Міністерство освіти і науки України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Лабо (Черній) Віра Русланівна

Алгоритми мультикритеріального тестування веб-додатків / Multicriterion testing algorithms of Web-applications

Спеціальність 8.091501 – Комп'ютерні системи та мережі

Дипломна робота за освітньо-кваліфікаційним рівнем «магістр»

Науковий керівник
к.т.н., доцент Березька К.М.
_____
Нормоконтролер
д.т.н., доцент Березький О.М.
_____

Дипломну роботу допущено до захисту
«___»_____ 20 __ р.
Зав. кафедри КІ
Березький О.М. _____

Тернопіль – 2017

Міністерство освіти і науки України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

«Затверджую»
завідувач кафедри
д.т.н., проф. Березький О.М.
_____
"7" 11 2017 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТКИ

Лабо Віри Русланівни

1. Тема дипломної роботи "Алгоритми мультикритеріального тестування веб-додатків"
затверджена наказом №629 від "3" 11 2015 р.

2. Термін здачі закінченої дипломної роботи "15" січня 2017р.

3. Об'єкт дослідження: веб-додатки.

4. Предмет дослідження: методи тестування веб-додатків.

5. Перелік задач, які мають бути вирішені:
   - дослідити відомі методи тестування програмного забезпечення;
   - дослідити відомі інструменти тестування веб-додатків;
   - здійснити аналіз функціонування системи неперервної інтеграції;
   - розробити алгоритм роботи системи тестування веб-додатками;
   - програмно реалізувати розроблений алгоритм;
   - провести тестування реалізованої системи з використанням реальних даних.

6. Перелік ілюстративного матеріалу:
- тема;
- актуальність теми;
- мета і завдання;
- об'єкт і предмет дослідження;
- наукова новизна;
- практичне значення і публікація;
- тестування ПЗ;
- веб-додаток;
- неперервна інтеграція;
- структура побудови проекту;
- інструменти функціонального тестування;
- Unit тести. Code coverage;
- Тестування проекту;
- Selenium тести;
- Unit тести;
- API тести.

7. Консультанти по роботі

Розділ	Консультант	Підпис
Нормо-контроль	Мельник Г.М.	

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітк
1	Аналіз методологій створення веб-додатків та рівнів їх тестування	4.11.2015-1.01.2016	
2	Методи та засоби тестування веб-додатків	2.01.2016-31.05.2016	
3	Програмна реалізація системи мультикритеріального тестування веб-додатку	1.06.2016-23.01.2017	
4	Нормоконтроль, попередній захист	24.01.2017-31.01.2017	
5	Захист	1.02.2017	

Завдання прийняв до виконання _____ (підпис) Лабо В.Р. (прізвище та ініціали)

Керівник дипломної роботи _____ (підпис) Берцка К.М. (прізвище та ініціали)

РЕЗЮМЕ

Дипломна робота на тему «Система мультикритеріального тестування веб-додатків» на здобуття освітньо-кваліфікаційного рівня «Магістр» зі спеціальності «Комп'ютерні системи та мережі» написана обсягом 102 сторінок і містить 25 ілюстрацій, 3 таблиці, 3 додатки та 52 джерела з переліком посилань.

Метою роботи є підвищення якості програмного забезпечення завдяки створенню системи, що поєднує у собі кілька способів тестування веб-додатку та її програмна реалізація.

Методи досліджень. Використовуючи різні методи тестування програмного забезпечення здійснюється їх порівняння та вибір найоптимальніших, що доповнюють один одного. Здійснюється реалізація системи з використанням безперервної інтеграції, управління конфігурацією. Також задіяно методи об'єктно-орієнтованого програмування для проектування програмних засобів системи.

Результати дослідження: проведено аналіз існуючих методів тестування веб-додатків, здійснено програмну реалізацію, яка використовує технологію безперервної інтеграції для ефективної роботи з різними методами тестування програмної системи. Результати роботи можуть бути використані в будь-яких системах, для яких потрібно створити та автоматизувати процес тестування веб-додатків, а також в навчальному процесі.

Орієнтовні напрямки розвитку досліджень: автоматизація тестування програмного забезпечення у веб, методи тестування веб-додатків.

КЛЮЧОВІ СЛОВА: ТЕСТУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ВЕБ-ДОДАТКИ, БЕЗПЕРЕРВНА ІНТЕГРАЦІЯ.

RESUME

Diploma work: «The system of multi-criterion testing of web applications» for the educational qualification of "Master" specialty "Software of systems" written 102 page volume and contains 25 figures, 3 tables, 3 applications and 52 sources for references.

The aim is to improve the quality of software by creating a system that combines several methods of testing web applications and its software implementation.

Research methods. Using different methods of software testing is carried out to compare and choose the most optimal, which complement each other. Implement realization of system using continuous integration, configuration management system. Also involve methods of object-oriented programming to design software of system.

The results: The analysis of existing methods for testing web applications, done software implementation with using continuous integration for efficient operation of the various methods of testing software of system. The results can be used in any systems which require of making and automating the process of testing web applications, as well as in the classroom.

The estimated directions of research: automation of software testing in the web, methods of testing of web applications.

KEY WORDS: TESTING, SOFTWARE, WEB APPLICATIONS, CONTINUOUS INTEGRATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – програмне забезпечення

АЗ – апаратне забезпечення

ПК – персональний комп’ютер

БД – база даних

ОС – операційна система

ОП – оперативна пам’ять

СІ – сервер інтеграції

API – прикладний програмний інтерфейс

ТЗ – технічне завдання

GUI – графічний інтерфейс користувача

ВСТУП

Актуальність теми. У сьогоденні, з розвитком програмного забезпечення, постає питання його надійності. Високу якість функціонування програмного забезпечення можливо отримати завдяки тестуванню як процесу виявлення дефектів. Не надійне програмне забезпечення, що не пройшло процес тестування, може обернутися великими втратами для підприємства чи компанії. Тому і постає потреба розвитку методів тестування програмного забезпечення. Для того, щоб підвищити якість вихідного продукту варто скористатись не одним методом тестування. Саме тому актуальним є побудова цілої системи тестування програмного забезпечення.

Мета і завдання дослідження. **Метою даної** роботи є підвищення якості програмного забезпечення завдяки створенню системи, що поєднує у собі кілька способів тестування програмного додатку. Дана система повинна показувати більш високі показники надійності та достовірності у порівнянні з існуючими рішеннями.

Об’єкт дослідження: веб-додатки.

Предмет дослідження: методи тестування веб-додатків.

Методи дослідження: використовуючи різні методи тестування програмного забезпечення здійснюється їх порівняння та вибір найоптимальніших, що доповнюють один одного. Здійснюється реалізація системи з використанням безперервної інтеграції, управління конфігурацією. Також задіяно методи об’єктно-орієнтованого програмування для проектування програмних засобів системи.

Наукова новизна одержаних результатів.

1. Запропоновано алгоритм поєднання кількох методів тестування веб-додатків.
2. Вперше одержано систему тестування веб-додатків багатьма методами одночасно.

Практичне значення отриманих результатів. Побудовано алгоритм виконання кількох методів тестування одного веб-додатку. Здійснено програмну реалізацію системи багатокритеріального тестування веб-додатків, побудовано основні алгоритми та проведено тестування системи.

