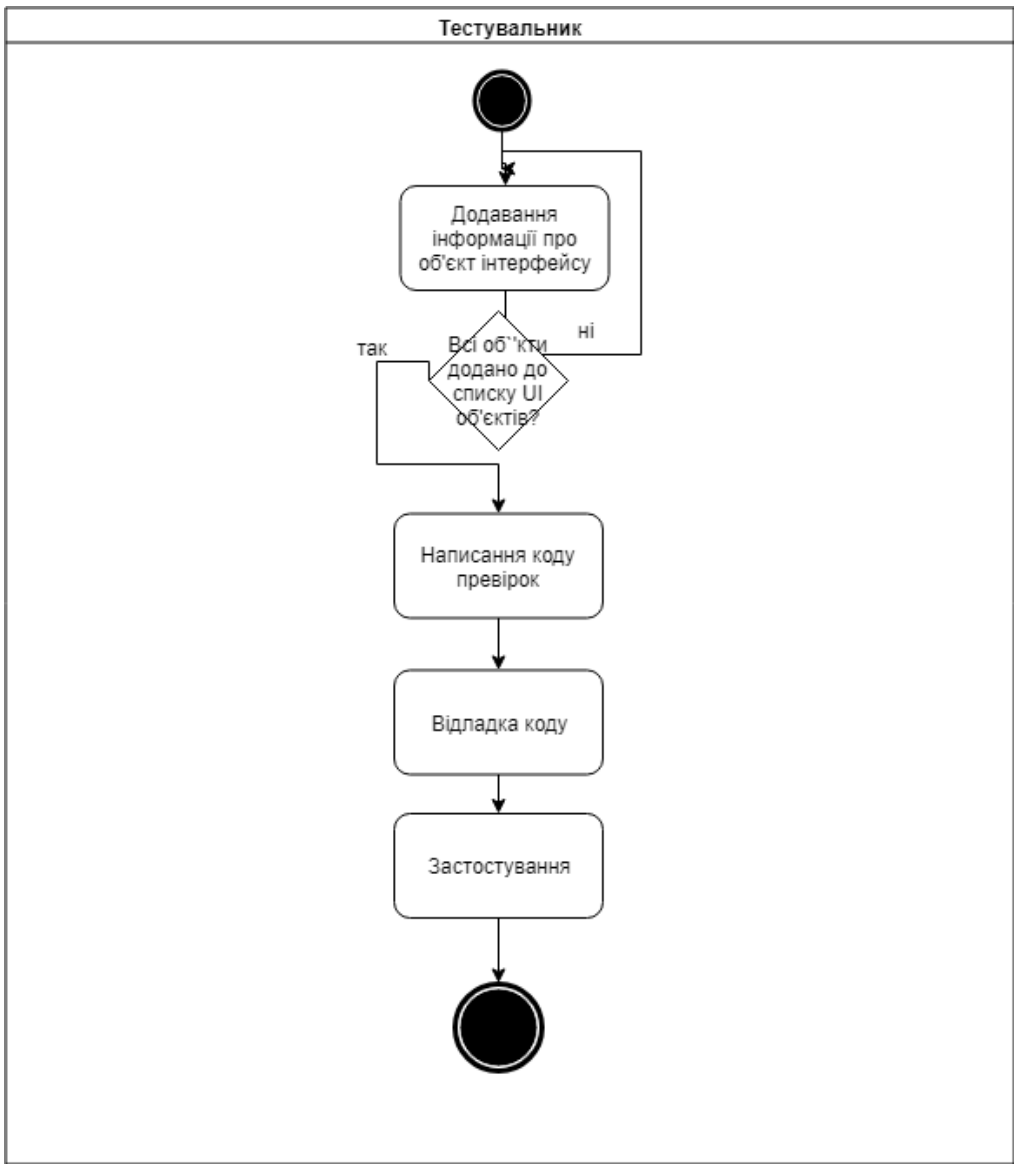

Рисунок 3.6 – Схема реалізації процесу тестування

У процесі взаємодії бере участь три окремі одиниці це проектувальник, тестувальник та система автоматизованого тестування веб-орієнтованої інформаційної системи. Їх спільна робота полягає у прямій залежності від дій один одного та взаємній співпраці.

Тестувальник у процесі проведення автоматизованого тестування веб-орієнтованої інформаційної системи здійснює підлаштування основних параметрів під веб-орієнтовану інформаційну систему, яка підлягає автоматизованому тестуванню.

