МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 2017 р.

БАКАЛАВРСЬКА РОБОТА

НА ТЕМУ:

Методи та засоби автоматизації тестування веб-додатків

.

Освітньо-кваліфікаційний рівень "Бакалавр"

Науковий керівник роботи:	_____ (підпис)	Нестеров М. В. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я. О. (ініціали, прізвище)
Студент:	_____ (підпис)	Ковальова А. О. (ініціали, прізвище)
Група:		КІ-13аД

Сєвєродонецьк 2017МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерної інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 Комп'ютерна інженеія
(шифр і назва)
Спеціальність 6.050102 Комп'ютерна інженеія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Ковальова Анастасія Олександрівна
(прізвище, ім'я, по батькові)

1. Тема роботи Методи та засоби автоматизації тестування веб-додатків

керівник проекту (роботи) Нестеров Максим Володимирович
(прізвище, ім'я, по батькові, науковий ступінь)
затверджені наказом вищого навчального закладу від "15" 05 2017р. № 124/48
2. Строк подання студентом роботи
3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні слайди

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада	Підпис, дата

	консультанта	завдання видав	завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка

Студент _____ Ковальова А. О.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Нестеров М. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту (роботи) бакалавра містить: 136 сторінки, 7 рисунків, 17 таблиць, 25 бібліографічних джерел, 3 додатки.

Об’єкт аналізу: аналіз методів та засобів автоматизіціїї тестування.

Мета роботи: метою роботи є розкриття змісту та призначення тестування прграмного забезпечення, проведення аналізу готових рішеннь для автоматизації цього процесу та розглянення па прикладі роботи сценаріїв тестування.

В проекті виконано:

1) Розкрито значення тестування програмного забезпечення під час розробки.

2) Виділено методи та класи тесттування ПЗ;

3) На основі методів та класів вибрано приклад додатку для автоматизаціїї тестування з задовільним функціоналом.

4) Проведено аналіз додатків для автоматизації.

5) Зроблено висновки.

Результатом роботи є інформація на основі якої можно вибирати для користуванння додатки для автоматизування тестування ПЗ, пояснення та розкриття суті тестування тта його принципівю

Ключові слова: ТЕСТУВАННЯ, ПРОКРАМНЕ ЗАБЕЗПЕЧЕННЯ, ВЕБ – ДОДАТКИ, ТЕСТ – КЕЙС,

ВСТУП

ВСТУП

У сучасному житті немає людини яка не використовує ПО. Програми вже давно стали невід'ємною частиною нашого життя і використовуються у різних його сферах. Часто буває так, що програма раптово закривається або має якісь помилки, що називаются багами. Саме ці баги і повинні знаходитися під час тестування програмного продукту.

Основними причинами прояву помилок ПЗ є недостатньо високий рівень технології виробництва програмних засобів і їхня надмірна складність. І незважаючи на те, що в області якості й надійності програмних засобів останнім часом досягнуті певні позитивні результати й помилки в процесі функціонування ПЗ порівняно рідкі, проблема забезпечення високої надійності складного ПЗ залишається досить важливою. Добре відомо парадоксальне протиріччя: чим надійніше продукт ми створюємо, тим важче оцінити й підтвердити його надійність.

Багато організацій, що займаються створенням програмного забезпечення, до 30% засобів, виділених на розробку програм, витрачають на випробування. І все-таки, незважаючи на величезні капіталовкладення, знань про суть випробувань явно не вистачає й більшість програмних продуктів ненадійна.

Під випробуванням програмної продукції варто розуміти експериментальне визначення кількісних і/або якісних характеристик властивостей продукції при її функціонуванні в реальному середовищі й/або моделюванні середовища функціонування. Неможливо гарантувати відсутність помилок у програмі; у кращому разі слід показати наявність помилок. Якщо програма правильно поводиться для солідного набору тестів, немає підстав стверджувати, що в ній немає помилок; з усією визначеністю можна лише стверджувати, що не відомо, коли ця програма не працює. Надійність неможливо внести в програму в результаті тестування, вона визначається правильністю етапів проектування. Найкраще рішення проблеми надійності - із самого початку не допускати помилок

у програмі. Однак імовірність того, що вдасться бездоганно спроектувати велику програму, нескінченно мала.

Випробування таких програм, як системи реального часу, операційні системи й програми керування даними особливо важкі. Треба було б тестувати програму не тільки для кожного вхідного значення, але й для кожної послідовності, кожної комбінації вхідних даних. Тому вичерпне тестування для всіх вхідних даних будь-якої розумної програми нездійсненно. Випробування є завершальним етапом розробки. Його випереджує етап статичного й динамічного налагодження програм. Основним методом динамічного налагодження є тестування. У вузькому змісті ціль тестування складається у виявленні помилок, мета ж налагодження - ні тільки у виявленні, але й в усуненні помилок. Однак обмежитися тільки налагодженням програми, якщо є впевненість у тім, що всі помилки в ній усунуті, не можна. Мета в налагодження й випробування різна. Повністю налагоджена програма може не мати певні споживчі властивості й тим самим бути непридатної до використання по своєму призначенню. Не може служити альтернативою випробуванню й перевірка працездатності програми на контрольному прикладі, тому що програма, працездатна в умовах контрольного приклада, може виявитися непрацездатної в інших умовах застосування.

1 АНАЛІЗ МЕТОДІВ ОЦІНКИ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Одним з основних методів оцінки якості програмного забезпечення (ПЗ) є тестування. Тестування ПЗ (Software testing) – це діяльність, виконувана для оцінки й поліпшення якості програмного забезпечення. Ця діяльність, у загальному випадку, базується на виявленні дефектів і проблем у програмних системах [1]. Одна із ключових проблем криється в правильному визначенні поняття тестування, тому що це далеко неоднозначне завдання. Для того, щоб переконатися в цьому, слід звернутися до міркувань основоположника теорії тестування Гленфорда Майерса (Glenford Myers). Але, насамперед, слід привести два головних закони теорії тестування програмних продуктів:

1) Неможливо відшукати абсолютно всі помилки в програмному продукті. Помилки залишаються завжди.

2) Побудова вичерпного вхідного тесту неможлива. Іншими словами, неможливо повністю протестувати програму: навіть для самої найпростішої системи це займе настільки велику кількість часу, що ніколи не буде заповнено вигодою від виробництва «ідеально протестованої» програми.

При тестуванні сучасних складних програмних систем вичерпне тестування стає нездійсненним завданням. Далі Майерс дає власне визначення тестування: “Тестування ПЗ – це процес виконання програми з метою виявлення помилок. Майерс вважає тест вдалим, якщо в процесі його виконання були виявлені помилки. Саме в цьому й складається завдання тестування”.

Варто відзначити, що визначення тестування, дане Майерсом, на сьогоднішній день є застарілим.

Тестування ПЗ (Software testing) – це процес аналізу й експлуатації програмного забезпечення з метою виявлення дефектів. Під дефектом розуміється невиконання вимоги, пов'язаної з передбачуваним або встановленим використанням. У визначенні варто звернути увагу на ключове слово «процес».

Тестування – це планова й упорядкована діяльність. Цей момент дуже важливий, оскільки в умовах, найчастіше дуже обмеженого часу виділеного на розробку й тестування, добре продуманий і систематичний підхід швидше приводить до виявлення програмних помилок, чим погано сплановане тестування, проведене в поспіху.

Визначення тестування, приведене Майерсом, варто пам'ятати при виконанні тестування. Але, все-таки розуміти сучасне тестування треба трохи ширше. Тому при описі технології тестування слід дотримуватися більше офіційного визначення тестування, наведеного в міжнародних стандартах по тестуванню.

В 1990 році стандартом ISO (International Organization for Standardization) прийняте наступне визначення тестування:

Тестування — це спостереження за функціонуванням ПЗ в специфічних умовах з метою визначення ступеня відповідності ПЗ вимогам до нього.

При переході до сучасних методологій розробки ПЗ спостерігається більш системний підхід у визначенні тестування ПЗ. Тестування орієнтоване, у першу чергу, на оцінку якості за допомогою наступних методів:

- пошук і документування дефектів якості;
- загальні рекомендації щодо якості;
- перевірка виконання основних припущень і вимог на конкретних прикладах;
- перевірка, що продукт функціонує так, як було запроектовано;
- перевірка, що вимоги виконані відповідним чином.

1.1 Модель тестування ПЗ на основі узагальненої моделі життєвого циклу ПЗ

Узагальнена модель життєвого циклу тестування ПЗ (ЖЦ ТПЗ) [2]може бути представлена у вигляді “V”-образної моделі, як показано на рисунку 1.1. Це один зі способів продемонструвати, як співвідносяться процеси тестування з основними процесами проектування й розробки. В “V”-моделі основні етапи ЖЦ ПЗ утворять ліву сторону “V”, кодування перебуває в самій нижній крапці діаграми, а тестування утворить її праву сторону .

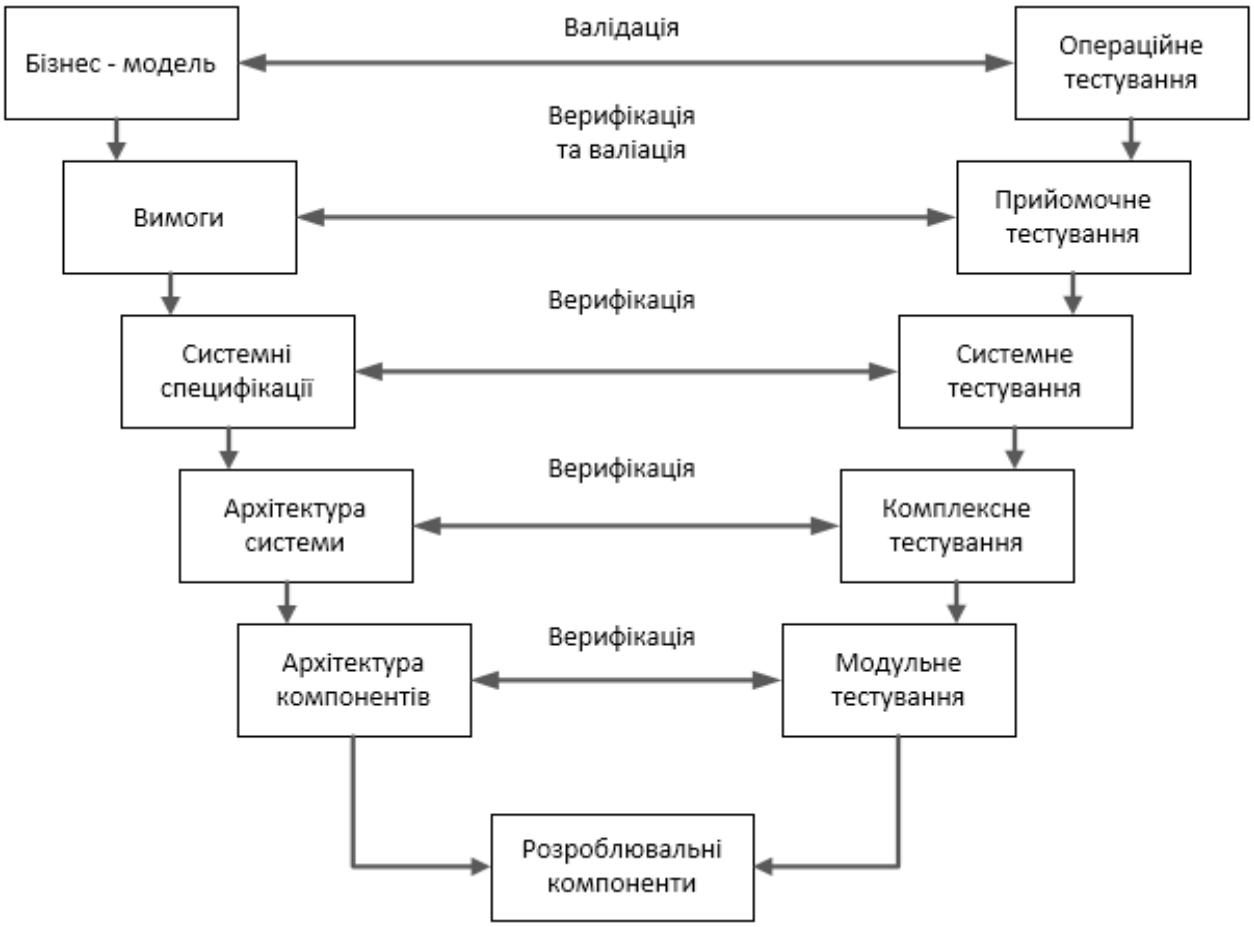


Рисунок 1.1 - “V”-образна модель узагальненого ЖЦ ТПЗ

“V”-модель показує основні етапи тестування ПЗ і їхній зв'язок з етапами розробки ПЗ. Узагальнена модель ЖЦ тестування ПЗ може бути перетворена до виду конкретної моделі ЖЦ ПЗ.

Таким чином, на основі сучасних методологій, стандартів якості й стандартів розробки ПЗ можна систематизувати загальне розуміння тестування й визначити узагальнену модель ЖЦ ТПЗ, що має наступні якості:

– ЖЦ ТПЗ систематично пов'язана із ЖЦ ПЗ;

– ЖЦ ТПЗ закриває всі рівні перевірки якості ПЗ

– ЖЦ ТПЗ використається для валідації та верифікації, які є вкрай важливими процесами по забезпеченню якості ПЗ; навіть назва узагальненої моделі ЖЦ ТПЗ “V-модель” відбиває свій головний зміст – Верифікація й Валіація (Verification and Validation) якості ПЗ, як невід'ємна складова будь-якої сучасної розробки;

– ЖЦ ТПЗ дозволяє зрозуміти, що, як і для чого повинне бути піддане тестуванню в кожній крапці ЖЦ ПЗ;

– Узагальнена модель ЖЦ ТПЗ може бути перетворена під будь-яку модель ЖЦ ПЗ.

Як вже визначалося, тестування в праві називатися самостійною дисципліною, що має свої мети, завдання, ролі, види, стратегії, методи, критерії, свою методологію й технологію.

1.2 Цикли та основні артефакти тестування

Узагальнена модель ЖЦ ТПЗ може здобувати ітеративну (багаторазову) природу. Тестування звичайно проводиться циклами, кожний з яких має конкретний список завдань і цілей. Тому можна зробити висновок про подвійну циклічність процесу тестування, якщо розробка ведеться по ітеративній моделі ЖЦ ПЗ. Можна виділити два види циклів тестування: загальний цикл тестування й приватний цикл тестування [3].

Приватний цикл тестування включає вибір фрагментів і формулювання завдань тестування (наприклад, визначити придатність архітектури, перевірити

реалізацію основної функціональності конкретного сценарію використання або перевірити виконання вимог Замовника в повному обсязі.

Також приватний цикл тестування дозволяє підтвердити правильність зборки. [3].

На цьому етапі також розробляються необхідні тести, після чого тести виконуються в ручному або автоматичному режимі, і проводиться оцінка результатів. Нижче, на рисунку 1.2 наведені короткі описи завдань, що входять у приватний цикл тестування, проведений для окремої зборки об'єкта.

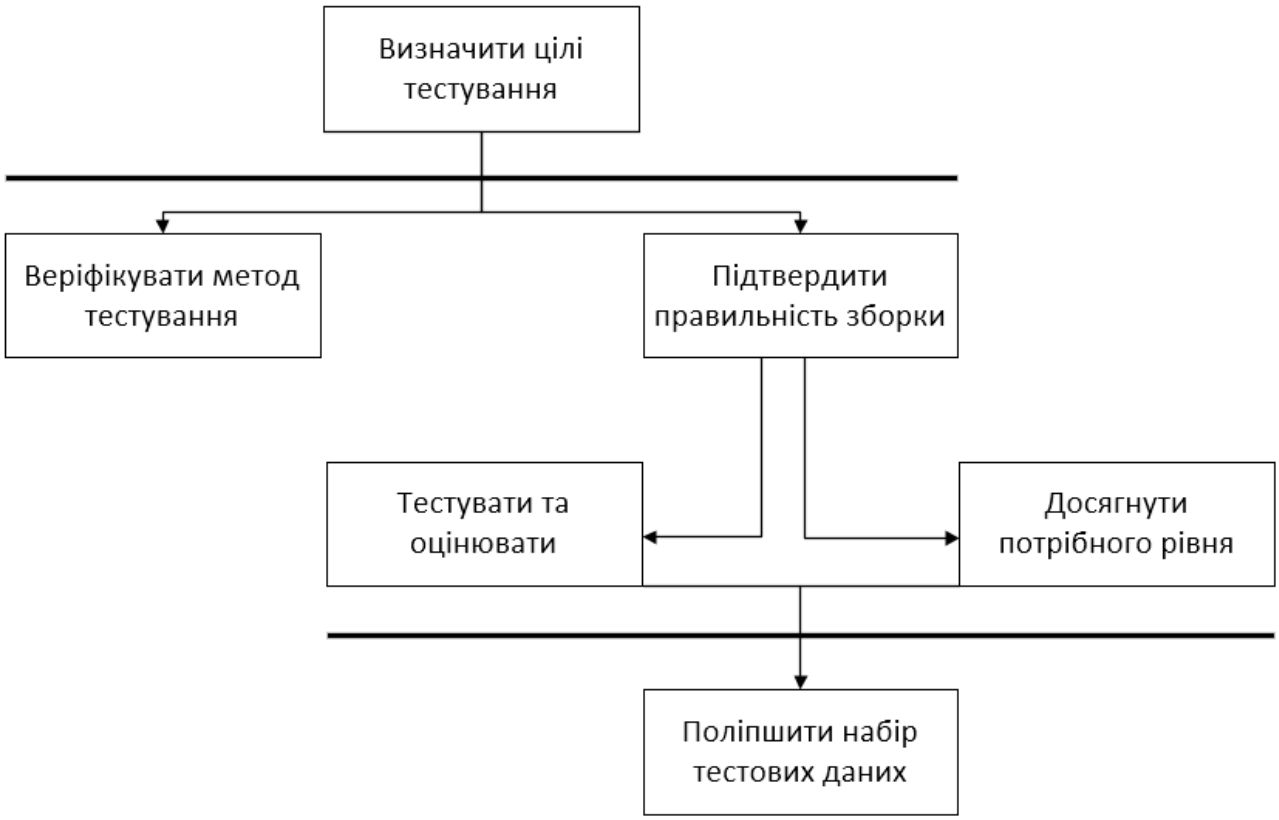


Рисунок 1.2 - Приватний цикл тестування, проведений для окремої зборки об'єкта тестування

Виконання завдань життєвого циклу супроводжується розробкою різних артефактів (документів, моделей і інших матеріалів проекту). Розробка артефактів може проводитися в різній формі з різними вимогами до способу виконання і якості оформлення.

Перш ніж описувати повний цикл тестування, необхідно визначити основні артефакти цього процесу. Нижче представлені основні робочі артефакти тестування, у тій або іншій формі зв'язані зі сценаріями використання:

– План тестування – це основний документ, що визначає стратегію тестування на кожній ітерації. У нього входять опис цілей і завдань тестування в поточній ітерації, переліки тестів, які повинні використовуватися, формовані метрики й критерії початку й завершення тестування.

– Сценарій тестування (тест-кейс) - це один з основних документів, з якими має справа інженер з якості програмного забезпечення. По суті, спрощений опис тесту. Тобто вхідної інформації, умов і послідовності виконання дій і очікуваного вихідного результату.

– Тестові дані. Вони покликані визначати набори вхідних даних для тестів, а також очікувані результати, з якими отримані результати виконання тестів повинні рівнятися як з еталонними.

– Тестовий скрипт. Звичайно говорять про програмну реалізацію тесту, хоча скрипт може описувати й ручні дії, необхідні для виконання конкретний тест кейса.

– Набір тестів. Як правило, сценарії тестування поєднуються в пакети або набори. По-перше, це просто спосіб групування тестів з подібними завданнями, а, по-друге, у такий набір можна включати залежні тести, які повинні виконуватися в певному порядку (оскільки наступні тести використовують дані, сформовані в ході виконання попередніх).

– Результати тестування. Являють собою сумарну інформацію про проходження тестів, на основі аналізу яких і порівняння з очікуваними результатами виконується детальна оцінка якості програмного продукту й поточного статусу процесу тестування.

– Дефекти. Основні артефакти процесу тестування - описують виявлені факти невідповідності системи пропонованим вимогам. Хоча базу даних дефектів можна вести в текстовому файлі або Excel таблиці, кращим є використання спеціалізованого інструментального засобу, що дозволяє передавати інформацію про виявлені дефекти від інженерів з якості програмного забезпечення до програмістів, а у зворотну сторону - відомості про усунення дефектів та також формувати необхідні звіти про тенденції зміни кількості що виявляють і дефектів,

що усувають. Системи обліку дефектів будуть обов’язково детально розглянуті у третьому розділі даного дипломного проекту.

Виходячи з вище сказаного можна сформувати основні задачі загального циклу тестування:

1) Планування тестів:

– визначення вимог до тестів;
– оцінка ризиків;
– розробка стратегії тестування;
– визначення ресурсів;
– створення розкладу/послідовностей;
– розробка плану тестування;

2) Дизайн тестів:

– аналіз обсягу робіт;
– визначення й опис тестових випадків;
– визначення й структурування тестових процедур;
– огляд і оцінка тестового покриття;

3) Розробка тестів:

– запис або програмування тестових скриптів;
– визначення тест - критичної функціональності в дизайні й моделі реалізації;
– створення/підготовка зовнішніх наборів даних;

4) Виконання тестів:

– виконання тестових процедур;
– оцінка виконання тестів;
– відновлення після збійних тестів;
– перевірка результатів;
– дослідження несподіваних результатів;
– запис помилок;

5) Оцінка тестів:

– оцінка покриття тестовими випадками;
– оцінка покриття коду;
– аналіз дефектів;
– визначення критеріїв завершення й успішності тестування.

На основі перерахованих завдань і активностей можна визначити повний цикл активностей тестування, наведений на рис. 1.3.

Рисунок 1.3 - Повний цикл тестування, що визначає основні активності фахівців з тестування

Таким чином, крім уже певної ітеративності “V”-моделі ЖЦ ТПЗ, вона здобуває подвійну циклічність за рахунок того, що загальні й/або приватні цикли тестування можуть відбуватися кінцеве число раз у межах ітерації.

1.3 Стратегії тестування

Розходження завдань і цілей тестування протягом життєвого циклу продукту приводить до необхідності розробляти й реалізовувати різні стратегії тестування. Кожна така стратегія визначає [4]:

1) ітерації, на яких використаються стратегія тестування й мета тестування на кожній ітерації;

2) стадії тестування для кожної ітерації;

3) критерій успішного завершення тестування;

4) типи використовуваних тестів;

5) набір методів і інструментальних засобів, необхідних для проведення тестування й оцінки якості;

6) критерії оцінки тестів.

Як вже визначалося вище, стратегії тестування повинні розроблятися при плануванні тестування.

1.4 Метрики й критерії тестування

Ще одним важливим поняттям у теорії тестування є поняття критеріїв покриття тестування. Останні дозволяють визначити ступінь покриття розроблювального продукту тестами. Тому часто критерії покриття використовуються для визначення метрик тестування.