Публікації та апробація ДР. Результати дипломної роботи були заслухані на VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп'ютерні інформаційні технології 2016»[52].

Впровадження результатів ДР. Дипломна робота на тему: «Система мультикритеріального тестування веб-додатків» відповідає замовленню підприємства, має певну практичну значимість і планується до використання у електронних ресурсах ТОВ «Медіспорт».

1 АНАЛІЗ МЕТОДОЛОГІЙ СТВОРЕННЯ ВЕБ-ДОДАТКІВ ТА РІВНІВ ЇХ ТЕСТУВАННЯ

1.1 Види веб-додатків і технології їх створення

1.1.1 Поняття веб-додатку та його класифікація

Веб-додаток – це сукупність статичних і динамічних веб-сторінок, тобто наперед створених сторінок і програм, що створюють веб-сторінки у відповідь на звернення користувача. Можна вважати що сайт і веб-додаток – тотожні поняття[1].

Зазвичай при створенні веб-додатків використовують архітектуру клієнт-сервер. Користувач дає запит веб-додатку, веб-додаток його обробляє та відправляє її через мережу до веб-сервера. Сервер відповідає на запит формуючи веб-сторінку і відправляє її назад веб-додатку також мережею. Веб-додаток в свою чергу вже відображає надану сервером інформацію користувачеві. Зазвичай веб-додаток використовує браузер, як інтерфейс користувача.

Веб-додатки можна розподілити на види:

- рейтинги;
- форми відправки повідомлень;
- форми реєстрації;
- гостьові книги;
- форуми;
- форми завантаження (upload);
- системи голосування;
- пошукові системи (по сайту чи по інтернет);
- движки сайту;
- content managment system (CMS);
- банерні движки та системи (локальні та глобальні);
- веб-пошта;

– персоналізовані веб-системи різного спрямування, які надають комплекс послуг свої користувачам[2].

До основних вразливостей вищенаведеної класифікації веб-додатків відносяться:

– повна відсутність перевірки вхідних даних (у веб-формах будь-яких систем) або тільки часткова перевірка даних;

– некоректна обробка вхідних даних (нульовий байт, символи рівня директорій);

– переповнення буферу;

– необережна робота програми з файлами, у випадку коли ім’я файлу передається програмі ззовні (GET або POST);

– неврахування особливостей GET та POST запитів;

– некоректна робота з паролями (під час зберігання, передачі та обробки даних);

– неправильні права доступу;

– неправильні права програм на сервері;

– неврахування особливостей роботи програм завантаження файлів на сервер;

– некоректна логіка роботи веб-програми, яка при деяких допустимих вхідних даних приводить до непередбачуваних наслідків;

– виведення інформації при помилках програми або доступу до бази даних, коли виводиться додаткова службова інформація, не призначена для сторонніх очей;

– некоректна робота з базами даних (паролі, виведення службової інформації, завищена кількість запитів до БД);

– вразливості недостатньої обробки вхідних даних при роботі з базою даних (SQL-ін’єкції);

– неоптимізований програмний код, котрий приводить до значних навантажень на веб-сервер (при своїй звичайній роботі та особливо у випадку збою при передачі некоректних вхідних даних);

– вразливість веб-додатків та систем до DoS та DDoS атак[2].

Класифікація видів веб-додатків за типом його призначення: CRM, ERP, ITRP, OSS, CGI. Проаналізуємо їх докладніше.

CRM (Customer Relationship Management) – веб-додатки для автоматизації та підвищення ефективності процесів, пов'язаних з бізнесом (обробка замовлень, маркетинг, обслуговування клієнтів). CRM використовуються в спеціалізованих операторських «контакт-центрах». Веб-сервіси Microsoft CRM реалізуються на основі використання SQL-сервера і передбачають створення основного сховища даних Microsoft CRM, БД метаданих, БД для побудови звітності та дистрибуційної БД, призначеної для відстеження взаємодії автономних користувачів клієнта Outlook з основною БД Microsoft CRM. Використання XML дозволяє інтегрувати Microsoft CRM з додатками подібного призначення незалежно від мови програмування і операційної системи, під управлінням якої працює стороннє додаток (наприклад, SAP R / 3). Система передбачає обмеження доступу і перевірку прав доступу клієнтів.

ERP (Enterprise Resource Planning) – веб-додатки, призначені для автоматизації процесів управління внутрішньогосподарської діяльністю корпорації, включаючи управління виробництвом, фінансами, постачанням, персоналом.

ITRP (IT Resources Planning) – клас веб-додатків, призначений для підтримки управління корпоративними IT-ресурсами та сервісами.

OSS (Operation Support Systems) – вид веб-додатків, призначений для забезпечення роботи операторів розподілених обчислювальних мереж. OSS забезпечує управління мережею, продуктивність, ліквідацію збоїв в роботі, створення і облік сервісів, планування мережевих ресурсів, моніторинг процесів, контроль за безпекою, якість послуг і рівень обслуговування клієнтів, збором статистичних даних. Різновидом OSS є система підтримки бізнесу – BSS (Business Support Systems). До них відносяться біллінгові системи, системи управління взаємовідносинами з клієнтами, управління мережами, замовленнями, якістю послуг.

CGI (Common Gateway Interface, загальний шлюзовий інтерфейс) - програми пошуку в віддалених БД, переадресації посилань, використання

графічних меню, зв'язку з базами даних (шляхом запуску програми перетворення форматів баз даних в формат мови HTML)[3].

1.1.2 Технології створення веб-додатків

Засоби створення веб-додатків: ISAPI, CGI, ASP, JSP, WAP. За час існування WWW зміст веб-додатків, функції які вони виконують, принципи і архітектура їх побудови зазнали значних змін. З найпростіших засобів збереження HTML сторінок вони «виросли» до готових рішень, що орієнтовані на підтримку роботи корпоративними інформаційними системами[3].

AJAX (Asynchronous JavaScript and XML). Підхід побудови інтерфейсу користувача веб-додатків, при якому відповідь на кожен запит користувача не змушує перезавантажувати сторінку браузера, а тільки оновлює чи додає нові дані, які йому необхідні[3]. Тобто запити на сервер формуються і надсилаються у фоновому режимі ніяк не відображаючи це користувачеві, а відповідь вставляється в готову веб-сторінку, при цьому оновляючи лише деяку її частину. Такий підхід дозволяє створювати більш зручні веб-інтерфейси для користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. Асинхронність в даному випадку дозволяє користувачеві далі переглядати контент веб-сторінки в той час, поки обробляється запит на сервері, що є дуже зручним. Класична модель роботи AJAX: користувач відкриває веб-сторінку, натискає на який-небудь її елемент, браузер відправляє відповідний запит на сервер, у відповідь сервер формує лише ту частину веб-сторінки, яка змінилась. Як результат – змінена певна частина веб-сторінки без її перезавантаження[4]. AJAX – це концепція використання суміжних певних технологій:

– Головною складовою підходу є JavaScript – динамічна, об'єктно-орієнтована мова програмування, яка і дозволяє динамічне підвантаження коду з використанням DOM, що здійснюється із використанням формату JSON[5].