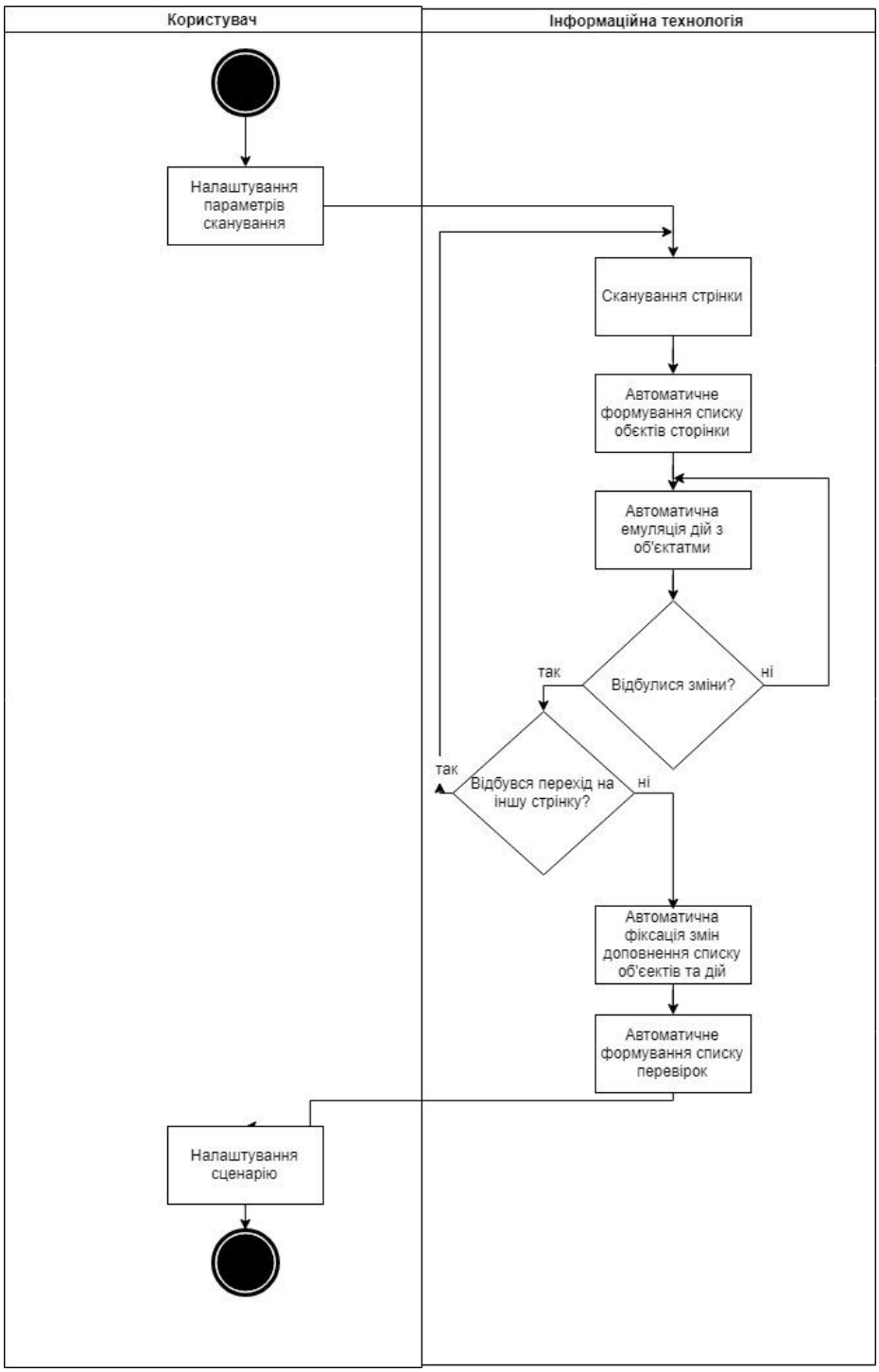

Рисунок 3.7 – Схема структурної діяльності взаємодії проектувальника з написання коду з тестувальником

Загальну взаємодію інструментів тестування наведено на рисунку 3.8.