При тестуванні функціональних вимог можуть бути виділені, принаймні, два типи покриття: покриття, засноване на специфікації, і покриття, засноване на коді [4].

1) Покриття, засноване на специфікації або на вимогах (Specification-Based Coverage or Requirements-based Test Coverage) - цей критерій оцінює ступінь покриття, беручи до уваги вимоги Замовника або системних специфікацій. Набір тестів повинен покривати всі або конкретно певні функціональні вимоги. Замовник (або системний аналітик) становить набір вимог, які можуть бути переведені в тестові сценарії. Таким чином, даний критерій показує у відсотковому відношенні кількість покритих тестами вимог. Найчастіше даний критерій використовується при тестуванні методом «чорного ящика».

2) Покриття, засноване на коді (Code-Based Coverage) - має відношення до потоку керування й потоку даних програми. Найчастіше даний критерій використається при тестуванні методом «білого ящика». Основні критерії покриття тестування коду наступні:

– Покриття рядків (Line Coverage) - міра виміру покриття коду, що вказує процентне відношення рядків програми, порушених тестами, до загального числа рядків.

– Покриття відгалужень (Branch Coverage). Це міра виміру покриття коду вказує у відсотковому відношенні, скільки відгалужень потоку керування було протестовано під час тесту.

– Покриття шляхів (Path Coverage). Ця одиниця виміру характеризує відсоток усіляких шляхів (і/або комбінацій відгалужень), які покриваються тестами.

Метрики й критерії тестування визначаються в стратегії тестування поряд з іншими складовими процесу.

1.5 Статичне й динамічне тестування

Технологій тестування існує ціла безліч. Умовно їх можна віднести до статичного або до динамічного [4].

Статичне тестування – це процес, що звичайно асоціюють із аналізом ПЗ. Статичним тестуванням користуються для верифікації практично будь-якого артефакту розробки: програмного коду компонент, вимог, системних специфікацій, функціональних специфікацій, документів проектування й архітектури програмних систем і їхніх компонентів, і т.д. Використання статичних методів тестування - один з найбільш ефективних способів виявлення дефектів на ранніх стадіях розробки ПЗ. Дійсно, статичне тестування - це єдиний спосіб тестування без запуску програмного коду.

Динамічне тестування – це процес тестування над працюючою системою або підсистемою. Воно не може бути здійснене без запуску програмного коду. Якщо бути більш точним, динамічне тестування складається з:

– запуску системи або підсистеми;

– виклику необхідних функціональних елементів або модулів;

– порівняння через графічний інтерфейс користувача поводження системи з очікуваним результатом поводження.

Динамічність (dynamic) - це термін, який має на увазі тестування, що припускає виконання програми із заданими вхідними даними. При цьому, величини, що задають на програмному забезпеченні, не завжди достатньо для визначення тесту. Складність і недетермінованість систем приводить до того, що система може по різному реагувати на ті самі вхідні параметри, залежно від стану системи.

1.6 Класифікація типів та методів тестування

Тести істотно розрізняються по завданнях, які з їхньою допомогою вирішуються, і по використовуваній техніці. Розходження завдань тестування приводить, природно, до необхідності використати досить різноманітні типи тестування. Прийняте підрозділяти тестування по наступних категоріях:

1) По знанню внутрішнього пристрою системи

- Білий ящик (White box testing)

Це метод тестування, який вивчає внутрішній пристрій системи. Такі методики узагальнено називають тестуванням "білого ящика" [5].

При тестуванні системи методом «білого ящика» відбувається перевірка логіки. Повним тестуванням у цьому випадку буде таке, котре приведе до перебору всіх можливих шляхів. Навіть для середніх по складності програм числом таких шляхів може досягати десятків тисяч.

На рівні окремих блоків перевіряється, чи не відбудеться крах всієї системи при передачі у функцію неочікуваних нею параметрів. Після такої перевірки можна бути впевненим у тім, що помилок, що приводять до краху системи, немає й що функції будуть стійко працювати в будь-яких умовах (тобто видавати передбачувані результати навіть при уведенні непередбачених вхідних параметрів).

Ціль тестових наборів для "білого ящика" виявити всі приховані дефекти шляхом всебічного тестування функції різноманітними вхідними параметрами.

Щоб створити ефективні тестові набори для "білого ящика", необхідно спочатку одержати повне подання про внутрішню структуру функції, написати набори, що забезпечують максимальне покриття функції, і знайти сукупність входів, що приводять до відмови функції.

- Чорний ящик (Black box testing)

Довгий час основним способом тестування було тестування методом "чорного ящика". Програмі подавалися деякі дані на вхід і перевірялися результати, у надії знайти невідповідності. При цьому як саме працює програма вважається несуттєвим. Навіть при такому підході необхідно мати специфікацію програми для того, щоб було із чим порівнювати результати.

Тестування "чорний ящик" - це пошук відсутніх або неправильно виконуваних функцій з метою оцінки, наскільки добре програма відповідає вимогам. Функціональні тести звичайно підтверджують правильність даних на введенні й виведенні. Іноді таке тестування називають користувальницьким. Тест функціональності полягає в тому, щоб перевірити правильне виконання окремих функцій системою.

Цей підхід дотепер є найпоширенішим у повсякденній практиці, але в нього є цілий ряд недоліків. По-перше, таким способом неможливо знайти взаємознищуємих помилок, по-друге, деякі помилки виникають досить рідко (помилки роботи з пам'яттю) і тому їх важко знайти й відтворити [6].

- Сірий ящик (Gray box testing)

Це змішана адаптивна методика, застосовувана досвідченими фахівцями з якості програмного забезпечення або розроблювачами при налагодженні. Аналіз програмного продукту з погляду виконання операцій кінцевим користувачем і контролю отриманих результатів із застосуванням знань про код й подробиць реалізації функцій продукту.

2) По об'єкту тестування [6]:

– Функціональне тестування (Functional testing) - перевірка відповідності функціональності продукту заявленим специфікаціям. Тест може бути як досить простим для перевірки базової функціональності, так і детальним для перевірки реалізації різних сценаріїв використання. Перевіряється відповідність очікуваних і реально отриманих результатів роботи програми ;

– Тестування інтерфейсу користувача (User Interface testing) - більша частина функціональності ПЗ реалізується, як правило, через користувальницький інтерфейс. Тестування інтерфейсу – це тестування функціональності візуальних форм (екранів, кнопок і т.п.) і перевірка на відповідність дизайну. Мета цього типу тестування - виявлення помилок в інтерфейсі й пошук помилок у функціональності за допомогою інтерфейсу

– Тестування локалізації (Localization testing) - перевіряє, чи правильно локалізований продукт. Тобто, переведений на іншу мову й коректно працює з обліком національних особливостей країни або регіону, у якому буде продаватися й використатися продукт.

– Операційне тестування (Release Testing) - навіть якщо система задовольняє всім вимогам, важливо переконатися в тім, що вона задовольняє потребам користувача й виконує свою роль у середовищі своєї експлуатації. Крім цього, тестування в середовищі експлуатації дозволяє виявити й нефункціональні проблеми, такі як: конфлікт із іншими системами, суміжними в програмних і електронних оточеннях; недостатня продуктивність системи в середовищі експлуатації та й ін.

Очевидно, що знаходження подібних речей на стадії впровадження - критична й

дорога проблема. Тому так важливе проведення не тільки верифікації, але й валідації, із самих ранніх етапів розробки ПЗ.

– Інсталяційне тестування (Installation testing) - у процесі інсталяційного тестування перевіряється коректність установки й деінсталяції програмного продукту в середовищі максимально наближеної до експлуатаційного. Перевірка правильності установки програмного продукту повинна бути обов'язковим елементом проекту по тестуванню будь-якого продукту. Основна мета полягає в тому, щоб переконатися, що продукт може бути встановлений / деінстальований при різних умовах - таких як: нова інсталяція, удосконалення системи (upgrade), установка за замовчуванням, повна установка, установка на вибір.

– Тестування інтернаціоналізації (Internationalization testing) - цей вид тестування візначає наскільки продукт готовий до того, щоб бути адаптованим для роботи в інших локальних зонах з іншою мовою користувальницького интервейса, відмінному від мови за замовчуванням (як правило, це англійський). Ціль тестування інтернаціоналізації - перевірити здатність продукту бути швидко локалізованим під необхідну локальну зону потенційних користувачів системи.

– Тестування швидкості, надійності та продуктивності (Performance testing) [7]:

– Тестування швидкості - це перевірка швидкості роботи системи (час відгуку, частота транзакцій і інші параметри системи, залежні від часу) в імітаційному та реальному середовищах.

– Тестування надійності – це перевірка поводження системи при тривалій безперервній експлуатації за умови високого навантаження системи.

Тестування продуктивності

Продуктивність є однією з найважливіших характеристик сучасних програмно-апаратних комплексів. Тестування продуктивності, з одного боку, дозволяє знайти вузькі місця й неоптимальні рішення в програмній системі й запропонувати методи їхнього виправлення. З іншого боку, тестування продуктивності може допомогти Замовникові підібрати необхідне встаткування, що забезпечує необхідні показники ефективності роботи системи в цілому.

Тестування продуктивності – це тести, що використовують постійний обсяг навантаження, але різні конфігурації системи й програмного оточення, а також для визначення прийнятності продуктивності системи і її настроювання (або оптимізації). Виміру підлягає кількість транзакцій у хвилину, число користувачів, розмір використовуваної бази даних і т.п.;

При тестуванні продуктивності проводяться наступні тести [7]:

а) Навантажувальне тестування (Load testing) - перевірка й визначення прийнятності діючих меж системи при різних обсягах навантаження, при цьому сама система залишається постійною. Виміряються характеристики обсягу навантаження й часу відповіді. Мета цього тестування – оцінити здатність системи правильно функціонувати при деякім перевищенні планованих навантажень при реальній експлуатації (система має деякий «запас міцності»);

б) Тестування розподілу й балансування навантаження - якщо системи містять розподілену архітектуру або балансування навантажень, проводяться спеціальні тести, для перевірки методів функцій розподілу й балансування навантажень;

в) Тестування конфліктів - перевірка можливості системи коректно обробляти численні запити користувачів до одного ресурсу (запису даних, пам'ять, і т.п.);

г) Тестування обсягів - перевірка й аналіз можливостей системи оперувати більшими обсягами даних. Перевіряється як робота з більшими обсягами даних на вході/виході системи, так і робота із внутрішніми даними системи;

д) Тестування на розширюваність - тести для виміру й аналізу швидкості виконання різних операцій продукту на безлічі конфігурацій програмного й апаратного забезпечення й систем керування базами даних.

е) Тестування мережного трафіка - аналіз ефективності мережного обміну між розподіленими компонентами систем. Аналізу піддається архітектура й технічні рішення, використані при реалізації комунікаційної підсистеми.

Юзабіліті тестування (**Usability testing**) [8]

З поняттям «юзабіліті» користувачі Інтернету познайомилися порівняно недавно - в 1998 році, коли провідний інженер компанії Sun Microsystems Якоб Нільсен створив разом з колишнім віце-президентом компанії Apple Дональдом Норманом компанію Nielsen Norman Group, що займається розробками в області веб-юзабіліті. Саме по собі англійське слово usability позначає «практичність», «легкість і простоту використання», а є й більш близьке за змістом слово «ергономіка».

«Юзабіліті» - це якісна ознака, що визначає зручність інтерфейсу й легкість у його використанні. Але є й більш об'єктивні складові цього поняття, які допомагають однозначно визначити якість виконання користувальницького інтерфейсу: навченість, ефективність; запам’ятовувальність, помилки, задоволеність.

Навченість — кількісний параметр, що позначає час, що витрачає непідготовлений користувач на навчання правилам користування інтерфейсом.

Ефективність — показник того, наскільки швидко й ефективно користувач може виконувати свої завдання, використовуючи інтерфейс.

Запам’ятовувальність — це фактор, що визначає те, наскільки швидко користувач зможе згадати принципи роботи із системою, якщо йому доведеться зштовхнутися з нею через певний проміжок часу.

Ще однією немаловажною ознакою юзабіліті буде кількість помилок, які користувач може зробити, використовуючи систему. Адже від них залежить і загальна ефективність роботи з користувальницьким інтерфейсом.

І останнім по переліку, але далеко не останнім по значимості параметром, можна вважати задоволеність користувача від використання інтерфейсу.

Юзабіліті є відмінним козирем у боротьбі з конкурентами. Як тільки з'явиться конкурент, інтерфейс сайту якого буде зручніше, відбудеться втрата більшої частини відвідувачів. Якоб Нільсен вважає, що сайти, творці яких вчасно не подбали про створення зручного користувальницького інтерфейсу, рано або пізно будуть забуті відвідувачами [8]. Якщо перша сторінка ресурсу відразу не пояснює користувачеві мету й завдання сайту, то більша частина людей покине її

й ніколи більше не повернеться. Принципи роботи із сайтом повинні бути зрозумілі користувачеві відразу, при першому відвідуванні, тому що ніхто не буде спочатку вивчати інструкцію, а потім визначати, потрібна йому ця інформація або ні. Цим простим і ефективним способом дослідження Юзабіліті є тестування [8].

Тестування сумісності (Конфігураційне тестування (Configuration testing))

Конфігураційне тестування – це тестування роботи на різних платформах. Різні варіанти апаратної конфігурації, версії операційної системи й оточення.

Прикладом тестування несумісності веб-сайту та апаратної частини є ситуація, коли повноцінне користування веб-сайтом можливе лише при використанні відео-карти особливого типу, на приклад, яка підтримує технологію DirectX версії X.X.

1) По суб'єкту тестування [9]:

– Альфа (Alpha tester) - це невелика група людей, яким передається в користування програмний продукт для виявлення невиявлених помилок перше ніж продукт буде доступний основній групі користувачів. Як правило це співробітники компанії які професійно проводять тестування розроблюваної системи.

– Бета (Beta tester) – це також невелика група людей, яким передається в користування програмний продукт для виявлення невиявлених помилок перше ніж продукт буде доступний основній групі користувачів. На відміну від альфа - тестування бета-тестування припускає залучення добровольців з числа звичайних майбутніх користувачів продукту, яким розсилається попередня версія продукту (так звана бета-версія). Такими добровольцями (їх називають бета-тестерами) зазвичай рухає цікавість до нового продукту — цікавість, ради задоволення якої вони цілком згодні миритися з можливістю випробувати наслідки ще незнайдених (а тому і невиправлених) помилок.

2) За часом проведення тестування

– До передачі користувачеві – А́льфа - тестування (Alpha testing)- використання майже готової версії продукту (як правило, програмного або апаратного забезпечення) штатними програмістами (розробниками і фахівцями з

якості програмного забезпечення) з метою виявлення помилок в його роботі для їх подальшого усунення перед бета - тестуванням. Альфа-тестування — це імітація реальної роботи з системою штатними розробниками, або реальна робота з системою потенційними користувачами або замовником на стороні розробника. Часто альфа-тестування застосовується для закінченого продукту як внутрішнє приймальне тестування. Іноді альфа - тестування виконується з використанням оточення, яке допомагає швидко виявляти знайдені помилки. Виявлені помилки можуть бути передані для додаткового дослідження в оточенні, подібному тому, в якому використовуватиметься ПЗ. Альфа - версія продукту — це така версія, в якій присутня велика частина функціональностей, що допомагає зрозуміти, на скільки це відповідає задумці. До альфа-тестування відносяться наступні види тестування:

а) Приймальне тестування (Acceptance testing, smoke testing) – це перший прогін програми (після написання або після внесення істотних змін). Як правило, використається для визначення, чи готова програма для проведення подальшого великого тестування. Приймальне тестування схоже із системним тестуванням, але з наступним розходженням: системне тестування перевіряє, що розроблена система відповідає специфікованим вимогам, а приймальне тестування перевіряє, що розроблена система задовольняє запитаним Замовником вимогам з упором на потреби кінцевих користувачів у даній предметній області.

б) Тестування нової функціональності (New feature testing) – цикл тестування, що виконується для перевірки якості нових функціональностей в розроблюваній системі.

в) Регресійне тестування (Regression testing) - цикл тестування, що виконується при внесенні змін на фазі системного тестування або супроводу продукту. Головна проблема регресійного тестування - вибір між повним і частковим повторним тестуванням і поповненням тестових наборів. При частковому повторному тестуванні контролюються тільки ті частини проекту, які пов'язані зі зміненими компонентами. Пропуск величезного обсягу тестів,

характерного для етапу системного тестування, вдається здійснити без втрати якісних показників продукту тільки за допомогою регресійного підходу.

г) Після передачі користувачеві - Бе́та-тестування (Beta testing) — інтенсивне використання майже готової версії продукту (як правило, програмного або апаратного забезпечення) з метою виявлення максимального числа помилок в його роботі для їх подальшого усунення перед остаточним виходом (випуском) продукту на ринок до масового споживача.

Крім того, бета-тестування може використовуватися як частина стратегії просування продукту на ринок (наприклад, безкоштовна роздача бета-версій дозволяє привернути широку увагу споживачів до остаточної дорогої версії продукту),а також для отримання попередніх відгуків про нього від широкого круга майбутніх користувачів. Бета-тестування виконується для того, щоб отримати зворотний зв'язок про продукт від його майбутніх користувачів.

Бета - версія продукту - така версія, в якій присутнє все. Це повноцінна версія, яку можна використовувати повністю за її призначенням. Звичайно, в ній можуть бути дрібні недоробки, але робота з нею повинна бути повноцінною.

3) За критерієм позитивності сценаріїв [10]

– Позитивне тестування (Positive testing)- на цій стадії тестування треба перевірити результат роботи програми при отриманні нею «правильних» вхідних даних. Позитивне тестування — це тестування на даних або сценаріях, які відповідають нормальній (штатній, очікуваній) поведінці системи. Основною метою позитивного тестування є перевірка того, що за допомогою системи можна робити те, для чого вона створювалася.

– Негативне тестування (Negative testing) - на цій стадії тестування треба перевірити як поводиться програма, отримуючи на вхід «неправильні» дані. Якщо такий варіант описаний в специфікації, а він повинен бути описаний, то треба порівняти очікуваний результат з отриманим результатом. Негативне тестування - це тестування на даних або сценаріях, які відповідають нештатній поведінці системи, - різні повідомлення про помилки, виняткові ситуації, «позамежні» стани і т.п. Основною метою негативного тестування є перевірка стійкості системи до

дій різного роду, валідація невірного набору даних, перевірка обробки виняткових ситуацій (як у реалізації самих програмних алгоритмів, так і в логіці).

4) По ступені ізольованості компонентів, що тестуються [10]

– Компонентне тестування (Component testing) - це тестування програми на рівні окремо взятих компонентів, функцій або класів. Ціль такого тестування складається у виявленні локалізованих у компоненті або модулі помилок у реалізації алгоритмів, а також у визначенні ступеня готовності системи до переходу на наступний рівень розробки й тестування. На рівні компонентного тестування найпростіше виявити дефекти, пов'язані з алгоритмічними помилками й помилками кодування алгоритмів, типу роботи з умовами й лічильниками циклів, а також з використанням локальних змінних і ресурсів. Помилки, пов'язані з невірним трактуванням даних, некоректною реалізацією інтерфейсів, сумісністю, продуктивністю й т.п. звичайно пропускаються на рівні модульного тестування й виявляються на більше пізніх стадіях тестування.

– Інтеграційне тестування (Integration testing) - це тестування частини системи, що складається із двох і більше модулів. Основне завдання інтеграційного тестування - пошук дефектів, пов'язаних з помилками в реалізації й інтерпретації інтерфейсної взаємодії між модулями.

5) Системне тестування (System or End - to - end testing) – це тестування розглядає систему в цілому й оперує на рівні користувальницьких інтерфейсів. Основне завдання системного тестування у виявленні дефектів, таких як невірне використання ресурсів системи, непередбачувані комбінації даних користувальницького рівня, несумісність із оточенням, непередбачувані сценарії використання, відсутня або невірна функціональність, незручність у застосуванні тощо.

6) По ступені автоматизованості тестування [10]

– Ручне тестування (Manual testing) - це виконання тест - кейсів без допомоги яких-небудь програм, що автоматизують роботу спеціаліста з якості програмного забезпечення. Тобто всі тест - кейси виконуються вручну.

Наприклад, створення нового акаунта користувача, заповнення сторінки реєстрації, введення логіна, пароля й т.д.

– Автоматизоване тестування (Automated testing) - це окрема дисципліна тестування. Автоматизація тестування дозволяє зробити процес тестування багаторазово повторюваним без участі реальних дій людини, що дозволяє звільнити ресурси спеціалістів з якості від рутинних, монотонних ручних перевірок функціональності. Автоматизоване тестування ПЗ дозволяє розроблювачам застосовувати підходи, відомі їм по завданнях розробки, для рішення питань тестування. Проектним менеджерам автоматизація тестування дає інструмент для інтеграції фахівців замовника в процес приймального тестування й приймання продукту.

– Змішане тестування (Semi automated testing) – це комбінація ручного підходу та автоматизованого. Наприклад, за допомогою спеціальної програми (tool) .

7) По ступені підготовки до тестування [10]

– Тестування по тест – кейсам (Documented testing) - кількість тест-кейсів визначається для кожного, підлягаючому тестуванню, елемента для кожного рівня тестування. Кожний тест-кейс визначає, як повинна проходити реалізація тестування окремої вимоги або проектного рішення, і які критерії успішного проходження тесту.

Тест-кейси можуть бути документовані в тестовому плані, як розділ програмної специфікації, або в окремому документі, іменованому специфікацією тестів або описом тестів:

1) Специфікація приймальних тестів, що визначає тест-кейси для приймального тестування ПЗ. Звичайно вона видається у вигляді окремого документа, але може бути видана разом із планом приймального тестування.

2) Специфікація системних тестів, що визначає тест-кейси для інтеграції й тестування системи. Звичайно вона теж видається у вигляді окремого документа, але може бути видана разом із планом системного тестування.

3) Специфікації тестів інтеграції ПЗ, що визначають тест-кейси для кожного етапу інтегрування протестованих програмних компонентів. Вони можуть становити розділи специфікації архітектурного дизайну.

4) Специфікації тестування компонент, що визначають тест-кейси для окремих програмних компонентів. Вони можуть становити розділи специфікацій деталізованого дизайну.

Системне й приймальне тестування вимагає величезної кількості окремих тест-кейсів [5].

– Эд хок – тестування (Ad hoc testing) - це інтуїтивне тестування. Воно застосовується як правило:

1) Якщо тест - кейси не формалізовані в документації;

2) У якості додаткового до документованого тестування нових функціональностей і регресійному тестуванню;

3) При необхідності у швидкому тестуванні;

4) У випадку відсутності тест - кейсів.

1.7 Висновок

Загальний погляд на тестування програмного забезпечення останні роки активно еволюціонував, стаючи усе більше конструктивним, прагматичним і наближеним до реалій сучасних проектів розробки програмних систем. Сьогодні тестування розглядається як діяльність, яку необхідно проводити протягом усього процесу розробки й супроводу і є важливою частиною створення програмних продуктів. Планування тестування повинне починатися на ранніх стадіях роботи, необхідно систематично й постійно розвивати й уточнювати плани тестів і відповідні процедури тестування. Сценарії тестування дозволяють виділяти ті аспекти вимог, які можуть неоднозначно інтерпретуватися або навіть бути суперечливими.