– DHTML (Dynamic HTML) використовується для динамічної зміни змісту сторінки. Ця технологія створення веб-сайту розглядає HTML-документ як об'єктну структуру, використовує поєднання статичної мови розмітки HTML, вбудованої скриптової мови JavaScript (сценарії виконуються на стороні

клієнта), CSS (каскадних таблиць стилів) і DOM (об'єктної моделі документа). Також може бути використана для навігації або для додання інтерактивності формам веб-додатку[6].

– Також важливим є використання XMLHttpRequest – браузер формує API-запит для звернення до сервера у фоновому режимі за протоколом HTTP, що дозволяє не перезавантажувати сторінку повністю. Запит цього формату можна використовувати як для синхронного, так і асинхронного обміну інформацією в довільному текстовому форматі (наприклад XML, JSON, HTML). Застосування XMLHttpRequest справляє враження «миттєвої» відповіді сервера, у порівнянні з класичними методом перезавантаження всієї сторінки для оновлення представленої на ній інформації[7].

– Динамічне створення дочірніх фреймів[4].

ASP (Active Server Pages) — технологія, яка дозволяє створювати динамічні веб-сторінки HTML, використовуючи об’єктну модель інтерфейсу, створеного на основі ISAPI (Internet Server Application Programming Interface) розширень і фільтрів. Принцип, що закладений в основу інтерфейсу полягає в тому, що на веб-сторінці знаходяться фрагменти коду, який інтерпретується веб-сервером, на якому встановлене розширення ISAPI, і він надає користувачу вже готовий результат виконання вибраних фрагментів коду. Спеціальний «динамічний» тег <%...%> вказує, що в ньому знаходиться потрібний код. Мови програмування можна використовувати практично будь-які. По замовчуванню підтримуються JavaScript і VBScript.

ISAPI (Internet Server Application Programming Interface) – інтерфейс до сервера Інтернету фірми Microsoft, призначений для програмного керування сервером. ISAPI підтримується більшістю виробників програмних засобів. ISAPI-програми являють собою спеціальний вид додатків, що обробляють запити користувачів і відображають їх вивід у вигляді потоку HTML, який надходить безпосередньо в браузер клієнта.

JSP (Java Server Pages) — технологія створення веб-додатків, що базується на однократній компіляції Java-коду (сервлету) при першому зверненні до нього з подальшим виконанням методів цього сервлету і

розміщенням отриманих результатів в набір даних, які відправляються в браузер[3].

1.2 Аналіз розробки програмного забезпечення

1.2.1 Структура веб-додатку

Найчастіше для створення програмного забезпечення беруться мінімум 3 компоненти за основу:

1. Клієнтська частина – графічний інтерфейс. Користувач взаємодіє з веб-додатком через браузер, який і відображає графічний інтерфейс. Дана частина може використовувати наступні технології: JavaScript, Flash, Java / JavaFX, ActiveX, Silverlight.

2. Серверна частина – програма або скрипт на сервері, який обробляє запити користувача (через браузер). Вона відповідає за централізовану обробку і збереження даних, підтримку веб-інтерфейсу користувача, інформування користувачів у вигляді e-mail повідомлень про події в системі, взаємодіє із обліковою системою клієнта. Найчастіше серверна частина веб-додатку програмується через PHP. При кожному переході користувача по посиланню браузер відправляє запит до сервера. Сервер обробляє цей запит, викликаючи при цьому певний PHP-скрипт, який і формує веб-сторінку, описану мовою розмітки HTML і відправляє її користувачу через мережу. В кінці браузер відображає отриманий результат у вигляді чергової веб-сторінки.

3. База даних – програмне забезпечення на сервері, яке відповідає за збереження даних і їх видачею в потрібний момент часу. У разі форуму або блогу, збережені в БД дані – це пости, коментарі, новини, і так далі. Сама база даних розташовується безпосередньо на сервері. Сервер відповідає за зв'язок із БД. Серверна частина веб-додатку, тобто PHP-скрипт, звертається до БД, витягуючи дані, які необхідні для формування сторінки, запитаної користувачем[8].

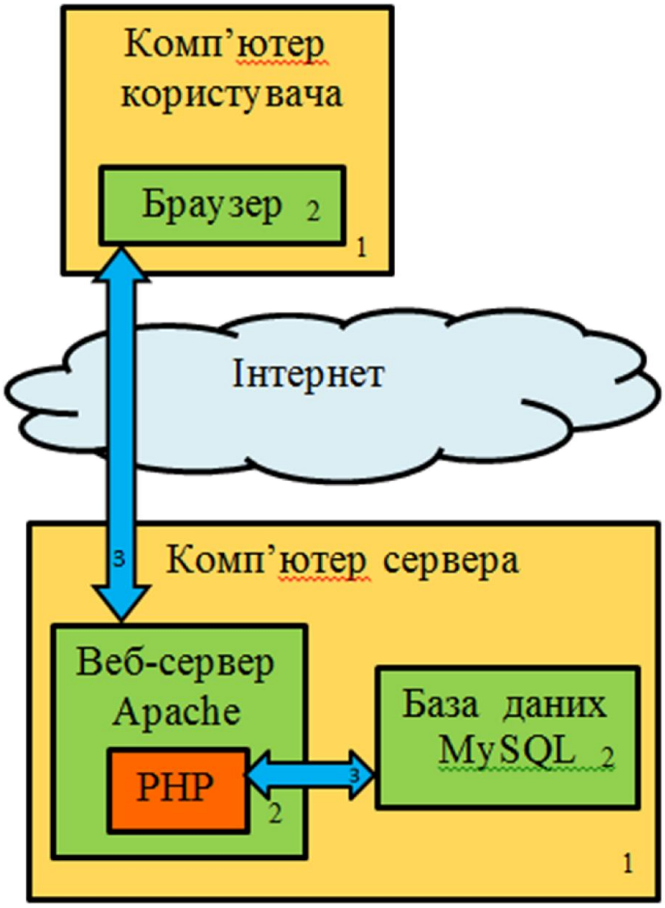

Рисунок 1.1 – Графічна схема взаємодії між клієнтською, серверною частинами і базою даних: 1 – комп'ютери, 2 – програми, 3 – зв'язок між програмами.

На рисунку 1.1 зображена схема взаємодії складових веб-додатку. Браузер через Інтернет відправляє HTTP-запити веб-серверу. Веб-сервер викликає PHP-скрипт, написаний розробником веб-додатку. PHP-скрипт звертається до бази даних, якщо потрібно. В результаті PHP-скрипт повертає клієнту веб-сторінку, яку і відображає браузер[8].

1.2.2 Класифікація методологій розробки ПЗ

Відомі наступні методології розробки програмного забезпечення:

- каскадна модель розробки (waterfall) (рисунок 1.2);
- гнучка модель розробки (agile) (рисунок 1.3);
- екстремальне програмування (XP) (рисунок 1.4).