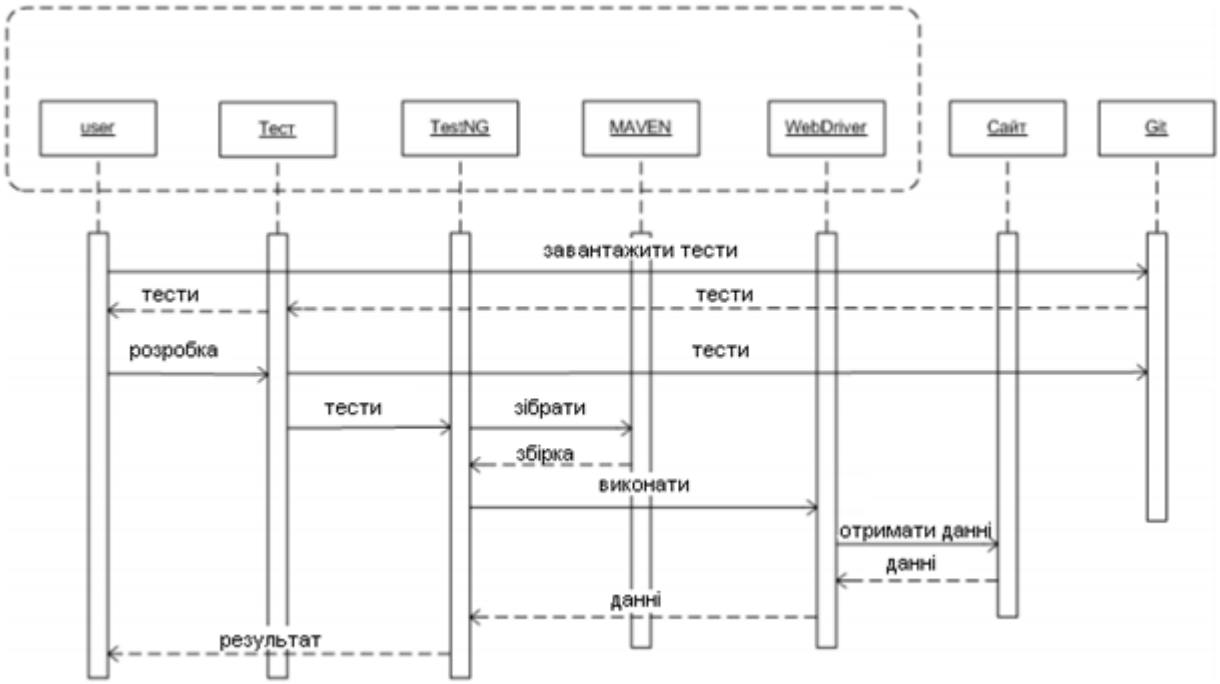

Рисунок 3.8 – Взаємодія інструментів тестування

Чітке виконання послідовності взаємодії дає максимально точний результат тестування. На основі наведених схем та діаграм із застосуванням описаного інструментарію та алгоритмів здійснимо розробку системи автоматизованого тестування веб-орієнтованої інформаційної системи та проведемо її тестування на реальному наборі даних. Наступним кроком виконаємо тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи.

3.2 Тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Проведемо тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи на реальному наборі даних за для виявлення слабких сторін та підтвердження її працездатності.

Результаты автоматизированного тестирования

Набор	Проверено	Ошибки	Пропущено	testng.xml
Выполнено	*21*	*5*	*0*	
Набор по умолчанию	21	5	0	Ссылка на сайт

Рисунок 3.9 – Результат тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Результати тестування наведемо на рисунках 3.9-3.14.