2 РОЗРОБКА МЕТОДІВ ТЕСТУВАННЯ WEB-ПРОЕКТІВ

Проекти, засновані на технологіях WWW (World Wide Web), несуть у собі нові проблеми як для розроблювачів, так і для фахівців з якості програмного забезпечення. До цих проблем можна віднести [11]:

- короткі цикли випусків;
- постійно мінливі технології;
- велика кількість користувачів при початковому запуску Web- вузла;
- неможливість контролю користувацького середовища запуску;
- доступність Web-вузла протягом 24 - х годин.

Якість Web-вузла повинна із самого початку бути бездоганною. Будь-які труднощі, будь той час відповіді, точність інформації або практичність будуть змушувати користувача перейти на вузол конкурента. Такі проблеми приводять до втрати користувачів, зниженого рівня продажів і погіршують імідж компанії. Відмінність пріоритетів тестування ґрунтується на типі Web-вузла. Для інформаційних вузлів основною якістю є доступність, у той час як для інтерактивних вузлів важлива швидкість і надійність взаємодії.

Першоджерелом багатьох стратегій, які використовуються для тестування систем, заснованих на технологіях WWW, є проекти, що базуються на технології клієнт-сервер (наприклад, приватні локальні мережі компаній). Цей розділ дипломного проекту охоплює підходи, необхідні для тестування проектів, заснованих на технологіях WWW для яких важливі практичність, безпека, бази даних і інші завдання тестування. Тут також представлені тестові приклади для зразка обраного проекту для тестування.

2.1 Обґрунтування Web-проекту, обраного для тестування

На сьогоднішній день найпоширенішим Web-проектом є електронний магазин. Характерними сторінками, які створює компанія, можна назвати презентації продуктів, дані про компанію (історія, розташування, контактна інформація і т.д.) і політика компанії (форма оплати та ін.). Також можуть задіятися основні принципи й досягнення компанії, заохочувальні призи, знижки або банери, які звичайно допомагають залучати користувачів та інвестиційні засоби для бізнесу. Ці загальні методи призначені для підтримки користувачів, які випадково натрапили на Web-вузол або тримають його серед закладок. Така нерозмірність користувачів (не тільки в плані досвіду, але й у плані мети відвідування Web-вузла) висуває серйозні вимоги до проектування тестових прикладів.

Даний проект представляє приклад електронної книгарні. Для кожної книги демонструється заголовок, ім'я автора, жанр, ціна й анотація. На екрані також може відображатися сама книга. Покупець переглядає книги аналогічно тому, як він це робить у бібліотеці виходячи з назви, автора, жанру, ціни або ключових слів, знайдених в описі.

Покупець вибирає різні товари, розміщаючи їх у кошику для покупок. Коли покупець готовий до покупки товарів, він може відобразити на екрані обрані товари, перш ніж дійсно зробити покупку — у такий спосіб покупець здобуває реальні книги.

Асортименти й орієнтація магазину міняються залежно від сезону й продажів, тому вітрина магазину (тобто сторінка, на якій представлені книги) повинна швидко й легко змінюватися, не виявляючи впливу на безпеку або цілісність вузла.

Середовище, у якому запускається Web-проект, містить безліч компонентів. Реальну конфігурацію визначає адміністратор мережі. Для завдання деяких тестів

може знадобитися проаналізувати окремі настроювання мережі. На рисунку 2.1 показана проста конфігурація, що полягає з наступних компонентів [11].

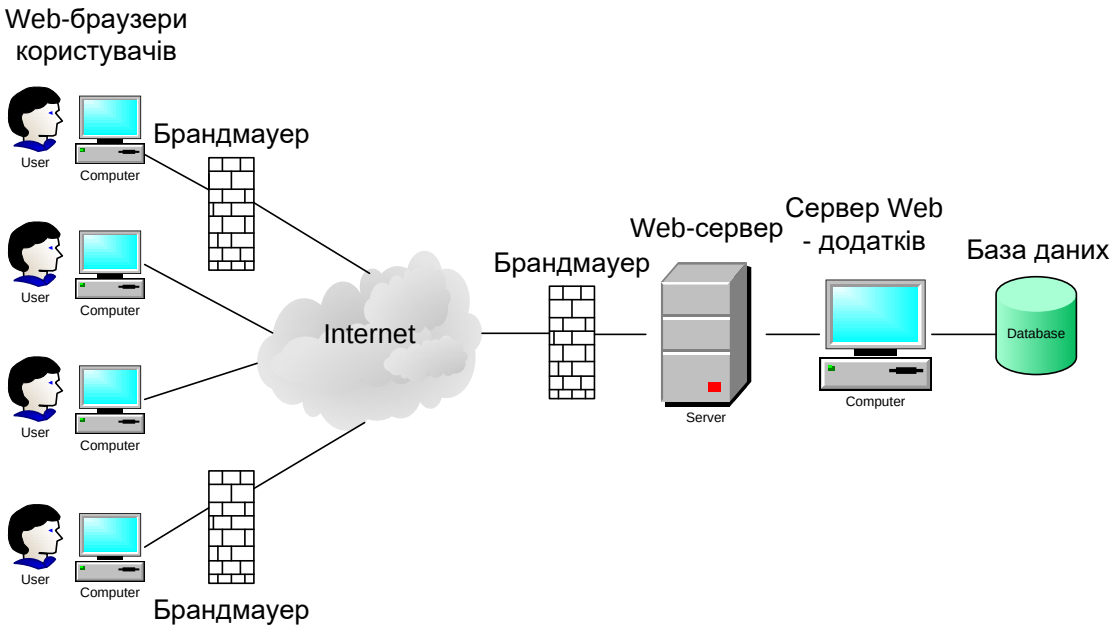


Рисунок 2.1 – Конфігурація середовища, у якому запускається Web-проект

Користувач переглядає Web-вузол за допомогою браузера, підключеного до мережі Internet.

Програмне забезпечення Web-вузла може виконуватися на браузері користувача, сервері мережі, приєднаному в якомусь іншому місці мережі Internet, або на сервері проектів — залежно від того, як розроблювачі спроектували систему.

Брандмауер, який являє собою комбінацію апаратного й програмного забезпечення, існує для того, щоб обгороджувати мережі від зловмисників. Користувач також може встановити брандмауер, щоб захистити комп'ютер від зовнішніх атак.

Проксі - сервер — це програмне забезпечення, ціль якого полягає в тому, що віно є єдиним з'єднанням приватної мережі й Internet. Проксі - сервер виконує безліч функцій (наприклад, запобігає проникненню деяких файлів або покидання ними мережі, поліпшує характеристики системи шляхом кеширування і т.д.).

Багато Web-проектів використовують бази даних для зберігання інформації, необхідної для запуску Web-вузла.

Наприклад, конфігурація клієнт-сервер складається з одного або декількох комп'ютерів користувача, приєднаних до сервера за допомогою локальної (Local Area Network-lan) або глобальної (Wide Area Network — WAN) мережі.

Розміщення магазину в Web приводить до того, що у вузла виникають ті ж проблеми, що й у звичайного магазину, за винятком того, що він відвідується з комп'ютера клієнта. До цих проблем ставляться:

- обробка потоку покупців;
- кількість замовників;
- оплата за продукт;
- випуск продукту;
- безпека інформації про покупців;
- обслуговування покупця під час візиту;
- схоронність бази клієнтів.

Покупець очікує від вузла простої і інтуїтивно зрозумілої навігації, невеликого часу відповіді, цілодобової доступності вузла, надійності, відсутності аварійних ситуацій або сильного запізнювання, а також індикації виконання при використанні вузла. Невідповідність Web-вузла перерахованим вище критеріям веде до швидкого розчарування користувача, таким чином, вузол може втратити своїх клієнтів, що, природно, викличе збитки.

Перші тести для Web-вузла повинні бути орієнтовані на передбачувану поведінку проекту. Необхідно оцінювати наступні проблемні моменти:

- функціональні можливості;
- практичність;
- навігацію;
- форму;
- зміст сторінки.

Проведення цих тестів в ізольованому і контрольованому середовищі тестування допомагає, перш ніж оцінювати вузол за допомогою Internet, переконатися в тому, що основні функції Web-вузла працюють коректно.

У додатку 1 представлені зразки Web-сторінок обраного для тестування електронного магазину.

2.2 Функціональні можливості Web-проекту, обраного для тестування

2.2.1 Тестування функціональних можливостей

Тестування функціональних можливостей полягає в підтвердженні того, що функції, які найбільше впливають на взаємодію з користувачем, працюють відповідним чином [12]. Сюди ставляться:

- форми;
- пошук;
- тимчасові робочі вікна;
- кошик для покупок;
- оплата в інтерактивному режимі.

При завданні тестових прикладів для тестування функціональних можливостей використовуються різні методи (включаючи таблиці й схеми). Повернемося наприклад із книгарнею. У таблиці 2.1 розглянуті зразки тестових прикладів для тестування функціональних можливостей обраної електронної книгарні.

Таблиця 2.1 - Зразки тестових прикладів для тестування функціональних можливостей

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні результати
Продовження на наступній сторінці			

1. Перейти на сторінку з переліком книг 2. Вибрати та дододати книгу в кошик для покупок 3. Проглянути зміст кошику для покупок	- Web-сторінка, - книги, що добавлялися у кошик для покупок	В кошику для покупок перелічуються всі обрані книги	В кошику для покупок перелічуються всі обрані книги
1. Добавити книгу в кошик для покупок, пройшовши по різним сторінкам або за допомогою пошуку 2. Проглянути зміст кошику для покупок	- Книги, що добавлялися у кошик для покупок, - Web- сторінка (сторінки), - критерій пошуку	В кошику для покупок перелічуються всі обрані книги навіть після проведення пошуку або зміни сторінки (сторінок)	В кошику перелічуються книги додані вибрані на сторінках, до кожної книги виводиться кількість
1. Вибрати але не добавляти книгу в кошик для покупок 2. Проглянути зміст кошику для покупок	- Обрані книги	В кошику для покупок не перелічуються всі обрані книги	В кошику для покупок не перелічуються всі обрані книги
1. Вибрати декілька книг та добавити їх в кошик для покупок 2. Не проглядати зміст кошику для	- Добавлені книги	Книги, що були добавлені у кошик для покупок перелічуються	Усі обрані книги перелічилися у контролі

покупок 3. Перейти до контролю		на контролі	
1. Вибрати декілька книг та добавити їх в кошик для покупок 2. Видалити книгу з кошику для покупок	- Добавлені книги, - Видалена книга	Зміст кошику для покупок оновлюється, щоб відобразити зміст без видаленої книги	Кошик оновився і видалена книга зникла

У цьому випадку відсутнє перевага схеми — вказівка на походження. Оскільки візок для покупок — це істотна частина взаємодії з покупцем, тести комбінують доступ до візка для покупок, продовжуючи взаємодіяти з Web-вузлом.

Стовпець "Опис тестового прикладу" містить загальний опис тесту, у той час як реальний дані, записувані тестером, містяться у стовпці "Вхідні дані". Таким чином, тестер використовує ті ж самі тестові приклади з різними вхідними даними, а також задає тести, перш ніж довідається про реальну інформацію, розташовувану на Web-вузлі.

Стовпець «Очікувані результати» містить перелік результатів, які очікується після проведення тесту. Поруч із ним стовпець «Реальні результати», в який тестер записує актуальний результат, який був отриманий після проведення тесту.

Тестування функціональних можливостей дозволяє також оцінити зміст сторінок, що динамічно генеруються. Ці сторінки задаються в міру того як їх запитує користувач, часто видаючи інформацію, яка є результатом пошуку в базі даних. У підрозділі "Тестування змісту

сторінки" динамічний код описаний більш детально, а в розділі "Тестування баз даних" аналізуються проблеми, пов'язані з тестуванням баз даних.

Ще один аспект тестування функціональних можливостей полягає в перевірці безлічі функцій, які неохоче показуються користувачеві: з'єднання з існуючими системами, з базами даних, з'єднання зі сторонніми Web-сторінками.

2.2.2 Тестування практичності

Багато користувачів відразу відкидають продукт, який складний у використанні або не працює, тому перше враження користувача від вузла дуже важливо, незважаючи на це багато Web-вузлів стають усе більш заплутаними в міру збільшення числа функцій [12]. Якщо Web-вузол належить до категорії загального користування, розчарований користувач може легко перемкнутися на вузол конкурента. Якщо ж це вузол підписки, користувачеві прийде очікувати відновлень, які розв'яжуть проблеми існуючого вузла. Невдале ж вирішення таких проблем приведе в результаті до втрати передплатників. Щоб розібратися із цими недоліками, необхідно провести тестування практичності, яке дасть можливість оцінити справність, а також зручність шляхом збору інформації про взаємодію користувачів з вузлом. Основна ідея — створити для команди, що займається продуктом, зворотний зв'язок з користувачем по "враженнях і відчуттям", структурі, навігації й функціям, оскільки у різних груп користувачів різні підходи до засвоєння й керування інформацією.

Ключем до тестування практичності є вивчення реальних дій користувача. Спостерігач фіксує дії користувача і його реакцію, щоб визначити, з яким типом

труднощів зустрічається користувач і як він їх долає. Основні етапи тестування практичності наступні:

- визначення задач Web-вузла;
- визначення групи користувачів, для яких призначений Web-вузол;
- завдання тестів і проведення тестування практичності;
- аналіз отриманої інформації.

Успішне тестування практичності показує не тільки позитивні сторони, але виявляє й слабкі місця проекту. Ці слабкі місця можуть бути так само невизначені, як і час, який витрачає користувач на виконання бажаної дії, або так само ґрунтовні, як і стан користувача, що повністю згубився в парадигмі, використовуваних сторінок. Можливість розібратися із цими слабкими місцями на ранній стадії процесу розробки дозволяє створити більш ефективний продукт.

2.2.3 Завдання тестів і проведення тестування практичності

Даний етап полягає в підготовці й реалізації тестування практичності. Тестові приклади по практичності є чітко заданим і нескладним набором завдань, якому повинні слідувати учасники. При завданні тестових прикладів потрібно брати до уваги знання учасників. У прикладі із книгарнею інструкція повинна включати пошук книги на задану тему з наступною подачею замовлення.

Виконання тестових прикладів по практичності полягає в спостереженні за реакціями й емоціями учасників під час виконання заданих завдань. З боку спостерігача не повинне надходити ніякої інформації й не повинна надаватися початкова допомога. Учасники, у свою чергу, відзначають, що їм сподобалося або не сподобалося у вузлі, а також що їх розчарувало в ході використання вузла. Спостерігач фіксує наступну інформацію:

- чи успішно користувач виконав завдання;

– скільки часу знадобилося користувачеві на виконання завдання;

– у яких місцях користувач зустрівся із труднощами, або де він робив помилки;

– як користувач шукав допомоги у випадку невдач;

– чи надає інтерактивна довідка достатню кількість інформації;

– чи переходив користувач по сторінках або використовував можливість пошуку;

– як користувач реагував на час завантаження окремих сторінок;

– місце, у якім користувачі заплутувалися або навіть не в змозі були виконати завдання,

– кількість переходів по лінкам між завданнями, а також число переглянутих сторінок і те, які саме сторінки проглядалися.

При тестуванні практичності можна застосовувати й інші методи збору даних, такі як відеозапис і реєстрація натискання клавіш. Тести практичності можна зустріти на вузлах розробки або в спеціалізованих лабораторіях, які займаються практичністю. Така лабораторія забезпечує контрольоване середовище з повним набором можливостей за спостереженням, фіксування й складання звітів.

Інше завдання тестування практичності полягає в складанні анкет для вивчення думок учасників про вузол або його прототип. Це дає можливість одержати якісну й кількісну інформацію. На одне питання користувачі відповідають по рейтинговій шкалі, інші питання остаточно відкриті. В анкеті аналізуються наступні питання:

- чи зручно почував себе користувач при використанні вузла;
- відповідна реакція користувача на навігацію й інші функції вузла;
- порекомендував чи б користувач цей продукт друзям;
- чи зрозумів користувач термінологію;
- ідеї щодо вдосконалень;
- що сподобалося й що не сподобалося користувачеві в Web-вузлі й чому.

Який би не був обраний підхід для тестування практичності, усі

результати повинні бути документовані. Документація — це ключ до успішного завершення даного етапу.

2.2.4 Тестування навігації

Гарна навігація — це важлива частина Web-вузла, особливо це стосується складних вузлів, які надають велику кількість інформації. Оцінка навігації є найбільш значимою частиною тестування практичності. Більшість користувачів очікують [8]:

– простого й швидкого доступу до інформації, яку вони прагнуть одержати;

– логічної ієрархії сторінок;

– підтвердження того, де вони перебувають у будь-якій крапці;

– можливість повернутися в попередній стан або на домашню сторінку,

– несуперечливий вид і розміщення кожної сторінки;

– відсутність плутанини серед сторінок.

– Слід охарактеризувати основні проблеми при тестуванні навігації:

– перехід на сторінку й з неї;

– прокручування сторінок;

– тестування всіх лінків (як усередині, так і зовні Web-вузла), щоб підтвердити їхню обґрунтованість і коректність;

– підтвердження того, що немає розірваних лінків;

– перегляд таблиць і форм, щоб переконатися в правильності їх розташування (для різних браузеров розміщення таблиць і форм може бути різним);

– вимір часу завантаження кожної сторінки;

– підтвердження сумісності й погодженого використання клавіш, швидких комбінацій клавіш і дій миші.

2.2.5 Тестування лінків

У ході тестування навігації тестер повинен переконатися в тому, що всі лінки працюють так, як передбачалося, а саме:

– проїхати по кожному лінку, представленому на Web-вузлі, включаючи текстові й графічні гіперлінки;

– підтвердження того, що перехід сторінки, на які переходить користувач, — саме ті, які передбачалося побачити;

– підтвердження того, що всі помилки «дружелюбні» до користувача (наприклад, підтвердження того, що немає помилок типу "404 — не знайдене"). Приклад такої помилки представлений на рисунку 2.2.

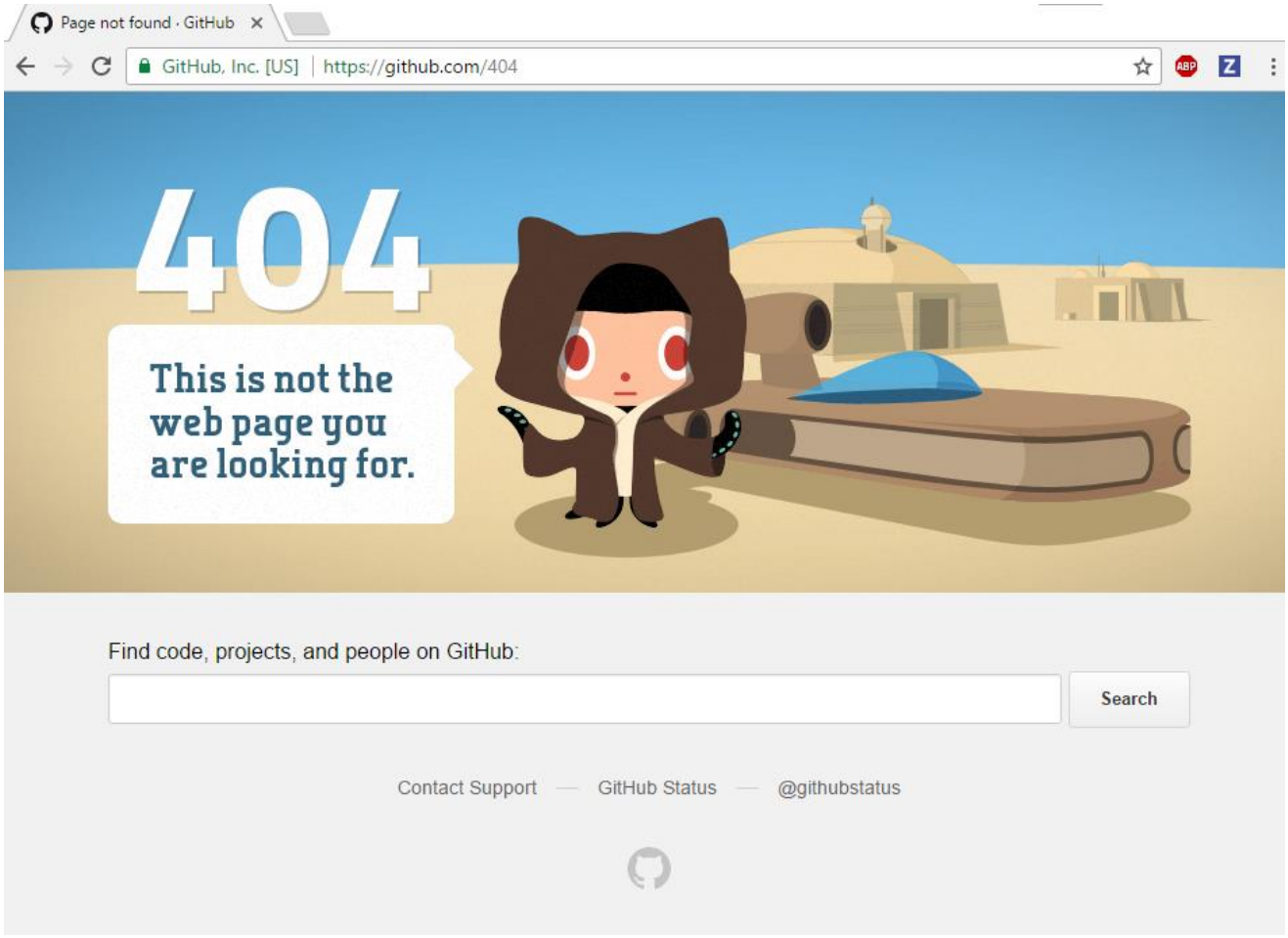


Рисунок 2.2 - Приклад помилки, що виникає при переході по лінку на сторінку, якої не існує

Тестування лінків вручну забирає багато часу й залишає лазівки для суб'єктивних помилок, якщо лінк відсутній. Рекомендується виконувати це

завдання за допомогою автоматизованих інструментів перевірки (так званих "павуків"), що простежують лінки, в Web-вузлі. Автоматизовані інструменти перевірки будуть розглянуті більш детально у третій частині дипломного проекту.

Інша розповсюджена функція Web-вузла — текст " над курсором миші". Він з'являється на екрані, коли курсор перебуває біля якогось об'єкту на сторінці (кнопка, лінк, зображення та ін.). Внаслідок того, що зміст може часто мінятися, найпростішим варіантом для тестерів буде підготовка таблиці контрольних перевірок властивостей тексту "над курсором миші", щоб вона була під рукою в ході виконання тестів по функціональних можливостях або змісту сторінки. Потім тестери тестують ці функції в міру того як вони зустрічаються й відповідно записують результати. Тестування функцій допомагає також переконатися в тому, що розроблювачі не пропустили заголовка " над курсором миші".