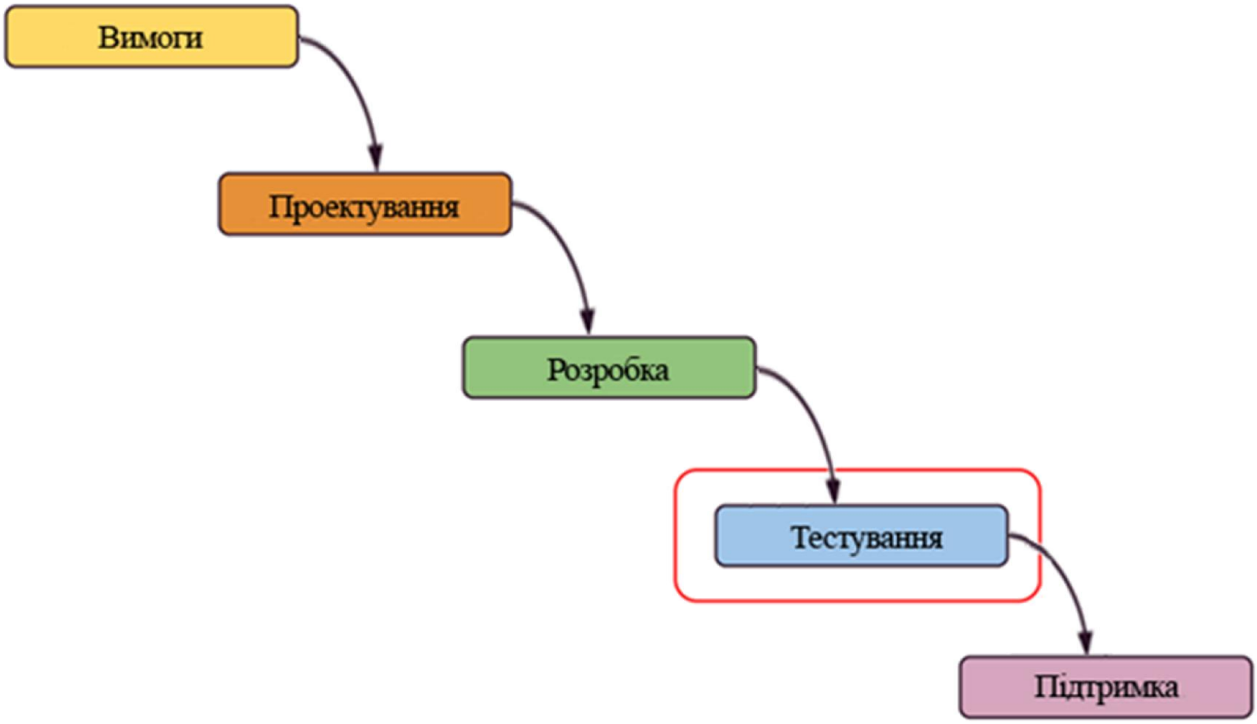

Рисунок 1.2 – Схема каскадної моделі розробки програмного забезпечення

Як видно з рисунку 1.2 етап тестування є четвертим. Перевагами даного методу є закінчений набір проектної документації після кожної стадії та чітка послідовність етапів. Недоліками є слабкий зв'язок з реальністю, жорсткі обмеження на формулювання вимог до проекту, результати доступні замовнику лише в кінці проекту.

Рисунок 1.3 – Схема гнучкої моделі розробки програмного забезпечення

У гнучкій моделі розробки ПЗ тестування не є одним і єдиним етапом, а повторяється в залежності від ітерацій і розробок (рисунок 1.3). Переваги: вимоги не встигають старіти (ставати не актуальними), фіксована довжина ітерацій, команда самостійно оцінює задачі, команда самокерована, зворотній зв'язок, команда крос-функціональна. Але на жаль великим недоліком є ефективність. Оправдується ця модель лише для інфраструктурно-складних проектів, проектів з тривалим циклом утвердження ТЗ, проектів без зворотного зв’язку, недостатньо кваліфікованої команди.

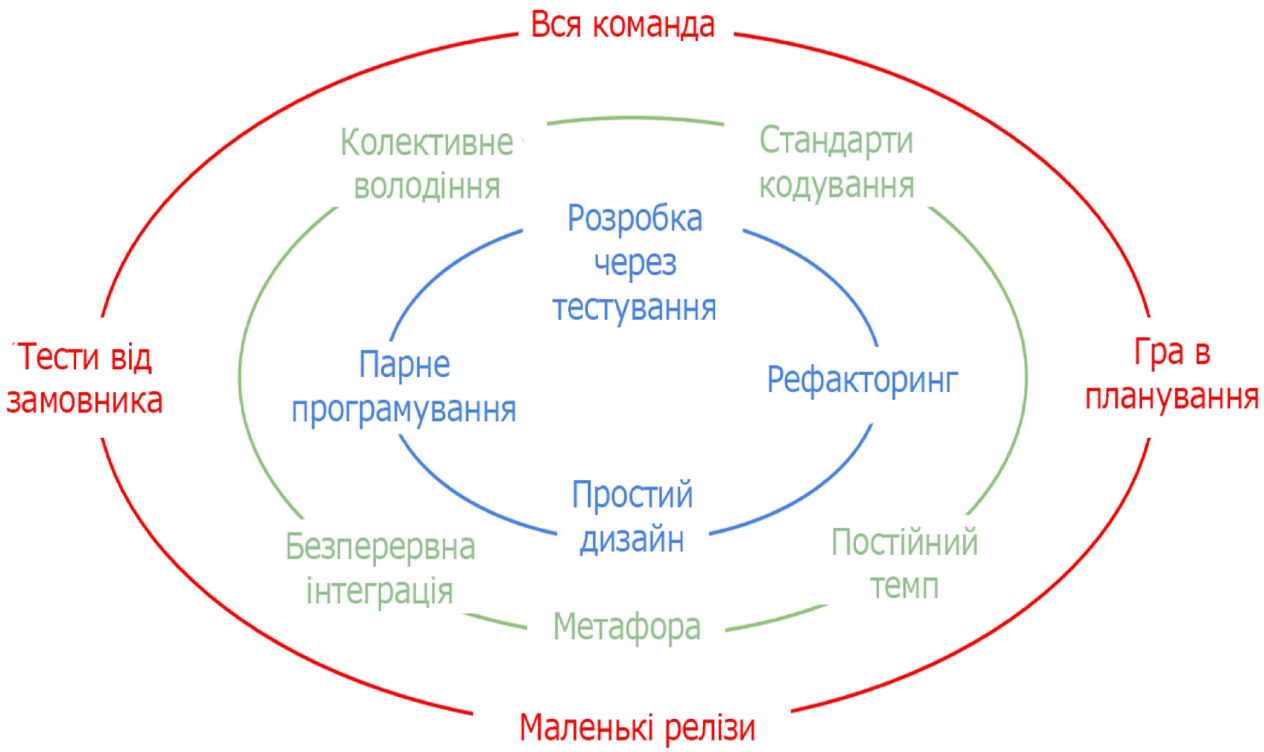

Рисунок 1.4 – Схема екстремального програмування

Екстремальне програмування – це спрощена методологія організація розробки ПЗ для невеликих і середніх по розміру команд розробників, які займаються створенням продукту в умовах неясних або, які швидко змінюються (рисунок 1.4).

Основними цілями є підвищення довіри замовника до програмного продукту шляхом надання реальних доказів успішності розвитку процесу розробки і різке скорочення термінів розробки продукту. При цьому XP зосереджено на мінімізації помилок на ранніх стадіях розробки, що скорочує час випуску продукту.