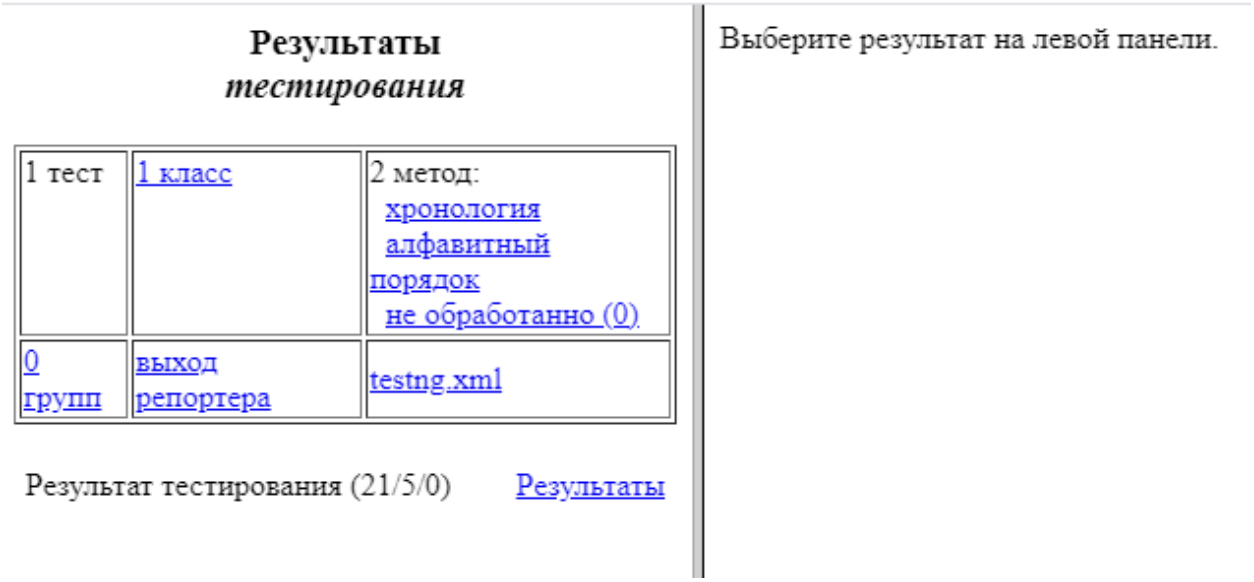

Рисунок 3.10 – Результат тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Рисунок 3.11 – Результат тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Результаты *тестирования*

1 тест	1 класс	2 метод: хронология алфавитный порядок не обработанно (0)
0 групп	выход репортера	testng.xml

Результат тестирования (21/5/0) Результаты

Автоматизированное тестирование

Тесты пройдены / не пройдены / пропущены:	21/5/0
Начато:	Ср май 05 18:29:04 год 2021
Общее время:	27 секунд (27448 мс)
Включенные группы:	
Исключенные группы:	

(Наведите указатель мыши на имя метода, чтобы увидеть имя тестового класса)

Результат тестирования

Тестовый метод	Исключение
	java.lang.AssertionError: expected:<0> but was:<31> at org.testng.AssertJUnit.assertEquals(AssertJUnit.java:115) at org.testng.AssertJUnit.assertEquals(AssertJUnit.java:125) at cs265.PizzaHouseSystemTest.TestTemplate(PizzaHouseSystemTest.java:117) at cs265.PizzaHouseSystemTest.invalidDataTests(PizzaHouseSystemTest.java:132) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.j

Рисунок 3.12 – Результат тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Тест	# Пройдено	# Пропущено	# Повторная попытка	# Не пройдено	Время (мс)	Включенные группы	Исключенные группы
Набор по умолчанию							
Тест по умолчанию	21	0	0	5	27,448		

Класс	Метод	Старт	Время (мс)
Набор по умолчанию			
Тестирование — не пройдено			
cs265.PizzaHouseSystemTest	invalidDataTests	1607970555668	767
	invalidDataTests		
	invalidDataTests		
	invalidDataTests		
	validDataTests	1607970566689	714
Default test — passed			
cs265.PizzaHouseSystemTest	invalidDataTests	1607970551831	752
	invalidDataTests		

Рисунок 3.13 – Результат тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Параметр #1	Параметр #2	Параметр #3	Параметр #4
0	3	CARD	0

Exception

```
java.lang.AssertionError: expected:<0> but was:<46>
        at org.testng.AssertJUnit.assertEquals(AssertJUnit.java:115)
        at org.testng.AssertJUnit.assertEquals(AssertJUnit.java:125)
        at cs265.PizzaHouseSystemTest.TestTemplate(PizzaHouseSystemTest.java:117)
        at cs265.PizzaHouseSystemTest.invalidDataTests(PizzaHouseSystemTest.java:132)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:564)
        at org.testng.internal.MethodInvocationHelper.invokeMethod(MethodInvocationHelper.java:132)
        at org.testng.internal.TestInvoker.invokeMethod(TestInvoker.java:599)
        at org.testng.internal.TestInvoker.invokeTestMethod(TestInvoker.java:174)
        at org.testng.internal.MethodRunner.runInSequence(MethodRunner.java:46)
        at org.testng.internal.TestInvoker$MethodInvocationAgent.invoke(TestInvoker.java:822)
        at org.testng.internal.TestInvoker.invokeTestMethods(TestInvoker.java:147)
        at org.testng.internal.TestMethodWorker.invokeTestMethods(TestMethodWorker.java:146)
        at org.testng.internal.TestMethodWorker.run(TestMethodWorker.java:128)
        at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
        at org.testng.TestRunner.privateRun(TestRunner.java:764)
        at org.testng.TestRunner.run(TestRunner.java:585)
        at org.testng.SuiteRunner.runTest(SuiteRunner.java:384)
        at org.testng.SuiteRunner.runSequentially(SuiteRunner.java:378)
        at org.testng.SuiteRunner.privateRun(SuiteRunner.java:337)
        at org.testng.SuiteRunner.run(SuiteRunner.java:286)
        at org.testng.SuiteRunnerWorker.runSuite(SuiteRunnerWorker.java:53)
```

Рисунок 3.14 – Результат тестування системи автоматизованого тестування веб-орієнтованої інформаційної системи

Під час проведення тестування збоїв та недоліків у роботі системи автоматизованого тестування веб-орієнтованої інформаційної системи не виявлено, що говорить про високу якість розробки та можливість впровадження її за потребою.