2.2.6 Тестування форм

Web-вузли, що використовують форми, потребують тестів, які дозволяють перевірити роботу кожного поля й допомагають проконтролювати форми, що відсилають усі дані так, як це було заплановано дизайнерами. Тестування форм складається з наступних дій [12]:

- використання клавіші табулятора для перевірки того, що форма проходить по полях у тому, що треба порядку як вперед, так і назад;
- тестування граничних значень;
- перевірка коректного відловлювання формами невірних даних, особливо форматів дати й числових форматів;
- перевірка правильного відновлення формами інформації.

У прикладі із книгарнею користувач вводить дані у форму, щоб оплатити покупку. У таблиці 2.2 представлено кілька зразків тестових прикладів.

Таблиця 2.2 - Зразки тестових прикладів для тестування форм електронної книгарні

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні результати
Перехід табулятора від поля до поля	- Начальне поле	Перехід від поля до поля виконується у потрібному порядку	Перехід спочатку переходить по полям форми, а потім переходить на кнопки
Ввести максимальне число символів, що можна ввести у поле	- Назва поля, - набрані символи	Введені в поле дані приймаються	Введені в поле дані приймаються
Збільшити максимальне число символів, що можна ввести у поле	-Набрані символи	Введені в поле дані не приймаються	Введені в поле дані приймаються
Пропустити інформацію в необов’язковому полі	- Назва поля	Форма приймається (за умови, що користувач коректно заповнив іншу частину форми)	Був ініціалізований перехід на іншу сторінку
Пропустити інформацію в обов’язковому полі	- Назва поля	Форма вимагає, щоб користувач надав інформацію в обов’язковому полі	Виникло вікно, яке повідомляє про незаповнені поля

Використання схеми або таблиці приводить до створення більш повного набору тестових прикладів. Розбивка на класи еквівалентності й аналіз граничних значень — це методи, що допомагають задати тести для перевірки роботи більшості полів. Приклади по застосуванню цих методів наведені у підрозділу 2.2.7.

2.2.7 Тестування еквівалентних класів та аналіз граничних значень (Boundary values testing)

Еквівалентний клас — це одне або більше значень введення, до яких ПЗ застосовує однакову логіку [6]. Наприклад, обраний книжковий Web-магазин пропонує нову кампанію: "Більше витрачаєш — більше знижка". Далі представлена таблиця зі специфікації:

Таблиця 2.3 – Категорії витрачених сум, що підлягають знижкам в залежності від величини витраченої суми

Витрачена сума, €	Знижка, %
20 – 40	5
40 – 70	10
70 - 100	15
100 і більше	20

Тут, звичайно відразу очевидні 3 помилки специфікації:

Помилка 1:

Незрозуміло, по якій ставці розраховується знижка, якщо витрачені наступні суми: рівно 40€, рівно 70€, рівно 100€, тому що кожна із цих сум перебуває не в одній, а у двох категоріях зі знижками.

Помилка 2:

Що означає "Витрачена сума"? Це кількість грошових одиниць, виплачених тільки за книги, або повна сума до оплати, включаючи оплату книг і витрати на доставку?

Помилка 3:

Необхідно дописати еквівалентний клас від 0 до 19, на значення якого ніяка знижка не поширюється.

У цьому випадку необхідно внести в специфікацію наступні зміни, які зображені в таблиці 2.4:

Таблиця 2.4 – Категорії витрачених сум, що підлягають знижкам в залежності від величини витраченої суми з урахуванням змін

Вартість куплених книг, €	Знижка, %
0 – 19,99	0

20,00 –39,99	5
40,00 — 69,99	10
70,00 —99,99	15
100,00 і більше	20

Кожне значення усередині кожного класу є еквівалентним усім іншим значенням цього класу.

Вийшло 5 еквівалентних класів, зображених нижче у таблиці 2.5:

Таблиця 2.5 – Категорії витрачених сум, представлені у еквівалентних класах

Клас 1:	0 – 19,99
Клас 2:	20,00 –39,99
Клас 3:	40,00 — 69,99
Клас 4:	70,00 —99,99
Клас 5:	100,00 і більше

До всіх значень класу повинна застосовуватися однакова логіка коду. Наприклад, при вартості куплених книг і 78,11€, 85,45€, і 90€. (клас 4) надається знижка 15%.

Складовими частинами класу є:

– Значення або кількість значень введення (наприклад, від 70,00 до 99,99);

– Логіка для виведення, тобто очікуваного результату (знижка 15% у випадку із класом 4).

Розкладення значень введення на еквівалентні класи полягає в тому, що відсівається величезна кількість значень введення, використовувати які для тестування просто безглуздо.

Відсівання відбувається шляхом застосування знань про тестування граничних значень.

Враховуючи еквівалентні класи з попереднього прикладу:

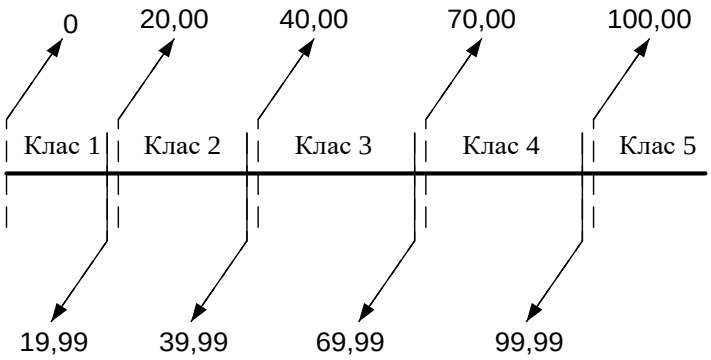


Рисунок 2.3 – Графічне зображення категорій витрачених сум, представлених у еквівалентних класах

Вертикальна пунктирна лінія — це перше можливе значення класу (нижня межа).

Вертикальна суцільна лінія — це останнє можливе значення класу (верхня межа).

Граничні значення – це конкретні значення, які утворюють розділ між еквівалентними класами.

Для кожного еквівалентного класу може бути один із трьох варіантів:

- Є тільки нижня границя (клас 5);

- Є нижня і верхня границі (клас 2, клас 3, клас 4);

- Є тільки нижня границя (не розглянутий у даному прикладі клас, який обмежений тільки зверху гіпотетичним негативним значенням, що безпосередньо передує класу 1).

Тестуванням граничних значень називається застосування методу тестування граничних значень.

Далі представлена методика тестування граничних значень:

1) Спочатку тестується нижня границя даного класу (якщо він є);

2) Потім тестується верхня границя даного класу (якщо він є);

3) Потім тестується будь-яке значення усередині даного класу;

4) Потім тестується верхня границя класу, що безпосередньо передує даному класу (якщо попередній клас є);

5) Потім тестується нижня границя класу, що є наступним за даним класом (якщо наступний клас є).

Для прикладу розглянутий еквівалентний клас 2. Суть тестування полягає в тому, щоб переконатися, що для покупок від 20€ до 39,99€ (включно) буде дана знижка 5%. Слідуючи методиці тестування необхідно лише 5 варіантів даних:

- 20,00;
- 39,99;
- 25,15;
- 19,99;
- 40,00.

У цьому випадку існує 2 000 потенційно можливих позитивних тестів для тестування класу 2 (1 999€ по кількості й 1 випадок, коли було витрачено 20€). Дана методика дозволяє лише трьома позитивними тестами, протестувати 2 000 значень.

Тепер про 5 сценаріїв, яких досить для позитивного й негативного тестування класу 2.

Схематична логіка коду для рішення питання про знижку для класу 2:

ЯКЩО сума ≥ 20 І сума ≤ 39,99, ТО знижка = сума / 100 * 5.

Слід розглянути кожний з тест-кейсів, що представлені у таблиці 2.6, які відображають можливі проблеми коду. Тут досить багато нюансів.

Таблиця 2.6 – Приклад тест-кейсів для тестування граничних значень еквівалентного класу 2

Тест - кейс	Код з виділеною жирним шрифтом частиною, яка перевіряється даним тест – кейсом
	Можлива проблема коду, що викривається тестом, і приклад проблеми
	Очікуваний результат
1. Спочатку тестується нижня границя даного класу (якщо	ЯКЩО сума ≥20,00 І сума ≤ 39,99, ТО знижка = сума / 100 * 5

вона є): 20,00	Помилка у знаку рівності й/або сумі нижньої границі. *Приклад(знак рівності перед 20,00 пропущений):* *ЯКЩО сума >20 І сума ≤ 39,99,* *ТО знижка = сума / 100 * 5*
	5% від 20,00
Потім тестується верхня границя даного класу (якщо вона є): 39,99	ЯКЩО сума ≥ 20,00 І сума ≤ 39,99, ТО знижка = сума / 100 * 5
	Помилка в знаку рівності й/або сумі верхньої границі. *Приклад(39,00 замість 39,99):* *ЯКЩО сума ≥ 20 І сума < 39,00,* *ТО знижка = сума / 100 * 5*
	5% від 39,99
Потім тестується будь-яке значення усередині даного класу: 25,15	ЯКЩО сума ≥20,00 І сума ≤ 39,99, ТО знижка = сума / 100 * 5
	Помилка в знаках більше (>) і менше (<). *Приклад(більше замість менше й менше замість більше):* *ЯКЩО сума ≤ 20,00 І сума ≥ 39,00,* *ТО знижка = сума / 100 * 5*
	5% від 25,15
Продовження на наступній сторінці	
4. Потім тестується верхня	ЯКЩО сума ≥100,00 І сума ≤ 199,99,ТО знижка = сума / 100 * 5

границя класу, що безпосередньо передує даному класу (якщо попередній клас є): 19,99	Тут перевіряється два випадки: 1. Наявність стрибка від верхньої границі попереднього класу до нижньої границі даного класу. Це робиться для наступної ситуації. Припустимо, програміст надрукував 10,00 замість 20,00: ЯКЩО сума ≥ 10,00 І сума ≤ 39,99, ТО знижка = сума / 100 * 5, Якщо зроблена така помилка, то вона не буде виявлена ні тестом 1, ні тестом 2, ні тестом 3.
	Знижка не рівна 5% від 19,99
5. Потім тестується нижня границя класу, що є наступним за даним класом (якщо наступний клас є) 40,00	ЯКЩО сума ≥20,00 І сума ≤ 39,99, ТО знижка = сума / 100 * 5
	Тут так само перевіряється два випадки: 1. Наявність стрибка від верхньої границі даного класу до нижньої границі наступного класу. Це робиться для наступної ситуації. Припустимо, програміст надрукував 49,99 замість 39,99: ЯКЩО сума ≥20,00 І сума ≤ 39,99, ТО знижка = сума / 100 * 5 Якщо зроблена така помилка, то вона не буде виявлена ні тестом 1, ні тестом 2, ні тестом 3, ні тестом 4.
	Знижка не рівна 5% від 40,00

Виходячи з вище перерахованого, можна зробити вивід, що для тестування всіх еквівалентних класів даної специфікації необхідно 14 тест – кейсів для тестування всіх можливих значень, представлених далі в таблиці 2.7:

Таблиця 2.7 – Тест – кейси для тестування всіх можливих значень для еквівалентних класів 1 – 5.

Клас	Значення	Очікувана ставка знижки, %
Клас 1	0	0
	15,55	
	19,99	
Клас 2	20,00	5
	25,15	
	39,99	
Клас 3	40,00	10
	52,11	
	69,99	
Клас 4	70,00	15
	84,17	
	99,99	
Клас 5	100,00	20
	175,15	

Сірим кольором відзначено 5 значень введення, які використовувалися для ізольованого тестування класу 2. Однак, варто врахувати, що для тестування класу 2 разом з його навколишніми класами, необхідно всього трьох тест – кейсів, тому що випадки 4 (19,99) і 5 (40,00) покриваються при тестуванні класу 1 і класу 3 відповідно.

2.2.8 Тестування змісту сторінки

Для кожної Web-сторінки потрібно провести тести, що підтверджують коректність її змісту з погляду користувача. Ці тести діляться на дві категорії: підтвердження адекватного функціонування кожного компонента й

підтвердження того, що вміст кожного з них коректний. Завдання першої категорії тестів — перевірити, що [13]:

– усі зображення й графіки правильно відображаються в різних браузерах;
– увесь зміст представлений згідно з вимогами;
– структура сторінок залишається постійної в різних браузерах;
– представлені всі частини таблиці або форми, а їх розташування правильне;
– лінки на важливий зміст всередині й зовні вузла точні;
– об'єкти з текстом спливаючої підказки коректні.

Web-сторінки зовні виглядають привабливо. Щоб переконатися в коректності змісту, необхідно перевірити зміст на точність, оцінити правильність орфографії, граматики й термінології.

Динамічні сторінки створюються автоматично в міру того як їх запитує користувач; повністю такі сторінки не розміщаються на жодному сервері. Наприклад, здійснюючи пошук окремого предмета, бази даних можуть обновлятися, перераховуючи, таким чином, зовсім різні результати щораз, коли буде проводитися той самий пошук. Для Web-сторінки потрібні підходящі повідомлення, відображувані при відсутності даних або за умови, що кількість представлених даних більше, ніж заданий розмір сторінки.

Звичайно HTML-сторінки, що динамічно генеруються, використовуються для представлення результатів пошуку. Що стосується прикладу із книгарнею, покупець може шукати книги, що ставляться до якої-небудь категорії (наприклад, тестування програмного забезпечення), зсилаючись на ім'я автора або на заголовок. У підрозділі "Тестування баз даних" представлений зразок тестів для перегляду різних результатів пошуку.

Оскільки більшість Web-вузлів припускають безперервні зміни їх змісту, необхідно переконатися, що ці зміни не виявляють шкідливого впливу на вузол у цілому. Регресійне тестування полягає в запуску існуючих тестів, щоб переконатися, що вузол як і раніше працює, як і передбачалося. Завдання таких тестів полягає в наступному:

- визначити, чи завантажує нова сторінка те ж, що й попередня;

- перевірити ідентичність нової сторінки з попередньою версією;

- перевірити, чи залишається важлива інформація тією ж або ж вона відрізняється;

- перевірити коректність лінків.

У таблиці 2.8 показано кілька зразків тестових прикладів для книгарні. Ці конкретні тести викладені у вільній формі й виконують функції скоріше загальної таблиці контрольних перевірок для тестування вмісту сторінки. Тестер може задавати тести, які включають певні зображення, таблиці або інші об'єкти тестування.

Таблиця 2.8 - Тестові приклади для тестування змісту сторінки

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні результати
Проглянути зображення та графіки	- Web-сторінка, - браузери	Зображення та графіки коректно відображуються у вибраному браузері	Зображення знаходяться де потрібно, і нормального розміру
Проглянути таблицю	- Таблиця на Web-сторінці, - браузери	Таблиця коректно відображується у вибраному браузері	Таблиця коректно відображується
Помістити курсор біля кожного об’єкту	- Об’єкти, що тестуються	Над кожним об’єктом з’являється потрібний текст	Над кожним потрібним об’єктом з’явився текст
Виконати пошук по категорії (наприклад, «тестування програмного забезпечення»)	- Категорія пошуку	В результаті пошуку перелічуються усі книги обраної категорії	В результаті пошуку перелічуються усі книги обраної категорії

2.3 Тестування конфігуріції й сумісності

Основне завдання тестування Web-проектів полягає в підтвердженні того, що користувач бачить Web-сторінку такою, якою вона була задумана дизайнером. Користувач може вибирати в якості браузера різне програмне забезпечення й різні його настроювання, використовувати різне мережне програмне забезпечення й інтерактивні послуги, а також запускати одночасно інші Web-сторінки. Web-проект може містити сервери, бази даних і різні пристрої для встановлення з'єднання [13].

Як для користувача, так і для Web-проекту установки апаратного забезпечення й конфігурація також впливають на середовище. Сюди входять типи ЦП, ОЗП, графічні адаптери, відео-карти, монітори, мережні плати й типи з'єднань (наприклад, ADSL, DSL, модемне). Усі вони можуть впливати на появу Web-сторінки. У той час як звичайне оточення Web-проекту залишається незмінним, можна чекати, що користувацьке оточення буде різноплановим з декількома варіаціями залежно від конкретного користувача. Таким чином, завдання тестування конфігурації й сумісності полягає в тому, щоб переконатися, що додаток функціонує коректно в Internet.

При написанні тестових прикладів для тестування конфігурації на враження користувача можуть впливати проблеми з різним оточенням і конфігураційними установками. У гарних вимогах даються відповіді на перераховані нижче питання:

– Чи перебуває користувач за брандмауером або проксі – сервером;

– Чи приєднує користувач за допомогою сервера розподілу навантаження (тобто машини, яка залишається однаково завантаженої на всіх доступних Web-серверах);

– Чи ухвалює браузер маркери cookie? (Маркер cookie — це пакет даних, відправлений Web-сервером, а потім збережений на жорсткому диску користувача. Він зберігає інформацію про користувача і його перевагах.);

– Чи є можливість побудувати надійний захист;

– Які технології використовують розроблювачі Web-сторінки. Наприклад, якщо для Web-сторінки використовуються керуючі елементи Activex або створення сценаріїв Java, тестер повинен знати, які версії браузера підтримують ці реалізації.

– Чи використовуються інструментальні засоби сервера, що забезпечують захист (блокувальні механізми)?

Тестування сумісності допомагає переконатися у функціональних можливостях і надійності продукту в підтримуваних браузерах і платформах користувацького комп'ютера. Зростання числа браузерів і їх версій (кожна з яких запускається в різних операційних системах і на різних платформах, а також має відмінні від інших графічні можливості й plug-in) призвело до виникнення сотень різних користувацьких оточень. Користувацький браузер приєднується до сервера й іншим комунікаційним засобам, які для виконання Web-проекту повинні коректно взаємодіяти [14].

Оскільки в циклі тестування може не виявитися часу на тестування всіх різновидів середовища, установити пріоритетні тести можуть допомогти наступні керівні принципи.

- Необхідно запитати в розроблювачів про окремі вразливі конфігурації або несумісні сценарії й націлитися на ці складні області.
- Сконцентруватися на найпоширеніших на ринку версіях браузерів. В ідеалі тестер повинен знати про поширеність даного браузера.
- Тестер повинен зосередити свою увагу на тих платформах, які найімовірніше використовуються клієнтами.
- Необхідно сконцентруватися на основних версіях програмних продуктів, включаючи функції й ключові зміни.

У таблиці 2.9 перераховані платформи й оточення браузерів, на яких проводиться тестування. Ціль полягає у виконанні тих самих тестових прикладів на різних комбінаціях платформ і браузерів.

Таблиця 2.9 – Таблиця сумісності браузерів з операційними системами

Браузер	Windows			Mac OS X	Linux
	Win 7	Win 8	Win 10		
Microsoft Edge	-	+	+	-	-
Mozilla Firefox 53.0.2	+	+	+	+	+
Opera 41.0	+	+	+	+	+
Safari 8	-	-	-	+	-
Google Chrome	+	+	+	+	+

В ідеалі, звичайно, потрібно виконувати кожний тестовий набір для кожної комбінації браузер/платформа. У дійсності такі ресурси, як час і люди, обмежують число тестів, які можна виконати. Поділ тестування комбінацій браузер/платформа серед команди по тестуванню може допомогти прискорити виконання тестів. Кожний член команди відповідає за тестування специфічного набору тестових прикладів у певному оточенні браузер/платформа. Щоб визначити, які тести потрібно виконувати обов'язково для всіх комбінацій браузер/платформа, а які — тільки в особливому середовищі, можна використовувати аналіз ступеня ризику, який гарантує повне покриття пріоритетних платформ.

Під час виконання тестування на сумісність можуть використовуватися тестові приклади, представлені у таблиці 2.10.

Таблиця 2.10 – Зразки тестових прикладів по сумісності

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні резуль-тати	Версія брау-зера
Добавити Web-вузол списку Favorites та повторно встановити сеанс, який вже був завершений	- Web-сторінка	Web-вузол має появитися та функціонувати потрібним чином	Web-вузол з'явився та коректно працює	Chrome 58
Використати кнопку «НАЗАД» після підтвердження інформації про кредитну картку або регістрації події	- Web-сторінка, що відображалася до використання кнопки «НАЗАД»	Інформація про кредитну картку не повинна відображатися, а користувачеві повинна бути представлена сторінка, яка задається за замовчуванням або головна сторінка	На екрані не відображається ніякої приватної інформації	Chrome 58
Встановити декілька сеансів з'язку з Web-вузлом	- Доступна Web-сторінка	Можна використовувати всі сеанси з'язку	Усі вузли працюють нормально	Chrome 58
Провести сеанс з набором маркерів cookie та без нього	- Web-сторінка	Функціональні можливості Web-вузла реалізовані так, як планувалося у проекті	Усі функції вузла працюють як і очікувалося	Chrome 58
Використати можливості друку в браузері	- Web-сторінка	Надруковані результати є зразком сторінки, яка була запрошена для друку	На друк видалася сторінка разом з вже доданим користувачем ткестом	Chrome 58

2.4 Тестування надійності та доступності

Ключова вимога до Web-вузла полягає в тому, що він повинен бути доступний користувачеві в будь-який момент часу протягом 24-х годин на добу. Число користувачів, що одночасно одержують доступ до нього, також може впливати на його доступність. Щоб оцінити доступність, необхідно створити тести навколо очікуваних піків, неаприклад, стосовно елктронної книгарні це стимулюючі компанії й продажі;

Тестери повинні також виконати перевірку на предмет проблем з ресурсами (таких як витік пам'яті й обмеження баз даних), які можуть погіршити характеристики або навіть припинити роботу проекту, що приведе в результаті до майбутніх втрат у бізнесі. Витік пам'яті відбувається тоді, коли пам'ять, яка більше не використовується, не вертається в пул вільної пам'яті. Витік пам'яті може відбуватися повільно і її складно встановити. В остаточному підсумку, неможливість виявлення витоку пам'яті приводить до зупинки комп'ютерної системи. Обмеження баз даних містять у собі зберігання баз даних, що запускаються раціонально (гарантуючих, що вхідні дані узгодяться із заданою змінною), тих, для яких виконується регулярне резервне копіювання й тих, які гарантують, що запис не використовує весь дисковий простір, що залишився [14].

Важливо розуміти архітектуру системи для того, щоб провести адекватне тестування доступності й надійності. Наприклад, якщо в системі втримується два Web-сервера для вирівнювання навантаження, то тестери повинні могти описати характеристики системи з одним Web-сервером. При вирівнюванні навантаження підтримується однакова завантаженість усіх доступних Web-серверів. Якщо за якимись причинами один з Web-серверів буде вилучений із продукту (наприклад, для обслуговування), то Web-сайт повинен продовжувати функціонувати на більш низькому, але все-таки прийнятному рівні.

Багато питань виникає щодо систем із цілодобовим доступом. Нижче перераховані ключові області для дослідження:

– тестування продукту при використанні не в години пік;

– необхідність в аналогічному або дублюючому тесті/промислових апаратних засобах;

– масштабування характеристик;

– аутентифікація користувача.

У таблиці 2.11 представлено кілька зразків тестових прикладів для тестування надійності й доступності в прикладі з вітриною магазину.