Загальні принципи XP:

– Інтерактивність. Розробка ведеться короткими ітераціями при наявності активного взаємозв'язку з замовником. Ітерації пропонується робити короткими, рекомендована тривалість – 2-3 тижні і не більше 1 місяця. За одну ітерацію група програмістів зобов'язана реалізувати кілька властивостей системи, кожне з яких описується в користувацькій історії (КІ). КІ в даному випадку є початковою інформацією, на підставі якої створюється модуль. Вони відрізняються від варіантів використання (ВВ). Опис КІ короткий – 1-2 абзаци,

тоді як ВВ зазвичай описуються досить докладно, з основним і альтернативними потоками, і доповнюються моделлю. КІ пишуться самими користувачами, які в ХР є частиною команди, на відміну від ВВ, які описує системний аналітик. Відсутність формалізації опису вхідних даних проекту в ХР прагнуть компенсувати за рахунок активного включення в процес розробки замовника як повноправного члена команди.

– Простота рішень. Приймається перше просте робоче рішення. Екстремальність методу пов'язана з високим ступенем ризику рішення, обумовленого поверховістю аналізу і жорстким тимчасовим графіком. Реалізується мінімальний набір основних функцій системи на першій і кожній наступній ітерації; функціональність розширюється на кожній ітерації.

– Інтенсивна розробка малими групами (не більше ніж 10 осіб) і парне програмування (коли два програміста разом створюють код на одному загальному робочому місці), активне спілкування в групі і між групами. Все це націлене на якомога більш раннє виявлення проблем (як помилок, так і зриву термінів). Парне програмування спрямоване на вирішення завдання стабілізації проекту. При застосуванні ХР методології високий ризик втрати коду з огляду на вихід програміста, який не витримав інтенсивного графіку роботи. У цьому випадку другий програміст з пари грає роль «спадкоємця» коду. Важливо й те, як саме розподілені групи в робочому просторі – в ХР використовується відкритий робочий простір, яке передбачає швидкий і вільний доступ всіх до всіх.

– зворотній зв'язок із замовником, представник якого фактично залучений в процес розробки;

– достатній рівень сміливості і бажання іти на ризики.

1.3 Рівні і види тестування програмного забезпечення у веб

1.3.1 Поняття тестування програмного забезпечення

Тестування – це процес перевірки відповідності між реальною і очікуваною поведінкою програми, здійснювана на кінцевому наборі тестів, вибраному певним чином[9].

Оскільки якість ніколи не може бути абсолютною, а лише суб'єктивним поняттям, тому тестування для вчасного виявлення помилок і дефектів не забезпечить повністю коректність протестованого забезпечення. Воно лише порівнює результати виконання програми із вимогами, які визначені в специфікаціях. Специфікація – це повний перелік вимог до поведінки системи, яку розроблятимуть. Специфікація складається із користувацьких сценаріїв (use cases), які описують варіанти взаємодії користувача і програмного забезпечення.

Отже, вирізнимо поняття тестування програмного продукту і забезпечення якості програмного продукту, до якого належить не лише тестування, а усі складові ділового процесу. Тобто тестування – це одна із технік контролю якості, що включає в себе дії по плануванню робіт (Test Management), проектуванню тестів (Test Design), виконання тестування (Test Execution) і аналізу отриманих результатів (Test Analysis). Після виконання тестування виявляється інформація про якість забезпечення.

Якість програмного забезпечення – це сукупність характеристик програмного забезпечення, які відносяться до його здатності задовольнити встановлені та запропоновані потреби[10].

Верифікація – це оцінка програмного забезпечення для визначення чи задовольняють результати даного етапу розробки умовам, які були сформульовані спочатку даного етапу. Тобто це перевірка чи в процесі виконання певного етапу розробки програмного забезпечення виконуються поставлені на його початку цілі, задачі і терміни[9].

Валідація – це визначення чи відповідає розроблюване програмне забезпечення потребам та очікуванням користувача, вимогам системи[9].

«Баг» - це відхилення фактичного результату (actual result) від очікуваного (expected result)[11].

Відмова (failure) – це відхилення програми від функціонування і неможливість програми виконувати функції, визначені вимогами і обмеженнями.

На рисунку 1.5 зображена схема ідеального тестування.

Рисунок 1.5 – Піраміда ідеального тестування

Етапи тестування:

1. Аналіз.
2. Розробка стратегії тестування і планування процедур контролю якості.
3. Робота з вимогами.
4. Створення тестової документації.

5. Тестування прототипу.

6. Основне тестування.

7. Стабілізація.

8. Експлуатація.

План тестування (Test Plan) – це документація, яка описує повний обсяг усіх робіт по тестуванню: опис об’єкту, стратегії, терміни, критерії початку і закінчення тестування, обладнання, оцінки ризиків, варіанти вирішення проблем і т.д. Тест-план відповідає на питання: Що потрібно тестувати? Що тестуватиметься? Як тестуватиметься? Коли тестуватиметься? Які критерії початку тестування? Як критерії закінчення тестування?[9].

В стандарті IEEE 829 перелічені пункти, з яких має (або може) складатись тест-план:

- ідентифікатор плану випробувань;
- введення;
- тестові завдання;
- характеристики для тестування;
- ознаки для не тестування;
- підхід;
- пункт критерій проходження/провалу;
- критерії тимчасової зупинки і вимоги відновлення;
- тест очікуваних результатів;
- завдання тестування;
- потреби навколишнього середовища;
- відповідальності;
- потреби у навчанні;
- розклад;
- ризики і непередбачені обставини;
- допуски[9].

Тест дизайн (Test Design) – це етап тестування програмного забезпечення, в процесі якого формуються і створюються тестові випадки (test cases), у відповідності до сформульованих цілей тестування і критерій якості.

1.3.2 Техніки тест дизайну

Відомі наступні техніки тест дизайну:

1. Еквівалентний поділ (Equivalence Partitioning - EP).
2. Аналіз граничних значень (Boundary Value Analysis - BVA).
3. Причина / наслідок (Cause/Effect - CE).
4. Передбачення помилки (Error Guessing - EG).
5. Вичерпне тестування (Exhaustive Testing - ET).

Еквівалентний поділ. Тестові дані розбиваються на певні класи допустимих значень. В межах кожного класу виконання тесту з будь-яким значенням тестових даних приводить до еквівалентного результату. Еквівалентні тести тестують одну й ту ж саму річ (частину системи, модуль, функцію). Якщо один з тестів знайде помилку, то скоріш за все інший також її знайде, і навпаки. Алгоритм еквівалентного поділу:

- виділити класи еквівалентності;
- вибрати одного представника кожного класу;
- виконати тести.

Наприклад, необхідно перевірити як працює поле, в яке можна ввести ціле число від 1 до 99. Визначаємо класи еквівалентності:

1) Будь-яке ціле число в діапазоні від 1 до 99. Зазвичай це буде середина діапазону (позитивний тест).
2) Будь-яке число менше 1 (негативний тест).
3) Будь-яке число більше 99 (негативний тест).
4) Дріб і «не число», тобто букви, спецсимволи (негативний тест).
5) Пустий ввід (негативний тест).

Тести, які необхідно виконати для визначених класів еквівалентності:

1) Ввести 1, 99, 50.
2) Ввести 0.
3) Ввести 100.
4) Ввести 50.5, букву, спецсимвол: ~`!”@’#$;%:^&?*()[]{},.\/+=-_.
5) Ввести пусту стрічку.