4 ОЦІНКА ПОКАЗНИКІВ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1.1 Опис методології дослідження ефективності розробленої системи

На сьогоднішній день, одним з головних напрямків ефективної реалізації будь-якої методики є експериментальне дослідження тобто виявлення якостей досліджуваних об’єктів, перевірка достовірності гіпотез, а також широке та глибоке вивчення досліджуваної наукової тематики. У рамках сучасної науки існує багато різних класифікацій експериментів в залежності від галузі науки, мети дослідження, структури об’єктів та явищ, організаційних заходів, характеру взаємодії об’єкту та засобів дослідження тощо.

Провідним місцем у досконалому проведенні експерименту, займає правильна розробка методики експерименту – визначена послідовність процесів, у результаті якої досягається мета дослідження.

Першочерговим етапом проведення експериментального дослідження є план програми дослідження, який складається за умови проведення дослідження, де визначається:

– гіпотеза;

– мета та задачі;

– вхідні і вихідні параметри, область їх визначення та крок дискредитації;

– порядок проведення власне експерименту;

– зазначаються необхідні засоби проведення дослідження, моделювання, обробки результатів;

– порядок і вимоги щодо оформлення результатів.

Далі, слідує етап визначення об’єму експериментальних досліджень та необхідних програмних та апаратних засобів тощо.

Останнім кроком є безпосередньо експеримент, який проводиться з регламентацією всіх кроків, та обробка і систематизація експериментальних і усіх числових даних, перевірка зведення до єдиної системи одиниць, побудова графіків залежностей, таблиць, діаграм тощо.

Гіпотеза

Експеримент базується на припущені, що запропонований програмний модуль автоматизованого тестування веб-орієнтованої інформаційної системи адекватно реагує на зміну ідентифікуючих та оціночних параметрів при різних умовах контрольованого середовища.

Мета та задачі експерименту

Метою експерименту є перевірка адекватності запропонованого програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи, а саме:

– дослідження запропонованого програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи стосовно ефективності його роботи;

– дослідження запропонованого програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи стосовно ефективності його роботи, а саме коректності тестування.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– обробка і верифікація отриманих результатів;

– проведення виявлення недоліків та оцінки критичності ситуацій в тестованому середовищі при зміні ідентифікуючих та оціночних параметрів;

– визначення можливості використання запропонованого програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи для реалізації автоматизованого тестування.

Потрібно відокремити показники ефективності системи автоматизованого тестування.

4.1.2 Перелік показників ефективності системи автоматизованого тестування

Вибір вхідних та вихідних параметрів

Для запропонованого програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи

вхідні параметри – значення параметру, КВ1, Пар23, КВ2, ..., Пар Н, КВ Н, а також порівняльні судження експертів, щодо важливості оціночних е -го та е'-го параметрів між собою відповідно до методу кількісного парного порівняння з визначенням квадратного кореня;

– вихідні параметри – показник рівня якості веб-орієнтованої інформаційної системи.

Вибір кроку зміни вхідних параметрів. Значення параметрів КВ1, ПР2,3, КВ2, ..., ПР Н, КВ Н для обробки в системі автоматизованого тестування веб-орієнтованої інформаційної системи параметри ПР1, ПР2, ..., ПР Н приймають значення від 0 до 1, попередньо нормуючись. Максимальні значення параметрів встановлюються експертом в залежності від галузі застосування, контрольованого середовища і виду інциденту, що є причиною поточної ситуації.

Крім того ідентифікуючі та оціночні параметри можуть бути описані у вигляді лінгвістичних змінних, наприклад, «низький» (Н), «середній» (С), «великий» (В) тощо.

Послідовність дій в експериментальному досліджені

Для дослідження системи автоматизованого тестування веб-орієнтованої інформаційної системи виконуються в повній відповідності до етапів методу оцінки критичності ситуації та режиму роботи системи – задається множина оціночних параметрів; визначаються КВ важливості кожного параметра; проводиться фазифікація їх поточних значень; обчислюється показник рівня критичності ситуації; отриманий показник

порівнюється з оціночним еталоном, на основі чого приймається рішення щодо критичності ситуації; проводиться дефазифікації значень оціночних параметрів та рівня критичності і на основі отриманих даних створюється індикатор критичності.

Засоби проведення експерименту

Для дослідження, створення необхідного програмного забезпечення, імітаційного модулювання, обробки результатів та представлення їх в табличному та графічному вигляді використовувалося середовище Microsoft Excel 2007.