Таблиця 2.11 - Зразки тестових прикладів для тестування надійності й доступності

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні результати
Чотири користувача зайшли в систему та одночасно намагаються купити одну й ту ж книгу	- Користувач 1 - Користувач 2 - Користувач 3 - Користувач 4 - Книга	Всі чотири користувача можуть одночасно добавити у свій кошик для покупок одну й ту ж книгу	Усі активні користувачі без проблем купили вибрану книгу
Повторити минулий тест після модернізації бази даних		Модернізація бази даних пройшла успішно та користувачі можуть покупати книги; у системі не виявлено будь-яких погіршень характеристик	Усе працює як і потрібно навіть після модернізації
Під час друкування звіту наприкінці місяця скористатися Web-вузлом для покупки книги	- Книга	Звіт надається наприкінці місяця; у системі не виявлено будь-яких погіршень характеристик	Як і очікувалося звіт видається у кінці місяця без проблем
Створити та виконати сценарій автоматичного тесту, щоб можна було покупати книги кожні три хвилини	- Книга 1 - Книга 2 - Книга 3 - Книга n	Всі книги покупаються; у системі не виявлено будь-яких погіршень характеристик	
Провести тест на об’єм, в якому кожні 20 хвилин ініціюються 10 нових користувачів. Запустити цей тест на 1 годину		Всі користувачі можуть отримати доступ до системи та працювати з різними функціями Web-вузла; у системі не виявлено будь-яких погіршень характеристик	
Повторити тест W-25 з меншою кількістю	- Виключені ресурси	Всі користувачі можуть отримати доступ до	

ресурсів, тобто число Web-серверів повинно бути на одиницю менше		системи та працювати з різними функціями Web-вузла; у системі не виявлено будь-яких погіршень характеристик	

2.5 Тестування характеристик Web- проекту

Тестування характеристик (тобто оцінка характеристик системи при нормальному і інтенсивному використанні) критично для успіху будь-якого Web-сайту. Система, у якої на відповідь іде занадто багато часу, може розчарувати користувача — у результаті він переходить на вузол конкурента. В остаточному підсумку кожна запитувана сторінка буде видана, однак це може не задовольняти потреби користувача. При тестуванні характеристик необхідно переконатися, що сервер Web-вузла відповідає на запити браузера в рамках певних параметрів [14].

Для WWW дуже природнім видасться той факт, що різні користувачі по-різному взаємодіють із сайтом, сильно впливаючи на характеристики. До аспектів, які можуть вплинути на характеристики, ставляться:

– висока активність і обсяги користувачів під час запуску;

– час доби;

– піки активності через маркетингове стимулювання;

– нестача ресурсів через безліч користувачів у мережі;

– час завантаження;

– модель користування;

– час обмірковування;

– інтенсивність вхідного потоку користувачів;

– платформи клієнтів;

– швидкість доступу до Internet;

– швидкість вихідного потоку.

Тестування, проведене в локальних системах, добре зарекомендувало себе при перевірці якості, але не дає можливості виміряти час відповіді, пов'язане із запізнюванням в Internet. Тестування, проведене в географічно віддалених районах, дозволяє оцінити внесок усіх відносних запізнювань. Розмір кожної сторінки й кількість зображень, використовуваних на Web-вузлі, також може

вплинути на характеристики програмного продукту. Реалізація браузера, що кеширує сценарії тестів, щоб підтвердити адекватність часу відповіді, є ключовою областю для спостережень у ході тестування характеристик.

Тестування характеристик повинне проводитися на ранній стадії циклу тестування. У багатьох організаціях тестування характеристик планується на останній етап тестування. Такий метод планування може виявитися не занадто ефективним, особливо якщо за результатами тестування характеристик визначаються основні діри в проекті. Тестування характеристик потрібно проводити тоді, коли команда по тестуванню встановила, що функціональні можливості стабільні.

У таблиці 2.12 представлені зразки тестових прикладів, розглянутих у ході тестування характеристик. Узгодження часу, основний предмет занепокоєння, повинне задаватися у вимогах.

Таблиця 2.12 - Зразки тестових прикладів для тестування характеристик

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні результати
1. Знайти книгу; 2.Зафіксувати час відповіді на запит; 3.Повторити цей тест декілька разів	- Запит	Зафіксувати мінімальне, максимальне та середній час відповіді та переконатися, що вони задовольняють вимогам	Запит обробився менше майже за секунду, задовольняє вимогам
1. Передати заповнену форму покупця; 2.Зафіксувати час відповіді на запит; 3.Повторити цей тест декілька разів	-Куплена книга	Зафіксувати мінімальне, максимальне та середній час відповіді та переконатися, що вони задовольняють вимогам	Запит обробився менше майже за секунду, задовольняє вимогам
Продовження на наступній сторінці			

1. Включити в браузері можливість кеширування; 2. Заповнити форму; 3. Передати форму 4.Зафіксувати час відповіді, яки виміряється між кінцевими точками ми (тобто браузер-WWW-Web-вузол -WWW - браузер) 5.Повторити цей тест декілька разів	- Передана форма	Зафіксувати мінімальне, максимальне та середній час відповіді та переконатися, що вони задовольняють вимогам	Запит обробився не набагато швидше, але результат став краще
1. Відключити в браузері можливість кеширування; 2. Заповнити форму; 3. Передати форму 4.Зафіксувати час відповіді, яки виміряється між кінцевими точками ми (тобто браузер-WWW-Web-вузол -WWW - браузер) 5.Повторити цей тест декілька разів	- Передана форма	Зафіксувати мінімальне, максимальне та середній час відповіді та переконатися, що вони задовольняють вимогам	Запит почав оброблятися зі старою швидкістю
1. Війти в систему з віддаленого місця; 2. Заповнити форму; 3. Передати форму 4.Зафіксувати час відповіді, яки виміряється між кінцевими точками ми (тобто браузер-WWW-Web-вузол -WWW - браузер) 5.Повторити цей тест декілька разів	- Віддалене розташування; - Передана форма	Зафіксувати мінімальне, максимальне та середній час відповіді та переконатися, що вони задовольняють вимогам	Час збільшився але ще задовольняє умовам

При тестуванні характеристик проводиться також тестування завантаженності. Ціль тестування завантаженості — змоделювати відчуття реального світу шляхом генерації безлічі користувачів, що одержали доступ до Web-вузла [14]. Автоматизація збільшує здатність проводити ефективні тести по завантаженості, оскільки з'являється

можливість эмулювати тисячі користувачів, відправляючи одночасні запити на сайт або на сервер.

Тестування завантаженості для успішного виконання тестів по завантаженості із самого початку повинно бути ретельно сплановане. Однак навіть при оптимальному плануванні тестування завантаженості іноді необхідно повторити, принаймні, декілька разів.

Далі представлено кілька важливих моментів у підготовці тестів по завантаженості для успішного їхнього завершення.

1) Необхідно зрозуміти вимоги системи до завантаженості, а також вивчити загальне й поточне число користувачів, яке може знадобитися для підтримки Web-вузла. У проектів, що були запущені вперше, немає історичної статистики. Команда по тестуванню може виявитися залежної від специфікації вимог, тому що необхідно буде зібрати такі специфічні дані:

– число користувачів у вигляді будь-якого унікального числа відвідувань за день, за тиждень або за місяць;

– поточна сумарна кількість користувачів: найгірший варіант сценарію в годинник пік;

– максимальне значення швидкості запитів: число сторінок, що обслуговуються за секунду.

2) Установити, які інструменти (наприклад, інструменти для тестування й моніторингу) будуть використовуватися при проведенні тестів по завантаженості;

3) Згенерувати достатнє число користувачів і транзакцій, щоб оцінити можливості й характеристики, які будуть підтримуватися в живому оточенні;

4) Створити базис: сценарії для імітації одного користувача з одним браузером;

5) Створити сценарії тестів для імітації декількох сеансів зв'язку, установлених на декількох браузерах;

6) Визначити всі інші Web-проекти, що запускаються на сервері Web-проектів, щоб фіксувати коректну діяльність системи. (Багато проектів, що запускаються одночасно в одній системі, можуть вплинути на характеристики);

7) Виконати тест (тести) кілька разів;

8) Визначити учасників, які будуть контролювати системні характеристики в ході виконання тесту;

9) Підготувати результати тестів, у яких буде фіксуватися мінімальний, максимальний і середній час відповіді системи.

На виконання тестів по завантаженості потрібно витратити час, особливу увагу слід звернути на настроювання реальних тестів.

2.6 Тестування безпеки

Безпека викликає особливий інтерес при діловім спілкуванні й веденні справ (особливо у випадку конфіденційних повідомлень і критичних для бізнесу транзакцій) через Internet. Користувач повинен бути впевнений, що його особиста й фінансова інформація буде в безпеці. Дуже важливо виявити вразливі місця, які могли б дозволити несанкціонований доступ користувача до системи.

Не беручи до уваги вимогу системи щодо введення користувачем кодового слова, щоб одержати доступ до Web-вузла, необхідно провести перевірку відомих на даний момент погроз у мережі Internet. У визначенні додаткових тестів, пов'язаних з безпекою, можуть допомогти відповіді на наступні питання:

1) Які існують заходи для запобігання або обмеження атак з боку хакерів;

2) Настроюються чи в браузері установки, що гарантують максимально безпечний захист;

3) Як Web-вузол оперує правами доступу;

4) Чи оперує система таємними діями транзакцій, тобто зміною первісної адреси відправлення;

5) Чи надають постачальники e-commerce механізм запобігання шахрайства із кредитними картками;

6) Як Web-вузол шифрує дані;

7) Яким чином Web-вузол аутентифікує користувачів;

8) Чи дає можливість перегляд вихідного коду виявити найбільш важливу інформацію;

9) Чи можна одержати доступ до системи й нанести їй збиток шляхом прямого виклику баз даних по лініях, що комутируються, зв'язку за допомогою модему;

10) Як захищені кредитні картки або інформація користувача, яка захищеність вважається достатньою.

Програмне й апаратне забезпечення, що містить брандмауерів, є ключовим компонентом у підтримці захищеності мережі від зловмисників. Завдання тестування брандмауера полягає в знищенні й блокуванню механізмів безпеки для того, щоб визначити їхню ефективність. Реєстрація й розгляд мережного інформаційного потоку (як вхідного, так і вихідного) — це основний вид діяльності при тестуванні брандмауерів. Ще один аспект полягає у виконанні тестів на проникнення, у яких авторизована особистість намагається втрутитися в роботу мережі. Тестування брандмауера підтверджує, що:

– фільтрація пакетів працює так, як було закладено в проекті;

– брандмауер коректно реалізує політику безпеки компанії (наприклад, якщо в політику затверджується, що брандмауер повинен пересилати весь поштовий потік даних на поштовий сервер, то тестер повинен у цьому переконатися);

– брандмауер при необхідності подає аварійний сигнал;

– у периметрі, створеному брандмауером, не повинне бути ніякого витоку (витік відбувається, коли процедура доступу до мережі відрізняється від доступу через існуючі захищені ворота, дозволяючи в такий спосіб зловмисникові обійти захист брандмауера, щоб потрапити в мережу);

– брандмауер приховує інформацію й адреси із внутрішньої мережі, яку він захищає;

– адекватна реєстрація подій допомагає відшукувати зловмисників і знаходити діри в системі безпеки, викликані якою-небудь подією.

Тестування безпеки вимагає особливої обережності. Недбалі процедури тестування можуть привести до ненавмисним ушкодженням або руйнуванням через перевантаження мережі запитами на обслуговування й скручування портів на головних машинах. Крім того, організація повинна оберігати результати тестів, щоб не дати можливості неавторизованим особистостям довідатися що-небудь про вразливі місця системи.

2.7 Тестування наскрізних транзакцій

Наскрізні транзакції дотримуються послідовності виконуваних клієнтом дій починаючи з моменту відвідування й закінчуючи моментом відходу з вузла. Тут тестуються окремі компоненти, які виконують окремі транзакції й до яких можуть відноситися Web-браузери, Web-сервери, бази даних і проміжне програмне забезпечення. Тестові приклади для тестування наскрізних транзакцій служать доповненням до тестування практичності. У прикладі з вітриною магазину послідовність виконуваних дій наступна [14]:

– Клієнт відвідує вузол;

– Клієнт переглядає книги;

– Клієнт вибирає книги й поміщає їх у візок для покупок;

– Клієнт указує адресу доставки;

– Клієнт платить за покупку.

– Магазин відсилає квитанцію користувачеві по електронній пошті;

До наскрізних транзакцій ставиться також обробка будь-яких несприятливих умов, які виникли в результаті запитів клієнтів, наприклад:

1) Обраної книги немає в наявності;

2) Книги потрібно спеціально замовляти;

3) Клієнт прагне вилучити з кошика для покупок одну або кілька обраних книг;

4) Клієнт прагне скасувати замовлення (до або після покупки);

5) Клієнт прагне повернути гроші.

Кредитна картка, використовувана для покупки, виявилася непридатною (тобто недійсної).

Внаслідок використання різних комбінацій дій користувача необхідно, щоб у своїй основі тестовий приклад походив на потік через Web-вузол. У таблиці 2.13 перераховані зразки тестових прикладів для тестування наскрізних транзакцій.

Таблиця 2.13 - Зразки тестових прикладів для тестування наскрізних транзакцій

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні результати
1. Проглянути вузол та добавити декілька книг (одну книгу) у кошик для покупок; 2.Оцінити змвст кошику для покупок; 3.Видалити книгу з кошику для покупок	-Додані книги; - Видалені книги	В кошику для покупок перелічуються книги (книга) за винятком тієї, що була видалена	Видалена книга зчезла
1. Проглянути вузол та добавити декілька книг (одну книгу) у кошик для покупок; 2.Купити книги; 3.Перед введенням номеру кредитної картки відмінити заказ	-Додані книги;	Вихід з меню покупок та повернення до сторінки кошику для покупок	У кошику збереглися книги, але даних картки немає
1. Проглянути вузол та добавити декілька книг (одну книгу) у кошик для покупок; включаючи; 2. Купити книгу; 3. Передати форму	-Додані книги; - Інформація, що була надана при покупці	Оброблюється транзакція та відправляється підтверджуюче поштове повідомлення	На пошту було отримане повідомлення про підтвердження
1. Проглянути вузол та добавити декілька книг у кошик для покупок; 2. Купити книги; 3. Представити недійсну кредитну картку	-Додані книги; - Кредитна картка	Покупка відміняється та відбувається перехід на іншу сторінку	При перевірці картки виникає помилка як і очікувалося
1. Проглянути вузол та добавити декілька книг у кошик для покупок; 2. Купити книги, але заповнити не всі обов’зкові поля; 3. Передати форму кредитну	-Додані книги; - Пропущені поля	З’являється повідомлення про помилку, що вказує на відсутність інформації у всі обов’зковому полі	Вимагається заповнення полів

картку			

2.8 Тестування баз даних

Тестування баз даних часто є дуже важливою частиною тестування WWW, оскільки багато систем надають який-небудь тип можливості пошуку. Продукти або інформація на Web-вузлах звичайно зберігається в базах даних, що полегшують і прискорюють пошук обраного елемента [14]. У деяких програмних проектах є дані, що вводяться користувачем, які потім стають частиною баз даних. Приклади сторінок бази даних електронного магазину представлені у додатку 2.

При тестуванні баз даних потрібні всебічні знання системи. Необхідний також додатково застережений план підходу до тестування даних. Важливо розуміти концепції проекту бази даних і правила реалізації. До ключових проблем, що виникають при тестуванні баз даних, ставляться:

- цілісність даних;
- достовірність даних (підходяща форма при введенні в бази даних);
- маніпуляції з даними й відновлення (відновлення значення кількості проданих книг, доступних книг і т.д.).

Цілісність даних — це основна умова успішної реалізації. Організація повинна встановити засоби для контролю викривлення даних. Викривлення даних часто виявляється тільки на пізніх стадіях, оскільки викривлення відбуваються в підході "створення блоків". Усе починається з малого, але згодом (або зі збільшенням числа транзакцій) проблема збільшується.

При тестуванні баз даних також потрібно враховувати достовірність даних. Вона гарантує надання клієнтам точної інформації й повернення цієї інформації в базу даних. Як правило, частка дефектів у даних значно більше, чим у самій програмі. У такому випадку необхідно вивчити послідовність виконуваних дій і перевірити базу даних у точках змін. Цей підхід полягає в ізоляції дій, які модифікують бази даних, а також в інспектуванні зміненого змісту на предмет

коректності. Важливо, щоб спеціаліст з якості усвідомлював області ризику й правильно розставляв пріоритети при тестуванні.

Тестування баз даних виконується на двох рівнях: адміністративних і користувацьких функцій. Адміністратор баз даних може виконати дії, недоступні для клієнтів Web-вузла. У прикладі із книгарнею до найпоширеніших адміністративних дій ставляться:

– додавання нових книг у перелік товарів;

– видалення існуючих книг з переліку товарів;

– відновлення специфічних полів, наприклад, таких як зміни в ціноутворенні.

Наступний контрольний список допоможе сформулювати безліч додаткових адміністративних тестових прикладів:

– Зрозуміти проект бази даних;

– Зрозуміти процедури відновлення й удосконалення обслуговування баз даних;

– Переконатися, що вимоги були задоволені;

– Переконатися в працездатності й продуктивності, коли кілька користувачів одночасно роблять запит (а також включити максимальне число спільно або одночасно діючих користувачів згідно зі специфікацією);

– Переконатися, що процедури резервного копіювання й відновлення працюють так, як було спроектовано, і не впливають на вимоги доступності;

– Переконатися, що база даних допускає максимальне число з'єднань, які система згідно із проектом повинна обробляти;

– Переконатися, що при виконанні операцій бази даних є досить місця й пам'яті для певної кількості даних. Розширити можливості системи, якщо ці фізичні межі простору й пам'яті перевищуються.

У прикладі із книгарнею Web-проект і Web-інтерфейс користувача згодом залишаються незмінними. Зміни відбуваються тільки при додаванні книг і відстеженні списку клієнтів. Доступ користування до цієї бази даних полягає

переважно в пошуку інформації. Для інших Web-вузлів, у яких бази даних змінюються безупинно, оптимізація й швидкість одержання результатів пошуку й уведення залишаються критичними.

У таблиці 2.14 представлено кілька загальних зразків тестових прикладів для тестування електронної книгарні. Інформація в стовпці "Очікувані результати" буде змінюватися залежно від вимог до баз даних. Наприклад, завдання імені неіснуючого автора може дати в результаті один або декілька можливих варіантів.

Таблиця 2.14 - Зразки тестових прикладів для тестування баз даних

Опис тестового прикладу	Вхідні дані	Очікувані результати	Реальні результати
Виконати пошук книги, задавши правильне ім’я автора	- Автор	Відправляється список книг, написаних вказаним автором	Виведено список книг вписаного автора
Виконати пошук книги, задавши правильне, але не унікальне ім’я автора	- Автор	Відправляється список книг, написаних всіма авторами з такими іменами	Пошуковий запит виводиться усі книги усіх авторів з таким ім’ям
Виконати пошук книги, задавши правильне ім’я автора та назву книги	- Автор; - Назва книги	Відправляється інформація про цю книгу	Виводиться сторінка книги цього автора
Виконати пошук книги, задавши неіснуюче ім’я автора та назву	- Автор	Ім’я автора не знайдено, але відправляється пропозиція альтернативних написів	Наводяться альтернативні автори
Виконати пошук книги, задавши існуюче ім’я автора, але неіснуючу назву книги	- Автор; - Назва книги	Назви книги не знайдено але перелічуються книги, написані цим автором	Виводяться книги цього автора
1. Виконати пошук вірної книги по назві; 2.Адміністратор змінює ціну на цю книгу 3. Виконати пошук тієї ж книги по назві;	- Назва книги; - Нова ціна	1. Відправляється інформація про книгу; 2. Оновлюється база даних; 3. Відправляється інформація про книгу з новою ціною	Ціну змінено на сторінці піся оновлення сторінки
Продовження на наступній сторінці			

Виконати одночасно наступні кроки: 1.Адміністратор добавляє нову книгу; 2. Клієнт шукає нову книгу			

* Кожний запит виконується на окремому комп'ютері з настойками Internet із використанням підходящого браузера.

2.9 Висновок

У цьому розділі були представлені параметри тестування та розробоені тест-кейси для тестування обраного електронного магазину, були розглянуті аспекти й відповідні підходи до підготовки тестів та їх виконання.

Тестування функціональних можливостей допомогає визначити працездатність основних функцій системи, чи працюють вони коректно та задовольняють вимогам специфікації. В інших типах тестування специфіка проблем переноситься в середовище WWW. Багато із цих типів тестування також ефективні й для оточення клієнт-сервер. При тестуванні практичності за допомогою спостереження за користувачами під час їх взаємодії з вузлом оцінюється «дружелюбність» вузла до користувачів.

Оцінка навігації є найбільш значимою частиною тестування практичності. При тестуванні навігації шляхом перевірки доступу до вузла, зображень, і іншим компонентам вузла підтверджується, що користувач може виконати бажані дії.

Тестування форм необхідне для перевірки працездатності кожного поля. Інформація, введена у поле повинна коректно відображатися після збереження та передавитися на сервер.

При тестуванні конфігурації й сумісності підтверджується, що програма функціонує коректно в різних апаратних і програмних середовищах.

При тестуванні надійності й доступності оцінюється доступність Web-вузла в будь-який час по запиту користувача. Така оцінка досягається шляхом тестування програми в піковий час використання (у періоди маркетингової стимуляції й циклів високої активності).

3 РОЗРОБКА ЗАСОБІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ WEB-ПОЕКТІВ

У цьому підрозділі доцільно розглянути такі питання, як: Автоматичні приймальні тести з Selenium; Структура й основні елементи тестів; Організація тестів; Складові команд тестів; Ієрархія тестів і складання тестів з компонентів; Режими Selenium.

3.1 Автоматизіція тестування Web-поектів

Боротьба з помилками в програмному забезпеченні - нелегка праця. Сотні і тисячі спеціалістів з якості програмного забезпечення намагаються виявити дефекти ПЗ до запуску в комерційне використання, інакше вартість помилки зросте в тисячу разів. Понизити витрати на забезпечення якості продукції можна за допомогою застосування автоматизованих систем пошуку помилок [15].

Неможливо уявити собі розробку ПЗ, яка була б вільна від тих або інших помилок. По даним, опублікованим Національним інститутом стандартів (NIST 2002 RTI Project 7007.011), основна кількість помилок в продукті - 70%! - закрадається на стадії вироблення вимог і побудови дизайну. А виявляється переважна більшість дефектів або в процесі тестування (близько 60%), або вже при експлуатації (21%). Бізнес-процеси зазвичай базуються на декількох програмних продуктах, і при організації тестування необхідно переконатися не тільки в правильній роботі кожного продукту окремо, але і в їх коректній стиковці між собою, тобто мова вже йде про розширення поняття тестування на бізнес-процесси.

3.1.1 Доцільність проведення автоматизованого функціонального тестування

Питаючи: «Що автоматизувати?», необхідно спочатку відповісти на питання: «Чи доцільна автоматизація тестування в умовах проекту». Якщо відповідь «ТАК», то необхідно, виходячи з вимог до об'єкту тестування, створити план, по якому розроблятимуться автоматизовані тести. Далі наведені випадки, де, на мій погляд, доцільно застосувати автоматизацію тестування:

1. Труднодоступні місця в системі (бекенд процеси, логірування файлів, запис в базу даних)

2.Рутинні операції, такі як перебори даних (форми з великою кількістю полів, що вводяться). Автоматизувати заповнення полів різними даними і їх перевірку після збереження;

3.Валідаціонне повідомлення. Автоматизувати заповнення полів не коректними даними і перевірку на появу тій або інший валідації;

4.Довгі end-to-end сценарії;

5.Перевірка даних, що вимагають точних математичних розрахунків;

6.Перевірка правильності пошуку даних.