Аналіз граничних значень. На відміну від еквівалентного поділу, що орієнтований на покриття тестами, дана техніка орієнтована на ризики – програма може «зламатись» в області граничних значень. Алгоритм:

– виділити класи еквівалентності;

– визначити граничні значення цих класів;

– зрозуміти, до якого класу буде відноситись кожна границя;

– провести тести по перевірці значень до границі, на границі, і одразу ж за границею.

Приклад. Перевірити розмір комісії при відміні броні за певних час до вильоту (рисунок 1.6)

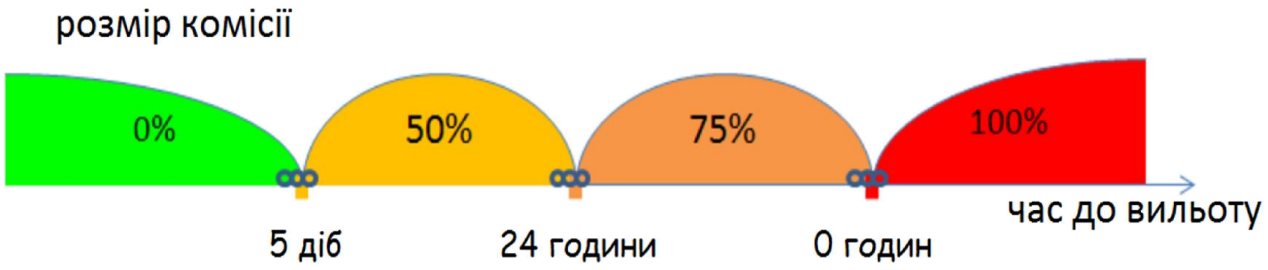

Рисунок 1.6 – Умови завдання

Виділимо класи еквівалентності:

– Час до вильоту > 5 діб.

– 24 години =< час до вильоту =< 5 діб.

– 0 годин < час до вильоту < 24 години.

– Час до вильоту =< 0 годин (виліт відбувся).

Визначимо границі відносно класів:

1) 5 діб;

2) 24 години;

3) 0 годин.

Визначимо, до якого класу відносяться границі:

1) 5 діб – до 2-го класу;

2) 24 години – до 2-го класу;

3) 0 годин – до 4-го класу.

Тести значень на границях, до і за ними:

– скасувати бронь за 5 діб + 1 секунду до вильоту (або просто виконати бронь якомога ближче до кордону, але зліва від неї) і перевірити, що комісія дорівнює 0%;

– скасувати бронь рівно за 5 діб до вильоту і перевірити, що комісія дорівнює 50%;

– скасувати бронь за 5 діб - 1 секунду до вильоту і перевірити, що комісія дорівнює 50%;

– скасувати бронь за 24 години + 1 секунду до вильоту і перевірити, що комісія дорівнює 50%;

– скасувати бронь рівно за 24 години до вильоту і перевірити, що комісія дорівнює 50%;

– скасувати бронь за 24 години - 1 секунду до вильоту і перевірити, що комісія дорівнює 75%;

– скасувати бронь за 1 секунду до вильоту і перевірити, що комісія дорівнює 75%;

– скасувати бронь рівно під час вильоту і перевірити, що комісія дорівнює 100%;

– скасувати бронь через 1 секунду після вильоту і перевірити, що комісія дорівнює 100%;

Причина / наслідок. Ввід комбінацій умов (причин), для отримання відповіді від системи (наслідок). Наприклад, треба перевірити можливість додавати клієнта, використовуючи певну екранну форму. Для цього необхідно ввести кілька полів, таких як «Ім'я», «Адреса», «Номер Телефону» а потім, натиснути кнопку «Додати» – це «Причина». Після натискання кнопки «Додати», система додає клієнта в базу даних і показує його номер на екрані – це «Наслідок»[12].

Передбачення помилки. Тест аналітик використовує свої знання системи та здатність до інтерпретації специфікації на предмет того, щоб «передбачити» за яких вхідних умовах система може видати помилку. Наприклад, специфікація каже: «Користувач повинен ввести код». Тест аналітик, буде

думати: «Що, якщо я не введу код?»; «Що, якщо я введу неправильний код? "; і так далі. Це і є передбачення помилки[12].

Вичерпне тестування – це крайній випадок. В межах цієї техніки тестувальник повинен перевірити всі можливі комбінації вхідних значень, і в принципі, це повинно знайти всі проблеми. На практиці застосування цього методу не представляється можливим, через величезну кількість вхідних значень[12].

Матриця відповідності вимог (Traceability matrix) – це двовимірна таблиця, яка містить відповідність функціональних вимог (functional requirements) продукту і підготовлених тестових сценаріїв (test cases). У заголовках колонок таблиці розташовані вимоги, а в заголовках рядків – тестові сценарії. На перетині – відмітка, що означає, що вимога поточної колонки покрита тестовим сценарієм поточного рядка. Матриця відповідності вимог використовується QA-інженерами для валідації покриття продукту тестами. Це є невід'ємною частиною тест-плану[12].

Тестова ситуація (test case) – це сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або певної її частини.

Тестові дані (test data) – це дані, які існують на початку виконання тесту і впливають на його роботу, або ж зазнають впливу зі сторони тестованої системи або компоненту. Тестові дані можуть бути створені тестувальником, модифіковані реальні, повністю реальні дані.

Звіт по тестуванню (Test report) – документ, основною ціллю якого є відображення результатів виконання плану тестування.

Тестове покриття (Test coverage) – одна із метрик оцінки якості тестування, що являє собою повністю покриття тестами вимог або виконуваного коду.

Баг/дефект репорт (Bug report) – це документ, який описує ситуацію або послідовність дій, що призводить до певної роботи об’єкту тестування, із зазначенням причин і очікуваного результату. Основними цілями і задачами його є:

– надати інформацію про проблему, її властивості і наслідки, які за собою несе проблема;

– визначити і позначити пріоритет проблеми за важливістю і швидкістю усунення;

– допомогти програмістам виявити і якомога швидше та якісніше усунути джерело проблеми.

1.3.3 Класифікація за рівнем тестування програмного забезпечення

Модульне тестування (Unit Testing). Компонентне (модульне) тестування перевіряє функціональність і шукає дефекти в частинах додатка, які доступні і можуть бути протестовані по-окремо (модулі програм, об'єкти, класи, функції і т.д.).

Інтеграційне тестування (Integration Testing). Перевіряється взаємодія між компонентами системи після проведення компонентного тестування.

Системне тестування (System Testing). Основним завданням системного тестування є перевірка як функціональних, так і не функціональних вимог в системі в цілому. При цьому виявляються дефекти, такі як неправильне використання ресурсів системи, непередбачені комбінації даних користувача рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або неправильна функціональність, незручність використання і т.д.

Операційне тестування (Release Testing). Навіть, якщо система задовольняє всім вимогам, важливо переконатися в тому, що вона задовольняє потребам користувача і виконує свою роль в середовищі своєї експлуатації, як це було визначено в бізнес-моделі системи. Слід врахувати, що і бізнес модель може містити помилки. Тому так важливо провести операційне тестування як фінальний крок валідації. Крім цього, тестування в середовищі експлуатації дозволяє виявити і нефункціональні проблеми, такі як: конфлікт з іншими системами, суміжними в області бізнесу або в програмних і електронних середовищах; недостатня продуктивність системи в середовищі експлуатації та ін. Очевидно, що знаходження подібних речей на стадії впровадження – критична і дорога проблема. Тому так важливо проведення не тільки верифікації, а й валідації, з самих ранніх етапів розробки ПЗ.