Аналіз результатів

Аналіз результатів імітаційного моделювання буде представлено у підрозділі 4.2 та 4.3 даної роботи. Результати представлені в табличній формі та у вигляді графіків і діаграм. Проведемо тестування ефективності системи автоматизованого тестування веб-орієнтованої інформаційної системи.

4.2 Тестування ефективності системи автоматизованого тестування веб-орієнтованої інформаційної системи

Використовуючи БД з показниками функціонування запропонованого програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи сформуємо табл. 4.1.

Таблиця 4.1 – Значення показників функціонування запропонованого програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи

Параметр	Значення									
$ПР_1$	1	0	1	1	0	1	1	1	0	1
$ПР_2$	1	0	1	1	0	0	0	1	0	1
$ПР_3$	1	1	1	1	1	1	1	1	1	1
$ПР_4$	1	1	0	0	0	1	0	1	1	0

$КВ_1$	1	2	3	1	3	2	4	1	3	2
$КВ_2$	3	2	4	1	3	2	2	3	4	4
$КВ_3$	2	5	8	8	2	1	5	2	3	5
$КВ_4$	1	2	3	1	3	2	4	1	3	2
$КВ_5$	2	5	8	8	2	5	8	8	7	7
ЗО	9	9	10	10	8	10	8	10	10	9

Відповідно до наведених даних здійснюємо аналіз отриманих даних. Розраховуємо середньоквадратичне відхилення та дисперсію, отримані результати наводимо у таблиці 4.2.

Таблиця 4.2 – Результати математичного аналізу

Параметр	Значення										Середнє відхилення	Дисперсія
$ПР_1$	1	0	1	1	0	1	1	1	0	1	0,42	0,21
$ПР_2$	1	0	1	1	0	0	0	1	0	1	0,5	0,25
$ПР_3$	1	1	1	1	1	1	1	1	1	1	0	0
$ПР_4$	1	1	0	0	0	1	0	1	1	0	0,5	0,25
$КВ_1$	1	2	3	1	3	2	4	1	3	2	0,84	0,96
$КВ_2$	3	2	4	1	3	2	2	3	4	4	0,84	0,96
$КВ_3$	2	5	8	8	2	1	5	2	3	5	2,1	5,69
$КВ_4$	1	2	3	1	3	2	4	1	3	2	0,84	0,96
$КВ_5$	2	5	8	8	2	5	8	8	7	7	2	5,2
ЗО	13	18	29	22	14	15	25	19	22	23	4,2	23,8

Наступним кроком є кореляційний аналіз. Результати у таблиці 4.3.

Таблиця 4.3 – Кореляційний аналіз отриманих результатів

	$ПР_1$	$ПР_2$	$ПР_3$	$ПР_4$	$КВ_1$	$КВ_2$	$КВ_3$	$КВ_4$	$КВ_5$	ЗО
$ПР_1$	1									
$ПР_2$	0,846564	1								
$ПР_3$	0,721958	0,949874	1							
$ПР_4$	0,696756	0,930106	0,953304	1						
$КВ_1$	0,888679	0,845974	0,749418	0,626177	1					
$КВ_2$	0,908195	0,883933	0,770116	0,74774	0,87989	1				
$КВ_3$	0,612761	0,89553	0,927541	0,869699	0,743616	0,80344	1			
$КВ_4$	0,825308	0,872774	0,831848	0,83447	0,730714	0,936313	0,824554	1		
$КВ_5$	0,835144	0,930264	0,886976	0,806971	0,881455	0,943425	0,904059	0,937171	1	
ЗО	0,805695	0,945124	0,972585	0,924287	0,789531	0,860236	0,907963	0,92644	0,94391	1

4.3 Аналіз отриманих результатів

Аналізуючи дані з табл. 4.3 та використовуючи шкалу Чеддока, можна зробити висновок, що найбільш впливають на ефективність такі чинники:

- $ПР_2$,
- $КВ_1$
- $КВ_2$
- $КВ_4$
- $КВ_5$
- ЗО

На виході цього етапу, згідно до формули:

$$KPI = \left\{ \bigcup_{w=1}^{v} KPI_w \right\} = \{KPI_1, KPI_2, \dots, KPI_v\}$$

де

$$KPI_w \subseteq KPI, (w = \overline{1, v})$$

v – кількість ключових показників ефективності.