А також, багато що інше, залежно від вимог до тестованої системи і можливостей вибраного інструменту для тестування.

Найчастіше ми проводимо тестування потрібних застосувань на відповідність вимогам замовника. Таке тестування називається приймальним або функціональним.

Функціональне тестування обраного електронного магазину полягає в перевірці роботи каталога товарів, навігації по розділах, вибір товарів, формування корзини, оформлення замовлення, проглядання стану замовлення, отримання підтвердження про ухвалення замовлення до обробки поштою і т.д. Таким чином, перевіряється працездатність сайту, а також правильність його роботи. [15]

Автоматизованим функціональним тестуванням називають таке тестування системи, коли тестування ведеться в повністю автоматичному режимі, без безпосередньої участі людини, без знання деталей реалізації цієї системи на відповідність вимогам замовника .

Автоматизовані функціональні тести в ідеалі не повинні знати нічого про архітектуру системи, з яких компонентів вона складається, на якій мові програмування реалізована і т.д. Це не дозволить міняти архітектуру, мову, базу даних, але гарантує, що можна завжди швидко перевірити, що система працездатна.

Тестування повинне проводиться так, як якби реальний користувач працював з системою. Приймальні тести можна використовувати для будь-яких застосувань (розміру і якості реалізації). Приймальні тести можна писати як до реалізації тестованої системи, так і після.

Функціональне тестування для web-сайтів має на увазі емуляцію роботи користувача з браузером, а саме: відкриття сторінок, перехід по лінкам, заповнення і відправка форм, перевірка значень полів форм, наявність певного тексту на сторінках, отримання пошти, відправка файлів і т.д. При цьому всі дії із здійснення маніпуляції браузером повинні проводити самі тести, в автоматичному режимі, без участі людини.

3.1.2 Інструменти автоматизованого функціонального тестування

Вибір інструменту часто залежить від об'єкту тестування і вимог до тестових сценаріїв, оскільки інструменти тестування не можуть підтримувати абсолютно всі технології, використовувані при розробці проектів.

Тобто, вибір інструменту зводиться до банального методу проб і помилок. У результаті, нерідко вибирається декілька інструментів для тестування функцій. Наприклад, GUI (англ. - Graphical User's Interface — Графічний інтерфейс

користувача) перевіряється засобами Mercury WinRunner, бекенд процеси - використовуючи "Java based test tools" або інші інструменти.

Зараз в середовищі розробників жоден продукт не використовується як стандарт для функціонального тестування. Існує 2 типи продуктів для функционально-го тестування[15]:

1) Продукти, що емулюють поведінку браузера, написані на мові високого рівня, зазвичай на тій же мові, що і додаток (але не обов'язково). Як приклади можна привести httpUnit, JWebUnit, WebTester з SimpleTest та інші;

2) Продукти реалізовані на JavaScript і реалізовуючих перевірках безпосередньо засобами браузера. Як приклади можна привести Watir і Selenium.

Емулятори добре себе зарекомендували і набули достатньої популярності. В першу чергу через те, що вони з'явилися раніше програм другого типу, окрім цього емулятори можуть використовувати код, який написаний для додатку, правильно і оптимально створювати фікстури (формувати середовище для правильної роботи тестів), важливий і такий показник як висока швидкість виконання. Але емулятори браузеров не можуть на 100% справитися з одним завданням – вони не можуть виконувати JavaScript код, який використовується на сторінках сайтів. Це накладає обмеження або на процес тестування, або на процес створення сайтів. Тобто неможливо перевірити роботу сайту, неначебто це робив реальний користувач, оскільки емуляція роботи браузера – зовсім не еквівалентна реальній роботі браузера. Через це автоматизоване тестування доводиться доповнювати ручним тестуванням.

Цього недоліку позбавлені продукти для функціонального тестування другого типу. Вони контролюють роботу браузера, виконують команди і роблять перевірки за допомогою JavaScript-коду, що практично гарантує 100% адекватність автоматизованого тестування ручному. На жаль, до недавнього часу у розробників не було можливості користуватися хорошим, зручним, а головне безкоштовним продуктом другого типу. Але тепер є продукт “Selenium”, опис якого буде наведено у наступному підрозділі.

3.1.3 Використання інструменту Selenium для автоматизованого функціонального тестування

Selenium- це інструмент для тестування Web-проектів [16].

Selenium як проект був розпочат в червні 2004 року, а вже в грудні 2004 року він став відкритим. Проект вела компанія Thoughtworks власником якої є Мартін Фаулер — автор ряду книг і статей про архітектуру ПО, об'єктно-орієнтованому аналізу й розробці, мові UML, рефакторингу, екстремальному програмуванню.

Selenium - це об'єктно-орієнтований Javascript додаток, який може аналізувати файли певної структури для того, щоб знаходити в них команди для маніпуляції браузером і команди для виконання певних дій і перевірок. Selenium підтримується Microsoft Internet Explorer, Mozilla Suite і Mozilla Firefox для Microsoft Windows, GNU/Linux і Apple Macintosh [16].

У рамках проекту Selenium також випускається інструмент Selenium IDE, що представляє собою версію досить популярної бібліотеки Selenium в GUI-бв'язці. Цей інструмент дозволяє записувати й відтворювати скрипти, що представляють собою звичайні HTML -сторінки з однієї таблицею команди, що містить.

1) Автоматичні приймальні тести з Selenium [17]

Приймальні тести звичайно включають неавтоматичні завдання відкриття браузера й виконання дій, описаних в тест-кейсах. Вручну виконання завдання займає багато часу й має більшу ймовірність помилки оператора. Тому, автоматизація цих завдань (де це тільки можливо), допоможе виключити людський фактор. Саме тут і з'являється необхідність у перевірочних інструментальних засобах подібних Selenium. Selenium допоможе автоматизувати приймальні тести й створити краще випробування й, отже, більш надійне й зручне для обслуговування програмне забезпечення.

– Приймальне тестування - це спосіб перевірки й контролю за тим, щоб робота web-проекту відповідала функціональним, нефункціональним і іншим важливим вимогам.

– Тести Selenium виконуються безпосередньо в браузері, як роблять звичайні користувачі. Жодне інший засіб тестування не покриває такий значний масив платформ.

Є багато інших переваг використання Selenium і виконання тестів у браузері. Це два основні:

– Створюючи тестові сценарії Selenium, які відтворюють дії користувача, тестується проект з погляду кінцевого користувача.

– Запустивши тести в різних браузерах буде легше визначити несумісність браузера.

Основний елемент Selenium, також відомий як основа браузера (browser bot), написаний в Javascript. Це дозволяє сценаріям тестів реалізуватися в підтримуваних браузерах. Основа браузера виконує команди, отримані від сценаріїв тесту, які написані або в HTML, з використанням схеми таблиці, або підтримуваною мовою програмування.

2) Структура й основні елементи тестів Selenium

Тест-кейси Selenium – це звичайні html-сторінки що містять таблицю команд. Кожний рядок таблиці містить 3 колонки. Перша з них є дією або перевіркою (action і assertion/check), друга – іменем елемента (target), до якого застосовується команда, і третя – значенням (value). Далі наведений приклад тест-кейса Selenium - Test Login:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <meta content="text/html; charset=win-1251" http-equiv="content-type">
 <title>Test Login</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
 <tbody>
  <tr>
   <td rowspan="1" colspan="3">Test Login<br>
   </td>
  </tr>
  <tr>
    <td>open</td>
    <td>/login</td>
    <td> </td>
  </tr>
```

```
    <tr>
      <td>type</td>
      <td>login</td>
      <td>asdasd@gmail.com</td>
    </tr>
    <tr>
      <td>type</td>
      <td>password</td>
      <td>dfsdfs</td>
    </tr>
    <tr>
      <td>clickAndWait</td>
      <td>login_button</td>
      <td> </td>
    </tr>
    <tr>
      <td>verifyLocation</td>
      <td>/</td>
      <td> </td>
    </tr>
    <tr>
      <td>verifyTextPresent</td>
      <td>Wellcome, Ivan Ivanov</td>
      <td> </td>
    </tr>
  </tbody>
</table>
</body>
</html>
```

Отже, що ж робить Selenium, виконуючи даний тест-кейс? Варто відразу вказати, що Selenium пропускає перший рядок таблиці.

```
<tr>
    <td>open</td>
    <td>/login</td>
    <td> </td>
</tr>
```

Виконується дія “Open”. Selenium дає команду браузеру перейти на зазначену сторінку - /login. Колонка для значення тут не використовується.

Тепер Selenium за допомогою дій type заповнить поля форми певними значеннями.

```
<tr>
    <td>type</td>
    <td>login</td>
    <td> asdasd@gmail.com </td>
  </tr>
  <tr>
    <td>type</td>
    <td>password</td>
    <td>dsffds</td>
  </tr>
```

Після цього дією clickandwait Selenium натисне кнопку для відправлення форми й буде чекати відповіді.

```
<tr>
    <td>clickAndWait</td>
    <td>login_button</td>
    <td> </td>
  </tr>
```

Після цього за допомогою перевірки verifylocation Selenium перевірить поточне положення браузера. Після аутентифікації відбувається перехід на головну сторінку сайту.

```
<tr>
    <td>verifyLocation</td>
    <td>/</td>
    <td> </td>
  </tr>
```

І, нарешті, після успішної аутентификации повинне з'явитися запрошення на головній сторінці. Selenium перевіряє наявність відповідного тексту на сторінці за допомогою перевірки veritytextpresent.

```
2.      <tr>
3.        <td>verifyTextPresent</td>
4.        <td>Wellcome, Ivan Ivanov</td>
5.        <td> </td>
6.      </tr>
```

3) Організація тестів Selenium [18]

Тест-кейси організовані в список, який називається Testsuite.

Всі файли тест-кейсів і Testsuite-ів є звичайними текстовими html-файлами. Хоча Selenium

сприймає тест-кейси тільки у вигляді html-сторінок, це зовсім не означає, що самі тест-кейси повинні бути html файлами. Вони можуть генеруватися за допомогою будь - якої технології, яка може здатися зручною.

4) Складові команд тестів

В Selenium існує кілька основних понять:

– Дії;

– Перевірки;

– Локатори.

Дії (actions) використовуються для того, щоб управляти браузером з-під Selenium. Набір дій досить широкий. Нижче наведен список найбільш часто використовуваних дій. У дужках буде дана форма застосування у форматі wiki-table. Отже от деякі з дій Selenium:

– open – указує браузеру відкрити сторінку по певній адресі (|open|location||);

– click – указує браузеру натиснути на лінк або по елементу форми, наприклад, для позначки checkboxha (|click|target||);

– type – указує браузеру ввести нове значення в елемент форми (|type|element|value|);

– select – указує браузеру вибрати певне значення <option> усередині <select> тегу форми(|select|element|value);

– selectwindow – указує браузеру перемкнути фокус на інше вікно (|selectwindow|window_name||);

– goback – указує браузеру повернутися на попередню сторінку;

– close – указує браузеру закрити поточне вікно.

Повний список і приклади використання дій надається в документації, що поставляється разом з Selenium.

Перевірки (checks) використовуються для перевірки правильності поведінки додатка. Будь-яка перевірка в Selenium представлена двома типами – assert і verify. Якщо в тесті стоїть перевірка assertsomething і вона не виконується, тест припиняє свою роботи, а якщо verifysomething – то видається повідомлення про помилку, але тест продовжує роботу. Далі наведено список найбільш використовуваних перевірок:

– verifylocation / assertlocation – перевіряє поточний URL вікна браузера (|verifylocation|url_needed||);

– verifytitle / asserttitle – перевіряє заголовок вікна (|verifytitle|title_needed||);

– verifyvalue / assertvalue – перевіряє, що поле форми має зазначене значення (|verifyvalue|field|value_needed|);

– verifytextpresent / asserttextpresent – перевіряє, що текст сторінки містить зазначений текст (|verifytextpresent|text_needed||);

– verifyelementpresent / assertelementpresent – перевіряє, що елемент присутній на поточній сторінці (|verifyelementpresent|element_needed||).

Повний список і приклади використання дій надається в документації, що поставляється разом з Selenium.

Локатори використовуються для знаходження елементів, до яких відносяться команди. Список локаторів Selenium досить великий:

– id – використовується атрибут ідентифікатору елемента;

– name – використовується атрибут імені елемента;

– identifier – використовується атрибут id елемента, якщо по id елемент не знайдений, то пошук буде вестися по атрибуту name;

– dom – використовується для пошуку елемента по DOM вираженню, яке повинне починатися з document;

– xpath – використовується для пошуку елемента по Xpath вираженню, які повинне починатися з //;

– link – використовується для знаходження лінків із зазначеним текстом.

Для заповнення поля login форми login_form можна скористатися наступними командами [18]:

Використовування локатору id.

```
<tr>
      <td>type</td>
Type id=login</td>
      <td> asdasd@gmail.com </td>
</tr>
Використовування локатору name.
 <tr>
      <td>type</td>
type name=login</td>
      <td> asdasd@gmail.com </td>
</tr>
Використовування локатору identifier.
<tr>
      <td>type</td>
type identifier=login</td>
      <td> asdasd@gmail.com </td>
</tr>
Використовування локатору xpath.
 <tr>
      <td>type</td>
type xpath=//input[@name=’login’]</td>
      <td> asdasd@gmail.com </td>
</tr>
Використовування локатору dom.
 <tr>
      <td>type</td>
typedom=document.forms[‘login_form’].login</td>
      <td> asdasd@gmail.com </td>
```

```
</tr>
```

Якщо в команді не зазначена явна назва типу локатору, тоді Selenium послідовно намагається знайти елемент за допомогою наступних типів локаторів:

– identifier;

– dom;

– xpath.

5) Ієрархія тестів і складання тестів з компонентів

Як уже було сказано вище, для написання тестів для Selenium можна використовувати будь-які засоби, які в остаточному підсумку дають просту html таблицю команд.

Якщо говорити про більш високий ступінь вкладеності тестів, то тут потрібно сказати, що Testrunner.html може ухвалювати атрибут test для безумовного завантаження Testsuite, наприклад: http://Eshop/selenium/Testrunner.html?test=suites/Allthetests.html

Таким чином, можна створювати групи тестів і за допомогою якої-небудь технології, формувати будь-який набір тестів, який є необхідним і видати Selenium той Testsuite, який потрібний у даний момент.

6) Режими Selenium [18]:

Selenium використовується у двох режимах: test runner і driven. Ці два режими відрізняються по своїй складності й способу створення. Сценарії driven тестів створювати складніше, оскільки вони записані мовою програмування. По складності вони будуть відрізнятися незначно, якщо використовувати динамічна мова програмування високого рівня.

Головна відмінність цих режимів полягає в тому, що тести частково запускаються поза браузером, якщо використовувати сценарії driven тестів, у яких сценарії test runner повністю запускаються в браузері.

– Режим test runner:

Сценарії Selenium test runner, також відомі як тест-кейси, написані в HTML з використанням простої схеми таблиці.

Сценарії test runner звичайно використовуються на одному сервері в якості додатку під тестом. Це відбувається, тому що основа браузера використовує Javascripts для імітування дій користувача. Тест-екйси і команди виконуються в послідовності, у якій вони з'являються в тестовому наборі й тест-екйсах.

Для одержання повного тестового покриття звичайно потрібно більше чим один тест-екйс. Тому Selenium використовує принцип тестових наборів. Тестові набори використовуються для того, щоб об'єднати тест-екйси зі схожими функціональними можливостями, так, щоб їх можна було запустити в певній послідовності.

Тестові набори складені по тому ж принципу що й тест-екйси, тобто, використовуючи прості таблиці HTML. Тестовий набір Selenium, що використовується за замовчуванням називається Testsuite.html. Далі показаний тестовий набір, який тестує додаток точно так, як це робить звичайний користувач. Потрібно відзначити, що тестові набори застосовують одноколонну таблицю й кожний рядок указує на файл утримуючий тест-екйс.