Приймальне тестування (Acceptance Testing). Формальний процес тестування, який перевіряє відповідність системи вимогам і проводиться з метою:

– визначення чи задовольняє система приймальним критеріям;

– винесення рішення замовником або іншою уповноваженою особою приймається додаток чи ні.

Класифікація видів тестування представлена на рисунку 1.7[13].

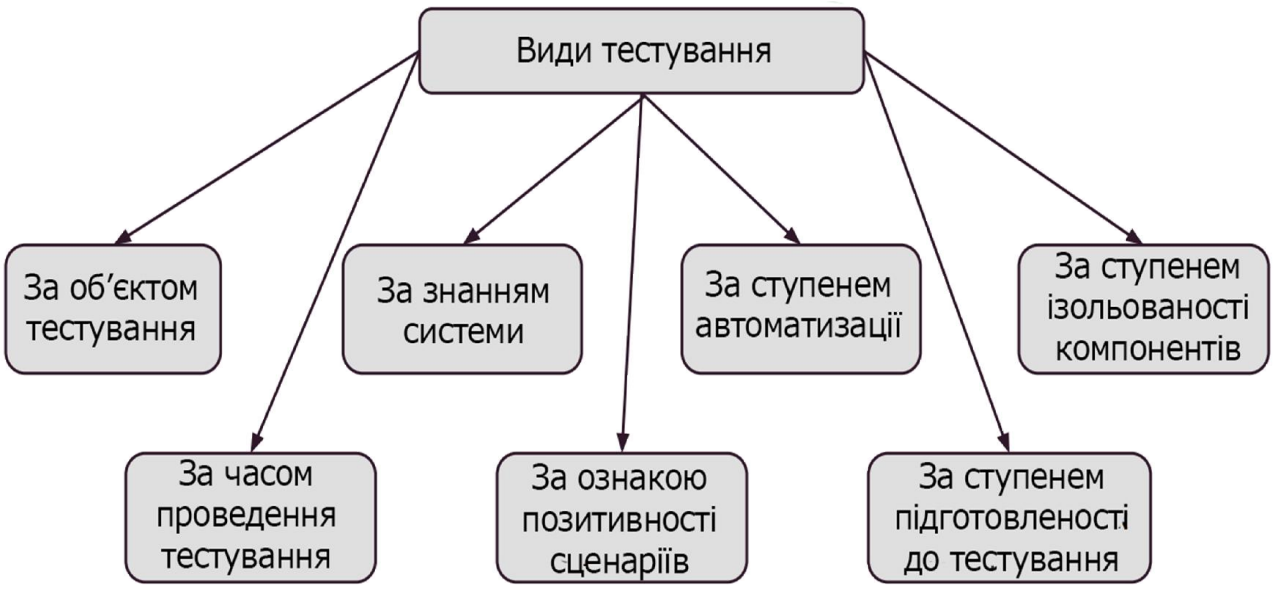

Рисунок 1.7 – Класифікація тестування ПЗ за ознаками

Види тестування за об'єктом тестування[13]:

– функціональне тестування (functional testing) розглядає наперед вказану поведінку і ґрунтується на аналізі специфікацій функціональності компоненту або системи в цілому;

– тестування продуктивності (performance testing), підвидами якого є:

1) навантажувальне тестування (load testing);
2) стрес-тестування (stress testing);
3) тестування стабільності (stability/endurance/soak testing).

– тестування безпеки (security testing) використовується для перевірки безпеки системи;

– тестування інтерфейсу користувача (UI testing);

– тестування зручності використання (usability testing) направлене на встановлення ступеню зручності використання, навчання, зрозумілості і привабливості для користувача в заданому контексті умов;

– тестування локалізації (localization testing) – це процес тестування локалізованої версії програмного продукту;

– тестування сумісності (compatibility testing) перевіряє здатність ПЗ працювати в певному середовищі (як апаратні платформи, так і програмні);

За знанням системи існують наступні види тестування[13]:

– тестування чорного ящика (black box) – відомі вхідні та вихідні дані, як відбувається перетворення невідомо;

– тестування білого ящика (white box) – відомі вхідні і вихідні дані та як відбувається перетворення;

– тестування сірого ящика (grey box) поєднує у собі два попередні види.

За ступенем автоматизації розділяються тестування на[13]:

– ручне тестування (manual testing);

– автоматизоване тестування (automated testing);

– напівавтоматизоване тестування (semiautomated testing).

За ступенем ізольованості компонентів[13]:

– компонентне (модульне) тестування (component/unit testing) полягає в тестуванні кожного окремого модуля системи;

– інтеграційне тестування (integration testing) – це тестування взаємодії скомбінованих в одне ціле частин програми;

– системне тестування (system/end-to-end testing) тестує інтегровану систему.

За часом проведення тестування виділяють[13]:

– альфа-тестування (alpha testing) – імітація реальної роботи з системою штатними розробниками або реальна робота з системою потенційними користувачами/замовником. Розділяється на:

1) тестування при прийманні або «димове» тестування (smoke testing).
2) тестування нової функціональності (new feature testing).
3) регресивне тестування (regression testing).

4) тестування при здачі (acceptance testing).

– бета-тестування (beta testing) - деяких випадках виконується поширення версії з обмеженнями (за функціональністю або часом роботи) для певної групи осіб, з тим щоб переконатися, що продукт містить достатньо мало помилок. Іноді бета-тестування виконується для того, щоб отримати зворотній зв'язок про продукт від його майбутніх користувачів.

Можна визначити наступні види тестування за ознакою позитивності сценаріїв[13]:

– позитивне тестування (positive testing);

– негативне тестування (negative testing).

За ступенем підготовленості до тестування виділяють[13]:

– тестування по документації (formal testing);

– тестування ad hoc або інтуїтивне тестування (ad hoc testing) — тестування без документації, яке базується на методиці передбачення помилки на власному досвіді тестувальники;

– дослідницьке тестування, де одночасно виконується дизайн тестів, виконання тестування та навчання.

Отже, в ході дослідження були розглянуті види веб-додатків та технології їх створення. А також було вияснено, що в часи розвитку інтернет технологій надійність веб-додатків є необхідністю. Було розглянуто основні методи тестування програмних засобів, для задоволення цієї необхідності. В ході аналізу виникла потреба у проектуванні цілої системи тестування веб-додатків та її програмної реалізації.

2 МЕТОДИ ТА ЗАСОБИ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

2.1 Системи управління конфігурацією програмними продуктами

Конфігураційне керування (англ. software configuration management, SCM) в програмній інженерії – комплекс методів, спрямованих на систематичний облік змін, що вносяться розробниками в програмний продукт в процесі його розробки та супроводу, збереження цілісності системи після змін, запобігання небажаних і непередбачуваних ефектів, формалізація процесу внесення змін[14].