при w = 6 отримуємо множину ключових показників ефективності KPI :

$$KPI = \{ПР2, КВ1, КВ2, КВ4, КВ5, ЗО\}$$

Розрахуємо загальну ефективність системи. Показник ефективності:

$$E = {Оцінка}/{max} \cdot 100\%$$

Для цього сформуємо таблицю максимальних показників (табл.4.4)

Таблиця 4.4 – Максимальні значення показників

Параметр	Значення
$ПР_1$	1
$ПР_2$	1
$ПР_3$	1
$ПР_4$	1
$КВ_1$	5
$КВ_2$	5
$КВ_3$	10
$КВ_4$	5
$КВ_5$	10
ЗО	39

Наступним кроком розрахуємо ефективності системи та результати зведемо до таблиці 4.5.

Таблиця 4.5 – Результати розрахунку ефективності

Параметр	Максимальне значення	Оцінка	Ефективність
$ПР_1$	1	0,7	70
$ПР_2$	1	0,5	50
$ПР_3$	1	1	100
$ПР_4$	1	0,5	50
$КВ_1$	5	2,2	44
$КВ_2$	5	2,8	56
$КВ_3$	10	4,1	41
$КВ_4$	5	2,2	44
$КВ_5$	10	6	60
ЗО	39	20	51,28205

Наведемо графічно результати розрахунку:

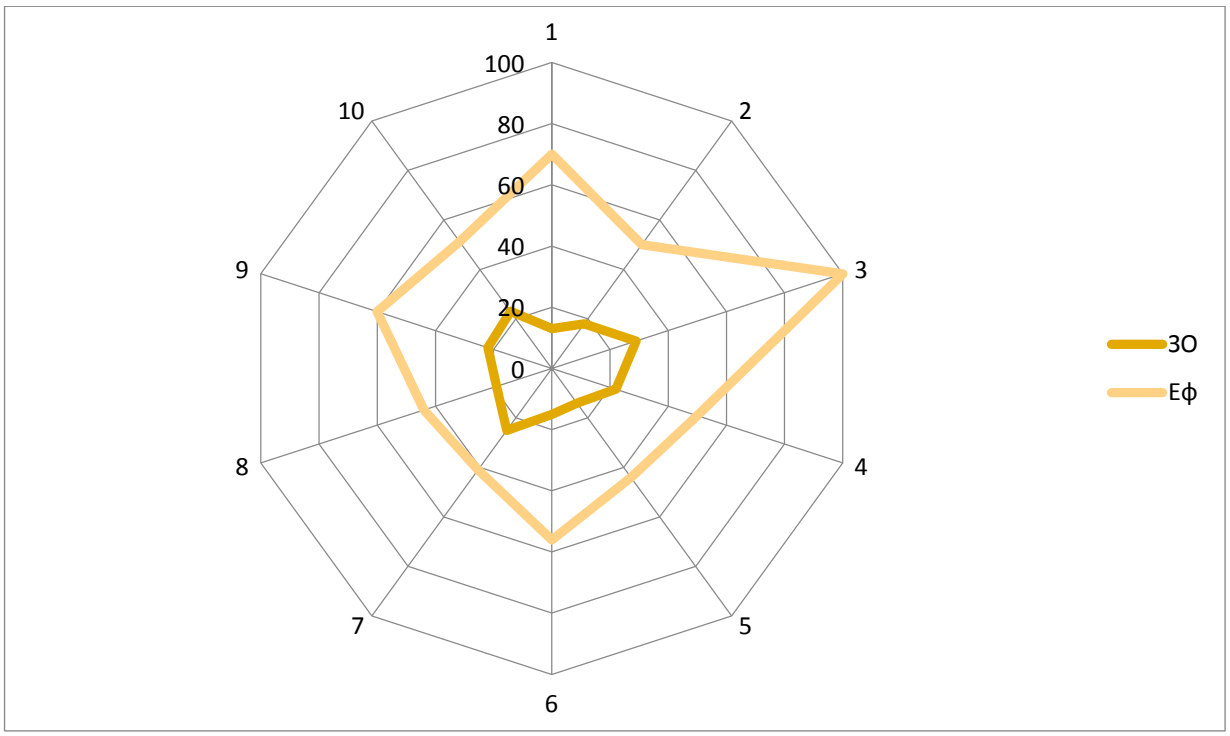

Рисунок 4.1 – Графік залежності ефективності від Загальної оцінки

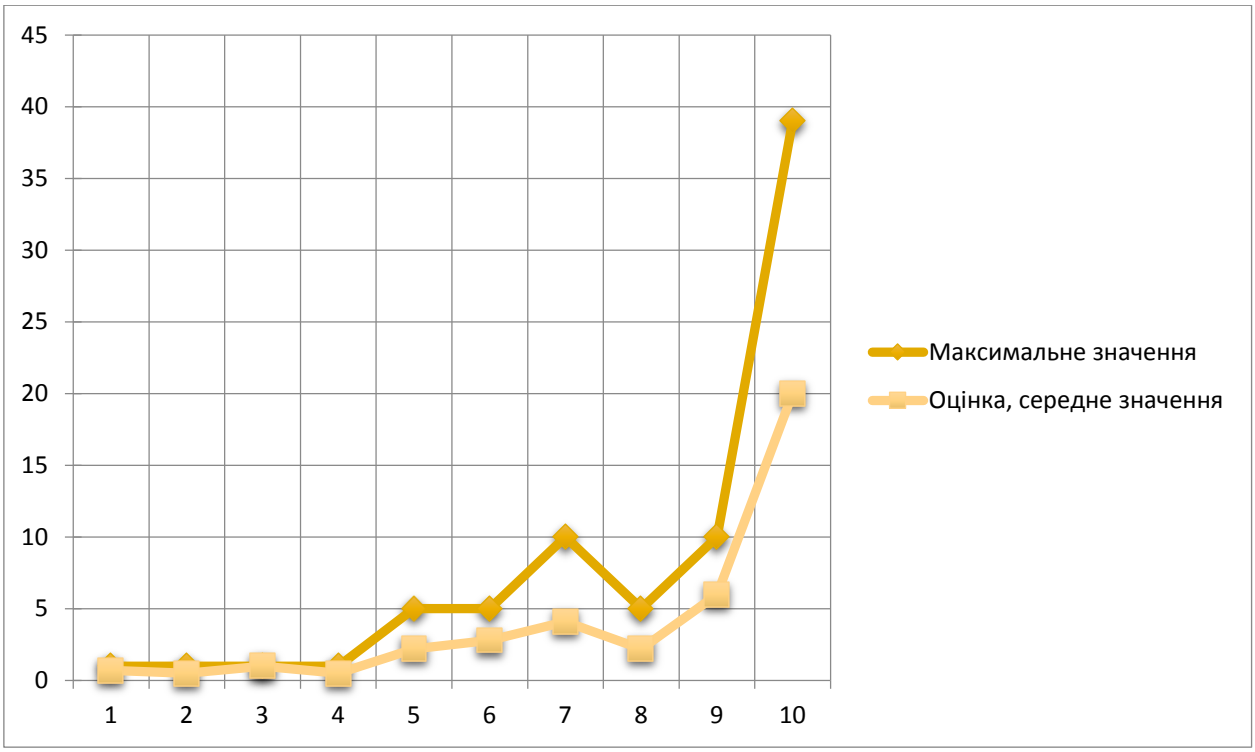

Рисунок 4.2 – Графік залежності ефективності від середньої оцінки

Провівши аналіз отриманих результатів (рис. 4.1 та 4.2), можна зробити висновок про залежність ефективності від кожного із визначених показників.

ВИСНОВКИ

У межах даної бакалаврської роботи проведено розробку програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи.

Найбільш ефективним способом контролю якості програмного продукту є здійснення процесу тестування, що розробляється протягом усього життєвого циклу розробки. Основною перевагою застосування автоматизації тестування є економія часу, що пов'язано з високою швидкістю виконання тестів, а також можливість виконувати їх неодноразово. Для автоматизації тестування існує безліч інструментів, засобів і рішень. Автоматизація в цілому не тільки дозволяє заощадити час на розробку, але збільшує надійність і безпеку створюваних продуктів.

За підсумками проведеної роботи можна зробити висновки:

1 Розглянуто поняття якості програмного продукту, роль тестування в процесі, його рівні, цілі і завдання. Проведено оцінку вартості помилки на різних стадіях розробки програмного забезпечення. Проаналізовано основні принципи автоматизації тестування. Зроблено оцінку ефективності використання автоматизації на проекті.

2 Розглянуто рівні автоматизації тестування. Проведено огляд існуючих інструментів для автоматизації тестування і описаний основний процес створення програмного модулю автоматизованого тестування веб-орієнтованої інформаційної системи.

3 Розроблено програмний модуль автоматизованого тестування веб-орієнтованої інформаційної системи, що дозволяє здійснювати автоматизоване тестування різних веб-додатків.

Розроблений програмний модуль є корисним засобом для розробки тестових скриптів не тільки для кваліфікованих тестувальників, але і для будь-якого іншого користувача, якому необхідно стежити за якістю свого застосування, навіть якщо він не має спеціальних навичок в тестуванні веб

додатків. Програмний модуль автоматизованого тестування веб-орієнтованої інформаційної системи є логічно завершений програмний продукт і повністю готовий до роботи.