```
<table>
    <tr>
      <td>Test suite for the application</td>
    </tr>
```

```
        <tr>
          <td><a href="test_main_page.html">Access main page</a></td>
        </tr>
        <tr>
          <td><a href="test_login.html">Login to application</a></td>
        </tr>
        <tr>
          <td><a href="test_address_change.html">Some actions in
application</a></td>
        </tr>
        <tr>
          <td><a href="test_logout.html">Logout from application</a></td>
        </tr>
      </table>
```

– Режим driven:

Сценарії driven Selenium написані на одному з підтримуваних мовах програмування — на даний момент існують драйвера з Java, Ruby і Python. Ці сценарії запускаються окремо поза браузером. Завдання драйверів полягає у виконанні сценарію тестів і керуванні браузером за допомогою з'єднання з основою браузера, яка запускається в браузері. Для з'єднання між драйвером і основою браузера використовується простій, спеціальний для Selenium провідна мова, яка називається Selenese.

Сценарії driven могутніші й гнучкі чим сценарії test runner, також можна поєднувати їх зі структурами xunit. Недолік сценаріїв driven (якщо порівнювати зі сценаріями test runner) полягає в тому, що їх складніше використовувати й встановлювати. Причиною тому є наступні завдання, які повинен виконувати драйвер:

– Запуск сервера;

– Застосування додатка під тестом;

– Використання сценарію тесту;

– Запуск браузера;

– Передача команд основі браузера;

– Перевірка результатів команд виконаних основою браузера.

– Крім того сценарії driven більше залежать від умов прогону додатку.

3.2 Ситеми відстеження помилок (багтрекіногові системи)

Баг (англ. bug—жук) — слово, що позначає помилку в программі. Термін звичайно вживається відносно помилок, що проявляють себе на стадії роботи програми, на відміну, наприклад, від помилок проектування або синтаксичних помилок. «Баги» локалізуються й усуваються в процесі тестування програми [18].

По легенді, 9 вересня 1945 року вчені Гарвардського університетау, які тестували обчислювальну машину Mark II Aiken Relay Calculator, знайшли метелика, що застряг між контактами електромеханічного реле і Грейс Хоппер вимовила цей термін. Витягнута комаха була вклеєна в технічний щоденник, із супровідним написом: «First actual case of bug being found» (англ. «перший випадок у практиці, коли був виявлений жучок»). Цей забавний факт поклав початок використанню слова «баг» у значенні «помилка».

У дійсності цей випадок відбувся 9 вересня 1947, а не 1945, року. Слово «bug» у сучасному значенні вживалося задовго до цього. Так, протягом Другої світової війни словом «bugs» називалися проблеми з радарною електронікою.

Система відстеження помилок (англ. Bugtracking system)— прикладна програма, розроблена з метою допомогти розроблювачам програмного забезпечення ураховувати й контролювати помилки (баги), знайдені в програмах, а також стежити за процесом усунення цих помилок ().

Головний компонент такої системи — база даних, яка утримує відомості про виявлені помилки. Ці відомості можуть містити в собі:

- хто повідомив про проблему;
- дата й час, коли була виявлена проблема;
- серйозність (критичність) проблеми;
- опис неправильного поводження програми;
- хто займається усуненням проблеми;
- стан помилки.

Типова система відстеження помилок використає концепцію «життєвого циклу» помилки, що відслідковується по стані, у якому перебуває помилка. Система може надавати адміністраторові можливість настроїти, які користувачі можуть переглядати й редагувати помилки залежно від їхнього стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використатися для одержання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає досить точних результатів, через те що різні помилки мають різний ступінь серйозності й складності. При цьому серйозність проблеми не має прямого відношення до складності усунення помилки.

3.2.1 Різновид багтрекінгових систем

Багтрекіногових систем існує величезна кількість. Далі будуть наведені системи відстеження помилок, які найбільш поширені в багатьох компаніях та корпораціях. [23]

Atlassian JIRA — система відстеження помилок, призначена для організації спілкування з користувачами, хоча в деяких випадках систему можна використати для керування проектами. Розроблена компанією Atlassian Software Systems. Платна. Має веб-інтерфейс. Назва системи (JIRA) було отримано шляхом модифікації назви конкуруючого продукту - Bugzilla. Система дозволяє працювати з декількома проектами. Для кожного із проектів створює й веде схеми безпеки й схеми оповіщення [19].

BUGS - the Bug Genie – безплатна система відстеження помилок і завдань із веб-інтерфейсом [19].

Bugzilla — безплатна система відстеження помилок із веб-інтерфейсом.

Bugzilla — це добре продумана система, з однієї сторони вона досить проста, з іншого боку, там є все, що потрібно для багтрекінгу типового проекту [26].

Trac — інструмент управління проектами та відстеження помилок у програмному забезпеченні. Trac є відкритим програмним забезпеченням, розробленим і підтримуваним компанією Edgewall Software. Trac дозволяє організувати перехресні гіперсилки між базою даних зареєстрованих помилок та системою управління версіями. Це дає можливість використати Trac у тому числі і як веб-інтерфейс для доступу до системи контролю версій .

TrackStudio Enterprise — платна система відстеження помилок, призначена для керування внутрішніми процесами в компаніях-розроблювачах програмного забезпечення. Використовується для керування проектами, помилками, вимогами. Розроблена компанією "ГРАН". Підтримує англійський, російський, український, німецький і китайський інтерфейс користувача .

Окрему увагу хочеться приділити найбільш розповсюдженій та дуже зручній системі відстеження помилок – Mantis, та розглянути її більш детально ніж інші системи у наступному підрозділі.

3.2.2 Опис системи відстеження помилок “Mantis”

Mantis — вільно розповсюджувана система відстеження помилок у програмних продуктах (bugtracker). Забезпечує взаємодію програмістів зі спеціалістами з якості програмного забезпечення. Дозволяє заводити повідомлення про помилки й відслідковувати подальший процес роботи над ними [21].

Система має гнучкі можливості конфігурування, що дозволяє відбудовувати її не тільки для роботи над програмними продуктами, але і як система обліку заявок для helpdesk.

Система побудована за принципом «клієнт-сервер», тому не вимагає для роботи встановлення спеціального ПО й працює через веб-браузер.

Переваги і можливості Mantis:

- Безкоштовна система;
- Легкість інсталяції;

- Веб-сервер-інтерфейс і платформонезалежність;
- Багатомовність;
- Інтеграція з поштовою системою;
- Пошук;
- Система фільтрів/запитів;
- Настроюваність/розширюваність (ядро, модулі, кастомізация);

Ключовим поняттям системи (як і у всіх системах-трекерах) є питання («Issue») — деяке завдання, питання, запит, звернення, рекламація з приводу помилки в системі, або просто повідомлення, що вимагає зворотного зв'язку, і призначення системи — реєстрація і надання зацікавленим особам цілісної інформації про стан цього «питання», включаючи інтерфейси редагування, запиту і пошуку, механізми поштового і RSS-сповіщень. В більшості випадків, в системах-трекерах, під питанням розуміється баг. Суть «Баг» має набір атрибутів, робота з якими — редагування і запити — є основними сценаріями використання Mantis. Далі будуть представлені наступні атрибути бага :

– «Ініціатор» («Reporter») - зареєстрований користувач, який створив запит/баг. Теоретично поле редаговане, але зловживати редагуванням небажано (тільки для виправлення помилки).

– «Видимість» («View Status») - видимість питання. Вона може бути загальною або обмеженою. Необхідна для забезпечення конфіденційності серйозних питань.

– «Категорія» («Category») - функціональна або організаційна частина проекту, до якої відноситься запити.

– Існує різна безліч категорій:

– Bug — помилка в програмному продукті;

– Change Request — запит на зміну (як правило запити на зміни в системі поступають від Замовника);

– Database — помилки, що виникають у базі даних;

– Design — помилки, пов'язані з дизайном продукту;

– Logic — помилки, пов'язані з порушенням логіки в системі, що розробляється;

– Missing Functionality — відсутність функціональності;

– Proposal — пропозиції, висловлювані розробниками, фахівцями за якості продукту, замовників, менеджерів проектів, які можуть бути направлені на додавання або зміну яких-небудь фунциональностей, дизайну і т.п.;

– Question — питання, що виникають у розробників, фахівців з якості, замовників, менеджерів проектів і т.п.

– Task — завдання, які необхідно реалізувати в якійсь конкретній версії або фазі проекту;

Usability — помилки, пов'язані з поняттям «юзабіліті» (незручна навігація, відсутність заголовків сторінок, складність інтерфейсу і т.д).

– «Відтворюваність» («Reproducibility»):

– «завжди» («always»);

– «іноді» («sometimes»);

– «довільно» («random»);

– «не перевірялася» («have not tried»);

– «не відтворюється» («unable to reproduce»);

– «непридатна» («N/A»).

– «Стан» («Status»):

– «нове»(«new») - новий запит;

– «потрібний відгук» («feedback») - потрібна додаткова інформація, ініціаторам запиту потрібно проявити увагу.

– «розглянутий» («acknowledged») - з запитом ознайомилися, але підтвердження (бага) ще не було, відповідальний не призначений.

– «підтверджений» («confirmed») - confirmed and reproducible (typically set by an Updater or other Developer)

– «призначений» («assigned») - запит призначений розробникові;

– «відпрацьований» («resolved») - запит ніби вирішений , очікується підтвердження що все працює добре;

– «закритий» («closed») - запит закрит.

– «Серйозність» («Severity»)

– «block» - блокує велику частину роботи в проекті;

– «crash» - приводить до «падіння» додатку (або навіть операційної системи);

– «major» - баг, важливість велика;

– «minor» - баг, але мала важливість;

– «tweak» - незручність у використанні — потрібна «підгонка»;

– «text» - невелика текстова помилка/друкарська помилка;

– «trivial» - дрібні причіпки;

– «feature» - запит нового функціонала.

– «Платформа» («Platform»);

– «Операційна система» («OS»);

– «Версія ОС» («OS Version»);

– «Трудомісткість» («Projection»);

– «Термін» («ETA»);

– «Збірка продукту» («Product Build»)

– Поля що настроюються («Custom Fields»).

Поля, що настроюються, — це додаткові до основних, атрибути, які можна «активувати» в окремих проектах.

– Проект

Суть «Проект» призначена для тематичного угрупування питань, багів і регулювання доступу до них користувачів. Інтерфейс управління проектами доступний користувачам з достатніми повноваженнями.

Атрибути проекту наступні:

– «Назву проекту» («Project Name») - може змінюватися після створення;

– «Стан» («Status»)

– «випущений» («release»);

– «стабільний» («stable»)

– «застарілий»,

– що «розробляється» («development») («obsolete»)

– «Видимість» («View Status»)

– «загальна» («public»): проекти видно всім користувачам системи;

– «обмежена» («private»): такі проекти видно тільки користувачам, вибраним для цього проекту, або що має достатні привілеї («адміністратори»), щоб бачити «private» проекти.

– «Моє зведення» («My View»)

– «Список питань» («View Issues») - основне вікно для вибору і проглядання списку (таблиці) запитів.

Для вибору підмножини запитів застосовуються так звані фільтри.

Можна використовувати заздалегідь приготовані особисті або загальні фільтри, що зберігаються, або створити новий фільтр безпосередньо на цій сторінці — інтерфейс побудови запиту показується над таблицею питань. Поле «Пошук:» («Search:») використовується для пошуку ключового слова в атрибутах «Суть»/ («Summary»), в описі, в ідентифікаторі (у коментарях не шукає). Список вибраних запитів показується у вигляді таблиці, де набір стовпців фіксований в наступному порядку:

– «Р» - пріоритет;

– «Номер» («ID») - ісловий ідентифікатор питання;

– «Число коментарів»;

– «Категорія» («Category») - категорія запиту;

– «Серьозність» («Severity») - виділяється жирним, якщо серйозність висока («major», «crash», «block»), а питання не вирішене.

– «Статус» («Status») Крім стану питання, якщо провести над цим полем мишею, з'явиться спливаюча підказка з атрибутом;

– «Змінений» («Updated») - дата останньої зміни.

– «Суть» («Summary») - опис запиту. відмовитися від відстежування

кнопкою Не відстежувати. Текст цього нагадування буде збережений в питанні.»

– «Друк» («Print») - виводить на «версію для друку» даного запиту;

– «Змінити питання» («Update Issue») перехід на сторінку редагування всіх атрибутів.

– «Призначити» («Assign To») - зміна відповідального за це питання (вибирається з випадного списку).

– «Змінити статус на» («Change Status To») - зміна стітусу (вибирається з випадного списку). Після натиснення на цю кнопку з'явиться додаткове вікно введення коментаря (що «пояснює» зміну) і дозапросу деяких атрибутів, наприклад:

– «Номер дубля» («Duplicate ID») — якщо питання закривається, як дублюючий;

– «Відповідальний»(«Assigned to») — якщо питання розглянуте, і призначається відповідальний;

– «Вирішений у версії» («Fixed in Version») — для вирішених9595 питань.

– Додаткові поля, видимі при зміні або рішенні питання.

– «Відстежувати» («Monitor/Unmonitor Issue») стеження за змінами питання за допомогою поштових сповіщень;

– «Клонувати» («Create Clone») створити копію поточного питання.

– Користувач прямує на сторінку

– «Создати запит» («Report Issue»), де всі поля заповнені аналогічно поточному запиту, а також преполагається зв'язок з «початковим» питанням.

– «Повторно відкрити» («Reopen Issue») - видний тільки для закритих питань, призначений для повторного їх відкриття.

– «Перемістити питання» («Move Issue») - переміщення запиту;

– «Видалити питання» («Delete Issue») - остаточне видалення питання.

Рекомендується цим ніколи не користуватися, хіба що запит був, умовно кажучи, «сміттєвий» (або містив конфіденційну інформацію, яку не можна було

поміщати в систему). Натомість треба встановити, що питання дозволене, і виставити відповідне рішення.

Нижче знаходиться панель «Зв'язку» («Relationships»), що відображає зв'язки питання з іншими питаннями. Можуть бути наступні типи зв'язків:

«залежить від/блокує» («parent of/child of») - користувач попереджатиметься, при спробі закрити «батьківський» запит, якщо запити -"потомки" ще не вирішені.

«пов'язаний з» («related to») - рівноправний інформаційний зв'язок.

«дублює/має дубль» («duplicate of/has duplicate») - в'язує питання-дублікати.

Далі розташовується панель тих співробітників, що відстежують це питання. Ще нижче — форма для завантаження файлових вкладень. Далі — форма введення коментаря і список раніше введених коментарів. І завершує сторінку історія змін питання: «Змінити питання» («Bug Update») - налогічно сторінці «Просмотр питання» («Viewing Issue»), включаючи «простій» і «розширений» види, тільки тут всі атрибути питання редактіруємі, і є можливість видаляти коментарі або обмежувати їх видимість.

4 ОХОРОНА ПРАЦІ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблено заходи з техніки безпеки та рекомендації з пожежної профілактики. Завданням даної роботи бакалавра було проаналізувати методи та засоби автоматизації тестуваннятему Так як в процесі проектування використовувався комп’ютер, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера який використовувався для пошуку інформації і написання дипломної роботи.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності. При роботі з обчислювальною технікою змінюються фізичні і хімічні фактори навколишнього середовища: виникає статична електрика, електромагнітне випромінювання, змінюється температура і вологість, рівень вміст кисню і озону в повітрі.

Повітря забруднюється шкідливими хімічними речовинами антропогенного походження за рахунок деструкції полімерних матеріалів, які використовуються для обробки приміщень та обладнання. Неправильна організація робочого місця сприяє загальному і локальній напрузі м'язів шиї, тулуба, верхніх кінцівок, викривлення хребта і розвитку остеохондрозу. На всіх підприємствах, в установах, організаціях повинні створюватися безпечні і нешкідливі умови праці. Забезпечення цих умов покладається на власника або уповноважений ним орган (далі роботодавець).

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. Роботодавець повинен впроваджувати сучасні засоби техніки безпеки, які запобігають 44 виробничому травматизмові, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників. Він не має права вимагати від працівника виконання роботи, поєднаної з явною небезпекою для життя, а також в умовах, що не відповідають законодавству про охорону праці. Працівник має право відмовитися від дорученої роботи, якщо створилася виробнича ситуація, небезпечна для його життя чи здоров'я або людей, які його оточують, і навколишнього середовища

4.3 Аналіз умов праці у приміщенні

Робота над аналізом засобів автоматизації проходила у стінах звичайної квартири, тому у розділі будуть описані засоби розглянуті саме для квартири. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі

стаціонарним комп’ютером, та необхідними компонентами для розробки та збору системи до купи.

4.3.1 Оцінка санітарно – гігієнічних умов праці

Нижче наведена детальна інформація і характеристики робочого приміщення інформація у таблиці 4.1

Таблиця 4.1 – Основні розміри кімнати

Параметр	Величина
Довжина, м	5
Ширина, м	3
Висота, м	2.7
Кількість робочих місць	1
Площа, $м^2$	15
Об’єм, $м^3$	40,5

Розробимо приблизну схему приміщення, за допомогою якої можна наглядно зрозуміти стан умов у яких виконувалася дипломна робота.

Рисунок 4.1 План робочого приміщення

Згідно з [23] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам. Розрахуємо фактичні значення цих показників, розділивши загальну площу та об'єм приміщення на кількість працюючих:

$$S' = \frac{S}{N} = \frac{15}{1} = 15$$

$$V' = \frac{V}{N} = \frac{40.5}{1} = 40.5$$

Отже, за характеристиками площі і об'єму приміщення відповідає нормам. Параметри вікон: Висота – 1.5м. Ширина – 3м. Відстань від підлоги – 0.7 м. Вікна виходить на захід, можуть відкриватися та мають штори. Двері відчиняються назовні, ширина коридору 3 м, висота до перекриття 2 м. Ширина дверей у приміщені 0,8 м. У освітленні приміщення, що розглядається, застосовується бокове природне освітлення (вікна: висота = 1.5 м, ширина = 3 м), штучне, створюване електричними лампами (2 світлодіодні лампи). Розглянемо тепер відповідність характеристик робочого місця нормативним. Для цього зведемо основні вимоги до організації робочого місця і відповідні фактичні значення для робочого місця, за яким виконується робота, у табл. 5:

Таблиця 5 – Фактичні та нормовані значення параметрів кімнати

Найменування параметра	**Значення**	
	Фактичне	**Нормативне**
Висота робочої поверхні, мм	**750**	**680-800**
Висота простору для ніг, мм	**600**	**>600**

Ширина простору для ні, мм	**500**	**>500**
Глибина простору для ніг, мм	**700**	**>650**
Висота поверхні сидіння, мм	**420**	**400-500**
Продовження на наступній сторінці		
Ширина сидіння, мм	**550**	**>400**
Глибина сидіння, мм	**500**	**>400**
Висота поверхні спинки, мм	**900**	**>300**
Ширина опорної поверхні, мм	**500**	**>380**
Радіус кривини спинки в горизонтальній площині, мм	**400**	**400**
Відстань від очей до дисплею, мм	**800**	**700-800**

Робочий стіл на досліджуваному місці також містить достатньо простору для ніг. Крісло, що використовується в якості робочого сидіння, є підйомно-поворотним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки. Екран монітору знаходиться на відстані 0.8м, клавіатура має можливість регулювання кута нахилу 5-15°. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам. У приміщенні знаходяться монітор Samsung S27E591CS. На все обладнання є паспорт та інструкція по експлуатації, перекладена російською мовою. Відповідно супроводжувальній документації обладнання відповідає стандартам України і його можна використовувати без загрози здоров’ю та життю працюючого.

4.3.2 Напруженість праці користувача

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м'язово скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок. Тобто наявне психофізіологічні небезпечні та шкідливі фактори:

а) фізичного перевантаження:

– 50 - статичного;

– динамічного;

б) нервово-психічного перевантаження:

– розумового перенапруження;

– монотонності праці;

– перенапруження аналізаторів;

– емоційних перевантажень.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи на ведені в [21].

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви: (потрібне вибрати):

– для розробників програм тривалістю 15 хв через кожну годину роботи

– для операторів персональних комп’ютерів тривалістю 15 хв через дві години роботи;

– для операторів комп’ютерного набору тривалістю 10 хв через кожну годину роботи.

4.4 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при розробці та експлуатації даної мікроконтролерної системи, пожежної безпеки надалі будуть розроблені (якщо потрібні) заходи для вирішення питання необхідності забезпечення людини достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.4.1 Аналіз небезпечних та шкідливих факторів під час розробки виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл.4.2). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання [21] «Правил охорони праці під час експлуатації електронно-обчислювальних машин», які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої. Основними робочими характеристиками персонального комп'ютера є:

– робоча напруга U=+220В +-5%;

– робочий струм I=2А;

– споживана потужність P=350 Вт.

Таблиця 6 Аналіз небезпечних і шкідливих виробничих факторів

Таблиця на наступній сторінці

Небезпечні і шкідливі фактори	Джерела факторів (види робіт)	Кіл-на оцінка	Небезпечні і шкідливі фактори
1	2	3	4
фізичні			
Підвищена рухливість повітря	Відчинене вікно	1	ДСН 3.3.6.042-99
психофізіологічні			
Нервово-психічна, перенавантаження (розумове, перенапруження очей)	пошук інформації для постановки теми; пошук та аналіз аналогів і літератури; пошук наявних технологій моделювання та аналіз алгоритмів; виконання роботи за темою диплома; оформлення роботи	4	НПАОП 0.00-1.28-10 ДСанПіН 3.3.2.007-98

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [22]. За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі

мікрокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом, яких можуть бути: принтер, сканер та інші джерела виділення багатьох хімічних речовин - напр., озону, оксидів азоту та аерозолів високодисперсних частинок тонера), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

4.4.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання,. Небезпека загоряння пов'язана з особливістю комп'ютерів - із значною кількістю щільно розташованих на монтажній 53 платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Надійна робота окремих елементів і мікросхем в цілому забезпечується тільки в певних інтервалах температури, вологості і при заданих електричних параметрах. При відхиленні реальних умов експлуатації від розрахункових можуть виникнути пожежонебезпечні ситуації.

Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80…100 ℃). При проходженні електричного струму по провідниках і деталей виділяється тепло, що в умовах їх високої щільності може привести до перегріву, і може служити причиною запалювання ізоляційних матеріалів. Слабкий опір ізоляційних матеріалів дії температури може викликати порушення ізоляції і привести до короткого замикання між струмоведучими частинами обладнання (шини, електроди). Також ймовірна небезпека внаслідок перевантаження напруги, розрядки зарядів статичної електрики, пошкодження обладнання та електропроводки. Електростатичний розряд виникає під час тертя двох ізольованих матеріалів. Розряд статичної електрики може виникнути під час роботи вентилятора або комп'ютера. Кабельні лінії є найбільш пожежонебезпечними місцем. Наявність пального ізоляційного матеріалу, ймовірних джерел запалювання у вигляді електричних іскор і дуг, розгалуженість і недоступність роблять кабельні лінії місцем найбільш ймовірного виникнення і розвитку пожежі.

4.4.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.5 Гігієнічні умови

4.5.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт Іа. Мікроклімат приміщення, визначається наступними параметрами:

- **температура повітря, t (0 С);**
- **відносна вологість повітря, ϕ (%);**

– швидкість руху повітря, v (м/с);

– інтенсивність теплового випромінювання, j (Вт/м 2);

– температура поверхонь будівельних конструкцій, tп (0 С).

Перші три параметри встановлюються відповідно до пори року і категорії роботи за енерговитратами. Робота оператора ЕОМ, яка розглядається, виконується сидячи і не потребує фізичного напруження; витрати енергії становлять до 120 ккал/год. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають [22] і наведені у табл. 5:

Таблиця 4.3 – Оптимальні параметри мікроклімату

Пора року	**Категорія робіт**	**Температура повітря, С**	**Відносна вологість, %**	**Швидкість руху повітря, м/с**
		Оптимальна	**Оптимальна**	**Оптимальна**
Холодна	**Легка – 1а**	**22-24**	**40-60**	**0,1**
Тепла	**Легка – 1а**	**23-25**	**40-60**	**0,1**

Слід зазначити, що у приміщеннях з ЕОМ рекомендується дотримуватися саме оптимальних параметрів мікроклімату, тобто таких, які забезпечують відчуття теплового комфорту та створюють передумови для високого рівня працездатності.

Температура повітря у приміщенні, що розглядається, визначається температурою атмосферного повітря і джерелами виділення тепла. Ними є

електрообладнання, сонячна радіація і теплота, яку виділяє організм людини. Сумарна кількість теплоти, що виділяється у приміщенні, не призводить до 65 виходу температури за встановлені межі. Суттєвого підвищення температури внаслідок дії сонячної радіації вдається уникнути, закривши вікна шторами; проникаюча радіація не спричиняє будь-якого помітного теплового ефекту внаслідок низької теплопровідності будівельних конструкцій. В даному випадку приміщення обладнане системою опалення та кондиціонером.

При низьких температурах у холодну пору року стабільність температури повітря підтримує опалювальна система. Як результат, протягом року температура повітря у приміщенні не виходить за встановлені межі. Тому по цим параметрам приміщення відповідає нормам викладених у. Температура приміщення становить 24С, що відповідає нормі.[23]

4.5.2 Освітлення приміщення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності. Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

В досліджуваному приміщенні використовується система загального рівномірного штучного освітлення. Мається люстра з двома лампочками. Люстра знаходиться точно в центрі приміщення. Нижче представлено розрахунок освітлення:

$$S = \left(\frac{1}{5} / \frac{1}{10}\right) * w * l = 0.125 * 15 = 1.875\ m^2$$

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n відбувається за формулою:

$$n = \frac{E * S * Z * K}{F * U * M} = \frac{300 * 15 * 1.1 * 1.5}{5400 * 0.575 * 2} = 1.2$$

На основі цих розрахунків можна сказати, що для освітлення приміщення відповідаючого усім нормам трема змінити лампу на більш потужну.

4.6 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, що супроводжує роботу користувачів персональних комп'ютерів (зумовлений як роботою системних блоків, клавіатури, так і друкуванням на принтерах, а також зовнішніми чинниками), коливається у межах 50–65 дБА [24].

Шум такої інтенсивності на тлі високого ступеня напруженості праці негативно впливає на функціональний стан користувачів. Тому на практиці рекомендують знижувати фактичний рівень шуму у приміщеннях, де створюють комп'ютерні програми, виконують теоретичні та творчі роботи, проводять навчання до 40 дБА, а в приміщеннях, де виконують роботу, що потребує зосередженості, — до 55 дБА. У залах опрацювання інформації та комп'ютерного набору рівні шуму не повинні перевищувати 65 дБА. Шум часто є причиною зниження рівня працездатності, підвищення рівня загальної та професійної захворюваності, частоти виробничих травм. Шум є загально біологічним подразником, який негативно впливає на всі органи і системи організму. У разі

тривалого систематичного впливу шуму може виникнути патологія з переважним ураженням слуху, центральної нервової і серцево- судинної систем. 60 Для зниження шуму на шляху його поширення передбачається розміщення в приміщенні штучних поглиначів.

Для зниження рівня шуму стелю або стіни вище 1.5 - 1.7 метра від підлоги повинні облицьовуватися звукопоглинальним матеріалом з максимальним коефіцієнтом звукопоглинання в області частот 63-8000 Гц. Додатковим звукопоглинанням в КВТ можуть бути фіранки, підвішені в складку на відстані 15-20 см. Від огорожі, виконані з щільної, важкої тканини. У приміщенні з ЕОМ коректований рівень звукової потужності не перевищує 45 дБА.

Оскільки рівень шуму не перевищує гранично допустимих величин, які встановлені санітарними нормами, заходи для зниження шуму не проводяться.. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам [25]. Допустимий рівень вібрацій на робочому місці: - для 1 ступеня шкідливості до 3 дБ; - для 2-3 - 1-6 дБ; - для 3 - більше 6 дБ.

4.7 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

– - правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

– - експлуатацію сертифікованого обладнання;

– - дотримання заходів електробезпеки;

– - забезпечення оптимальних параметрів мікроклімату;

– - забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

– - облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

а) якщо об'єм приміщення 20 м3 , то потрібно подати не менш як 30 м3 /год повітря;

б) якщо об'єм приміщення у межах від 20 до 40 м3 , то потрібно подати не менш як 20 м3 /год повітря;

в) якщо об'єм приміщення становить понад 40 м3 , допускається природна вентиляція, у випадку, коли немає виділення шкідливих речовин.

- зниження рівня шуму та вібрації:

а) у джерелі виникнення, шляхом застосування раціональних конструкцій, нових матеріалів і технологічних процесів;

б) звукоізолювання устаткування за допомогою глушників, резонаторів, кожухів, захисних конструкцій, оздоблення стін, стелі, підлоги тощо;

в) використання засобів індивідуального захисту).

Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

– - постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;

– - постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

– - не тягнути за мережевий кабель, щоб витягти вилку з розетки;

– - не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

– - не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

– - не залишати включені електроприлади без нагляду;

– - не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

– - не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

4.8 Висновки

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Була наведена схема, розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВОК

На сьогодняшній час тестування є важливим процесом під час створення програмного забезпечення. Компанії інвестують велику кількість грошей на процес тестування і все одно програмне забезпечення на виході має безліч багів та недоліків.

Незважаючи на те, що роль тестування на перший погляд може здатися не такою вже значною, процес тестування програмного забезпечення є настільки ж невід'ємну частину розробки, як і проектування. Яка б методологія розробки програмного забезпечення не застосовувалася, роль процесу тестування для забезпечення якості продукту важко переоцінити.

Роль тестування ще більше зростає із застосуванням прогресивної итеративной і інкрементальною методології розробки ПО. Специфіка даної методології полягає в малій тривалості окремих етапів розробки - ітерацій. У той же час кожна ітерація включає в себе всі етапи життєвого циклу, аж до впровадження розроблюваного ПЗ і отримання реакції користувачів.

Очевидно, що такий підхід, по-перше, вимагає від тестової інфраструктури виявляти значну кількість дефектів програми, на якомога більш ранніх стадіях, по-друге, фаза впровадження програмного продукту на кожній ітерації вимагає від тестової підсистеми виявити таку кількість помилок, щоб продукт міг вступити до кінцевого користувача. Все це все більш і більш підвищує вимоги до якості тестів і максимально завантажує тестову інфраструктуру

Тестування є складовою частиною процесу налагодження ПЗ, після виявлення помилок дефекти в програмному коді повинні бути усунені розробниками.

Завданнями сучасного тестування є не тільки виявлення помилок в програмах, а й виявлення причин їх виникнення. Такий підхід дозволяє розробникам функціонувати максимально ефективно, швидко усуваючи виникаючі помилки.

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.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Тестування програмного забезпечення Режим доступу - http://www.protesting.ru/testing/ Дата доступу 25.05.2017

2. Життєвий цикл ПЗ Режим доступу - http://5fan.ru/wievjob.php?id=21297 Дата доступу – 25.05.2017

3. Цикли тестування Режим доступу http://www.4stud.info/software-construction-and-testing/lecture9.html Дата доступу 27.05.2017

4. Метрики і критерії оцінювання Режиму доступу http://www.4stud.info/software-construction-and-testing/lecture11.html Дата доступу 28.05.2017

5. Стратегії тестування Режим доступу - http://www.4stud.info/software-construction-and-testing/lecture10.html Дата доступу 29.05.2017

6. Тестування методом чорного ящика Режим доступу - http://lviv.qalight.com.ua/baza-znan/white-black-grey-box-testuvannya/

7. Тестування продуктивності Режим доступу – http://lviv.qalight.com.ua/baza-znan/testuvannya-produktivnosti/ Дата доступу 29.05.2017

8. Юзабіліті тестування Режим доступу – http://webstudio2u.net/ua/design-web/655-usabiliti-testirovanie.html Дата доступу - 30.05.2017

9. Тестування сумісності Режим доступу - http://lib.mdpu.org.ua/e-book/vstup/L11.htm Дата доступу - 30.05.2017

10. С. Орлик Якість програмного забезпечення 2004 – 2005

11. Животова А. А., Моденов Ю. Б. Методи та засоби тестування Web – додатків. 2014

12. Липаєв В. В. Основи тестування програм 2010

13. Круг С. Веб - дизайн або не змушуйте мене думати 2005

14. Тестування веб додатків Режим доступу http://is.unicyb.kiev.ua/files/student_reports/poliachenko_2015.pdf Дата доступу 31.05.2017

15. Шрімптон С. Розробка засобів тестування 2005

16. Selenium Режим доступу https://selenium2.ru/ Дата доступу 31.05.2017

17. Що таке Selenium Режим доступу - https://habrahabr.ru/post/152653/ Дата доступу – 01.05.2017

18. Selenium Документація Режим доступу https://selenium2.ru/docs.html Дата доступу - 01.06.2017

19. Баг трекінгові системи Режим доступу - https://xakep.ru/2014/10/08/bug-tracking-systems/ Дата доступу 01.06.2017

20. Mantis Режим доступу https://ru.wikipedia.org/wiki/Mantis_Програма Дата доступу - 02.06.2017

21. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно- обчислювальних машин

22. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин

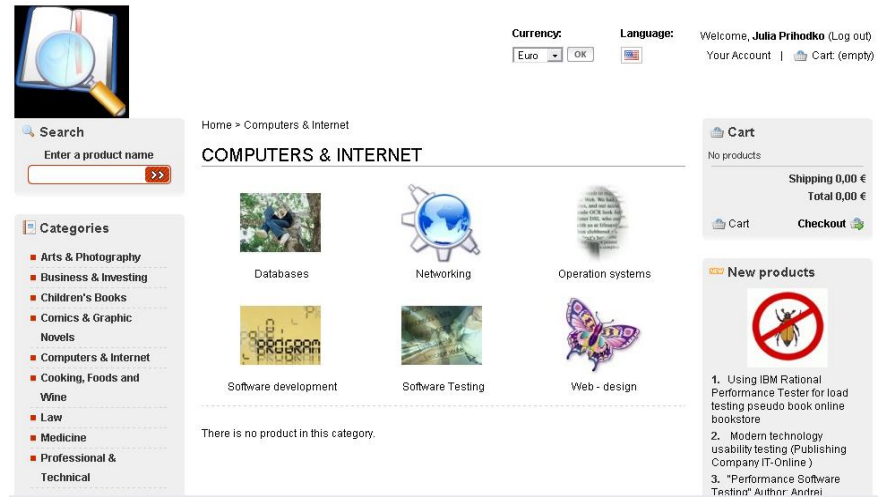
23. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень»

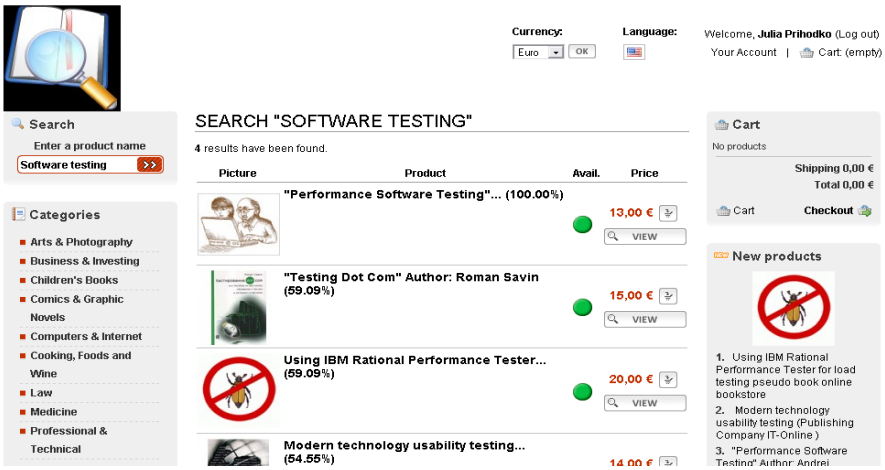
24. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку

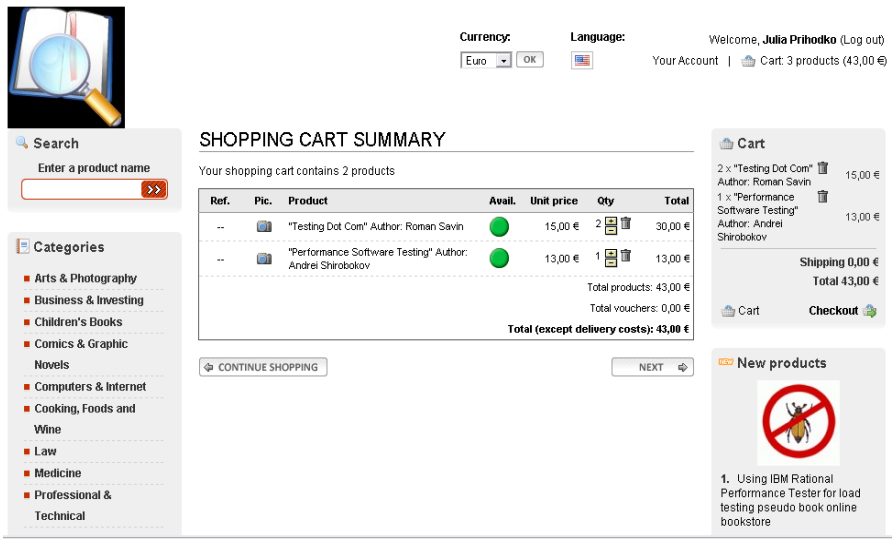
25. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку

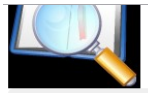
Додаток А

Сторінки веб-книгарні, обраної для тестування









Euro OK

Your Account | Cart: (empty)

Search

Enter a product name

Categories

- Arts & Photography
- Business & Investing
- Children's Books
- Comics & Graphic Novels
- Computers & Internet
- Cooking, Foods and Wine
- Law
- Medicine
- Professional & Technical

Home > Computers & Internet > Software Testing > **"Testing Dot Com" Author: Roman Savin**

"TESTING DOT COM" AUTHOR: ROMAN SAVIN

15,00 €

Quantity : 1 ADD TO CART

Availability : 9 in stock

More details about " "Testing Dot Com" Author: Roman Savin"

This course was created for those who want to learn testing, get a job tester in Russian or Western Internet companies, understand how to behave in a corporate environment, and to achieve professional and personal growth. It will be interesting to the participants and the process of developing software, recruiters, people connected to the Internet or a typewriter on it, and all those wishing to understand just cuisine Internet startups. The book is based entirely on the personal experiences of development - from scratch - a profession tester and author of many years of work in this capacity in the United States of Internet companies.

Cart

No products

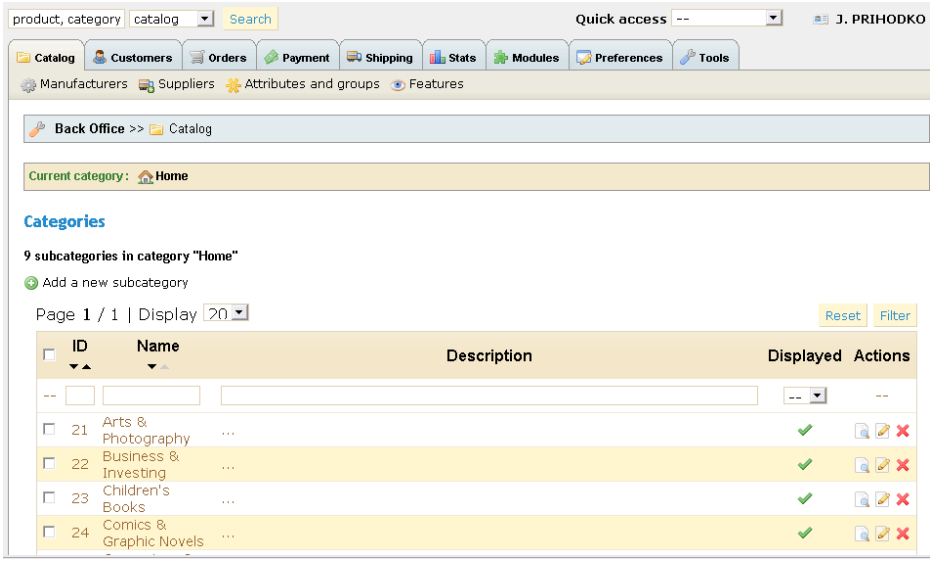
Shipping 0,00 €

Total 0,00 €

Cart Checkout

New products

1. Using IBM Rational Performance Tester for load testing pseudo book online bookstore
2. Modern technology usability testing (Publishing Company IT-Online)
3. "Performance Software Testing" Author: Andrei Shirobokov



INVOICE #000002

Delivery
Quadecco
Julia PRIHODKO
Gagarina str.
Shironintsev str.
61032 Kharkov
United Kingdom
80577773456
809727227060

Invoicing
Quadecco
Julia PRIHODKO
Gagarina str.
Shironintsev str.
61032 Kharkov
United Kingdom

INVOICE #000002 from 2008-05-30		
Order date: 2008-05-30	Carrier: My carrier	Payment method: Cheque

Description	Reference	U. price	Qty	Pre-Tax Total	Total
"Testing Dot Com" Author: Roman Savin		15,00 €	1	15,00 €	15,00 €

Total products TI :	15,00 €
Total discounts :	0,00 €
Total shipping :	7,00 €
Total with Tax :	22,00 €

Tax detail	Tax %	Pre-Tax Total	Total Tax	Total with Tax
Carrier	19,60	5,85 €	1,15 €	7,00 €

Додаток Б

Автоматизовані функціональні тест кейси для електронної книгарні

TestElementPresent		
open	../category.php?id_category=32	
verifyElementPresent	css=a.product_link	
verifyElementPresent	css=a.add_to_cart	
verifyText	class=product_link	"Testing Dot Com" Author: Roman Savin
clickAndWait	class=product_link.view	
verifyElementPresent	class=product_link	
verifyText	class=product_link	Modern technology usability testing
clickAndWait	class=product_link.view	
goBackAndWait		
verifyElementPresent	class=product_link	
verifyText	class=product_link	Performance Software Testing
clickAndWait	class=product_link.view	
goBackAndWait		
clickAndWait	css=div.cart_block	
verifyText	Shopping cart summary	
verifyElementPresent	css=tr.cart_entry	
verifyTextNotPresent	css=tr.cart_entry	"Testing Dot Com" Author: Roman Savin
verifyTextNotPresent	css=tr.cart_entry	Modern technology usability testing
verifyTextNotPresent	css=tr.cart_entry	Performance Software Testing

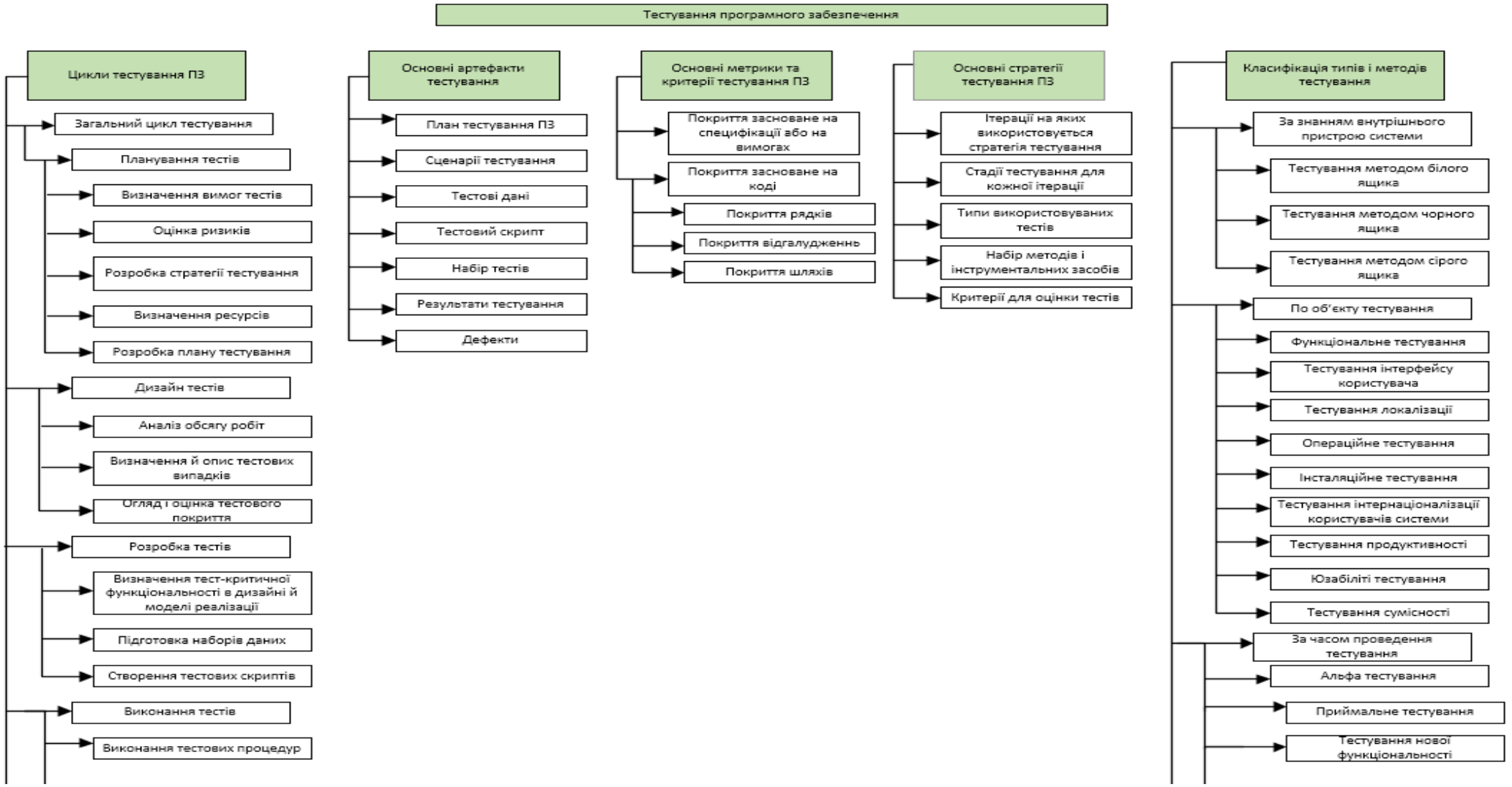
```
<body>
<table cellpadding="1" cellspacing="1" border="1">
<tbody>
<tr><td rowspan="1" colspan="3">TestElementPresent</td></tr>
<tr>
    <td>open</td>

    <td>../category.php?id_category=32</td>
    <td></td>
</tr>
<tr>
    <td>verifyElementPresent</td>
    <td>css=a.product_link</td>
    <td></td>
</tr>
<tr>
    <td>verifyElementPresent</td>
    <td>css=a.add_to_cart</td>
    <td></td>
</tr>
<tr>
    <td>verifyText</td>
    <td>class=product_link</td>
```

```
    <td>"Testing Dot Com" Author: Roman Savin</td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>class=product_link.view</td>
    <td></td>
</tr>
<tr>
    <td>verifyElementPresent</td>
    <td>class=product_link</td>
    <td></td>
</tr>
<tr>
    <td>verifyText</td>
    <td>class=product_link</td>
    <td>Modern technology usability testing</td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>class=product_link.view</td>
    <td></td>
</tr>
<tr>
    <td>goBackAndWait</td>
    <td></td>
    <td></td>

</tr>
<tr>
    <td>verifyElementPresent</td>
    <td>class=product_link</td>
    <td></td>
</tr>
<tr>
    <td>verifyText</td>
    <td>class=product_link</td>
    <td>Performance Software Testing</td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>class=product_link.view</td>
    <td></td>
</tr>
<tr>
    <td>goBackAndWait</td>
    <td></td>
    <td></td>

</tr>
<tr>
    <td>clickAndWait</td>
    <td>css=div.cart_block</td>
    <td></td>

</tr>
<tr>
    <td>verifyText</td>
    <td>Shopping cart summary</td>
    <td></td>
</tr>
<tr>
```

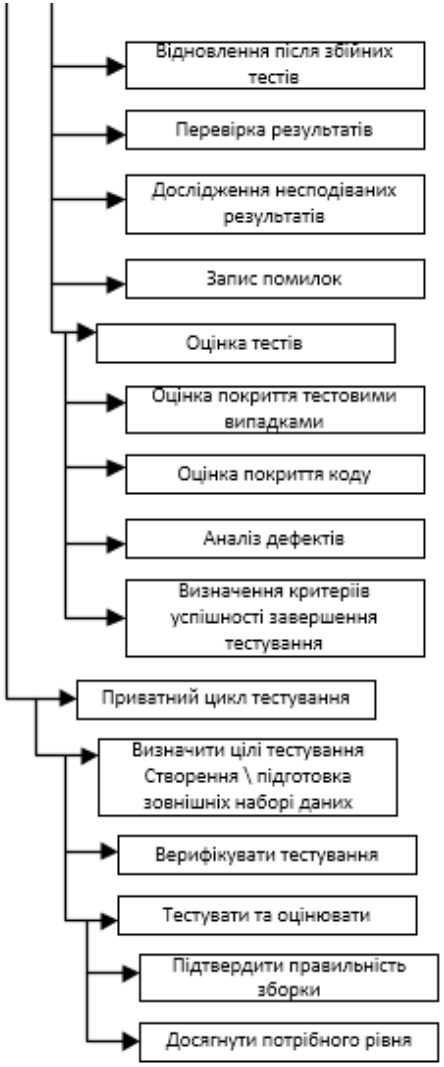
```
    <td>verifyElementPresent</td>
    <td>css=tr.cart_entry</td>
    <td></td>

</tr>
<tr>
    <td>verifyTextNotPresent</td>
    <td>css=tr.cart_entry</td>
    <td>"Testing Dot Com" Author: Roman Savin</td>
</tr>
<tr>
    <td>verifyTextNotPresent</td>
    <td>css=tr.cart_entry</td>
    <td>Modern technology usability testing</td>
</tr>
<tr>
    <td>verifyTextNotPresent</td>
    <td>css=tr.cart_entry</td>
    <td>Performance Software Testing</td>
</tr>
</tbody></table>
</body>
</html>
```

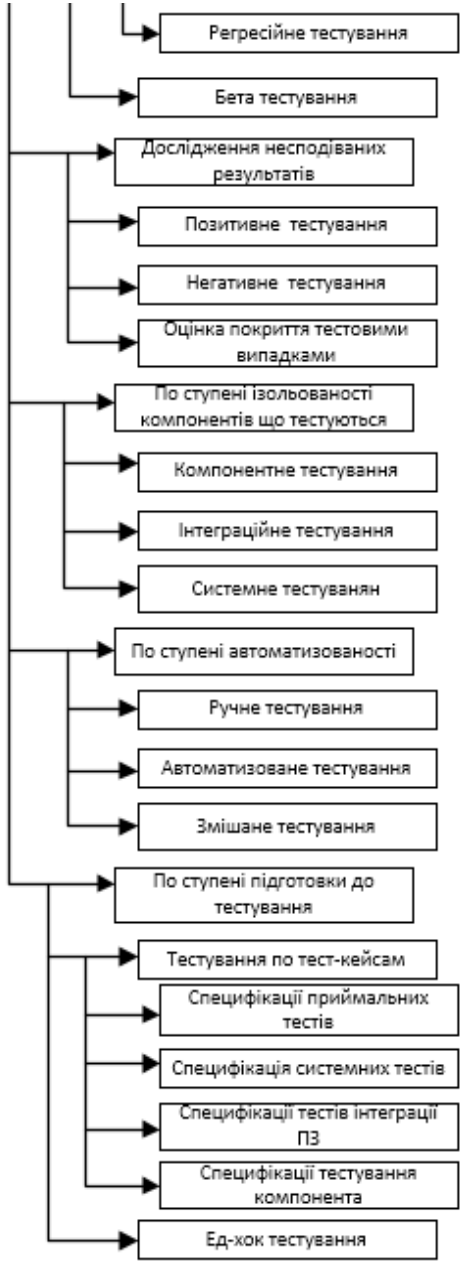
Додаток В

Схема тестування програмного забезпечення



Продовження на наступній сторінці


Відновлення після збійних тестів
Перевірка результатів
Дослідження несподіваних результатів
Запис помилок
Оцінка тестів
Оцінка покриття тестовими випадками
Оцінка покриття коду
Аналіз дефектів
Визначення критеріїв успішності завершення тестування
Приватний цикл тестування
Визначити цілі тестування Створення \ підготовка зовнішніх наборі даних
Верифікувати тестування
Тестувати та оцінювати
Підтвердити правильність зборки
Досягнути потрібного рівня


Регресійне тестування
Бета тестування
Дослідження несподіваних результатів
Позитивне тестування
Негативне тестування
Оцінка покриття тестовими випадками
По ступені ізольованості компонентів що тестуються
Компонентне тестування
Інтеграційне тестування
Системне тестуванян
По ступені автоматизованості
Ручне тестування
Автоматизоване тестування
Змішане тестування
По ступені підготовки до тестування
Тестування по тест-кейсам
Специфікації приймальних тестів
Специфікація системних тестів
Специфікації тестів інтеграції ПЗ
Специфікації тестування компонента
Ед-хок тестування