Найпопулярніші сервери безперервної інтеграції програмного забезпечення:

– Jenkins інструмент безперервної інтеграції, написаний на Java. Запускається в контейнері сервлетів, таких як Apache Tomcat або GlassFish. Підтримує інструментарій для роботи з різними системами контролю версій, включаючи CVS, Subversion, Mercurial, Git і Clearcase, може збирати проекти Apache Ant і Apache Maven, а також виконувати shell-скрипти і команди Windows[15].

– BuildBot написаний в основному на Python і заснований на Twisted фреймворку. Починався як більш легка альтернатива Mozilla Tinderbox і зараз використовується в таких проектах як Mozilla, Chromium, Webkit і багатьма іншими. Buildbot встановлюється у вигляді основного сервера (master) і підлеглих (slave). З основного сервера здійснюється моніторинг змін в репозиторії, координація роботи підлеглих серверів і розсилка звітів користувачам і розробникам. Підлеглий сервер може запускатися під різними операційними системами. Налаштовується за допомогою конфігураційних скриптів Python на основному сервері. Ці скрипти для настройки компонентів збірки дуже прості в розумінні, однак володіють всією потужністю Python[16].

– CruiseControl – інструмент безперервної інтеграції програмного забезпечення на платформі Java, націлений на автоматизацію процесу складання. Управління та перегляд По збірки здійснюється через веб-інтерфейс.

Інтегрується з Apache Ant, різними системами управління версіями. Є відкритим програмним забезпеченням, поширюється під BSD-подібною ліцензією. Кроме Java-версии существуют варианты инструмента для платформы Microsoft .Net (CruiseControl.NET, CCNet) и версия для Ruby (CruiseControl.rb)[16].

– GoCD. Як і інші просунуті CI сервери, GoCD дозволяє виконувати збірки в різних системах і контролювати їх в одному місці. Регулярно виконувані дії можуть бути додані в джерела і потім вони викликаються для виконання. GoCD має простий і зрозумілий інтерфейс, а також детальну документацію[16].

– Strider написаний на JavaScript і його серверному фреймворку Node.JS. Використовує MongoDB для зберігання даних. Для початку роботи з Strider, необхідно встановити MongoDB і Node.js. Потім його просто встановити однією командою через установник пакетів Node.JS. Strider налаштовується за допомогу плагінів, проте, найімовірніше, буде потрібна ручна правка конфігураційних файлів, для того, щоб зробити його придатним для використання в своєму проекті[16].

– TeamCity серверне програмне забезпечення від компанії JetBrains написаний на мові Java, білд-сервер для забезпечення безперервної інтеграції[15].

– Travis CI ймовірно найпростіший CI сервер, для того щоб почати. Крім того, що він розповсбджується з відкритим вихідним кодом і відповідно безкоштовний при установці на свій сервер. Travis CI має SaaS версію, яка є безкоштовною для проектів з відкритим кодом. При реєстрації та налаштування просто потрібно закріпити свій GitHub аккаунт, отримати необхідні права і додати .travis.yaml файл в своєму проекті. Travis CI збере новий білд після додавання змін на GitHub[16].

У таблиці 2.1 порівнюються особливості деякого з найбільш популярного програмного забезпечення безперервної інтеграції. Порівняння проводиться на основі платформи, обчислювальної платформи, побудови і інструментів інтеграції та підтримки різних середовищ інтеграції[17].

Таблиця 2.1 – Порівняння серверів інтеграції[17].

Назва	Плат-форма	Побудова, Windows	Побудова, Java	Побудова, інші	Інтеграція, середовище розробки	Інтеграція, інші
Jenkins	Web container	MSBuild, NAnt	Ant, Maven 2, Kundo	Cmake, Gant, Gradle, Grails, Phing, Rake, Ruby, SCons, Python, shell script, command-line	Eclipse, IntelliJ IDEA, NetBeans	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac, HP ALM
BuildBot	Python	Command-line	Command-line	Command-line	-	-
Cruise Control	Cross-platform	NAnt, Rake, Xcode	Phing, Apache Ant, Maven	catch-all 'exec'	Eclipse	-
GoCD	Cross-platform	Command-line	Command-line	Command-line	Hi	GitHub
Strider	Node.js	Hi	Hi	C, C++, Clojure, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Scala	Hi	GitHub, Bitbucket, Heroku, GitHub Enterprise, Git
Team City	Web container	MSBuild, NAnt, Visual Studio, Duplicates finder for .NET	Ant, Maven 2-3, Gradle, IntelliJ IDEA .ipr based and Inspections and Duplicates finder	Rake, FxCop, command-line	Eclipse, Visual Studio, IntelliJ IDEA, RubyMine, PyCharm, PhpStorm, WebStorm	Jetbrains Youtrack, Jira, Bugzilla, FishEye, FindBugs, PMD, dotCover, NCover
Travis CI	Hosted	Hi	Ant, Maven, Gradle	C, C++, Clojure, Elixir, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Rust, Smalltalk	Hi	GitHub, Heroku

У наступній таблиці (таблиця 2.2) порівнюються особливості деякого з найбільш популярного програмного забезпечення безперервної інтеграції на основі системи керування версіями, що є невід'ємною частиною системи безперервної інтеграції програмного забезпечення. Деякі з популярних SCM наведені і, якщо вони підтримуються чи ні системами безперервної інтеграції, що згадується в таблиці[17].

Таблиця 2.2 – Підтримка найбільш популярних SCM систем розглянутими раніше серверам інтеграцій[17].

Назва	Darcs	Git	GNU Bazaar	Integrity	Mercurial	Perforce	PVCS	Subversion
Jenkins	Так	Так	Так	Так	Так	Так	Так	Так
BuildBot	Так	Так	Так	Ні	Так	Так	Ні	Так
Cruise Control	Так	Так	Ні	Так	Так	Так	Ні	Так
GoCD	Ні	Так	Ні	Ні	Так	Так	Ні	Так
Strider	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні
Team City	Ні	Так	Ні	Ні	Так	Так	Ні	Так
Travis CI	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні

2.2 Засоби модульного тестування веб-додатків

Фреймворк jUnit є досить вдалим вирішенням задач, пов'язаних з тестуванням java-додатків. jUnit застосовується для модульного тестування, яке дозволяє перевіряти на правильність окремі модулі вихідного коду програми. Перевага даного підходу полягає в ізолюванні окремо взятого модуля від інших. При цьому, мета такого методу дозволяє програмісту упевнитися, що модуль, сам по собі, здатний працювати коректно. jUnit – це бібліотека класів.

Альтернативним фреймворком є TestNG, який призначений для тестування і поєднує в собі JUnit та NUnit. В TestNG додані нові інноваційні функції, що роблять його більш потужним та простим в користуванні.

Можливості TestNG:

– анотація (annotations);
– використання XML для гнучкого конфігурування тестів;
– підтримка data-driven тестування (за допомогою анотації @DataProvider);
– залежні методи для тестування серверних додатків;
– підтримується в Eclipse, IDEA, Ant, Maven, Netbean, Hudson;
– тестування коду проходить багатопоточно, що дає безпеку і швидкодію;
– легкий перехід від JUnit[18].

NUnit – відкрите середовище юніт-тестування додатків для .NET. Бібліотеку було перенесено з Java (JUnit).

PHPUnit – це спеціальний фреймворк, призначений для тестування скриптів мови PHP. Переваги PHPUnit:

– інструменти для створення модульних тестів і організації їх в ієрархічні набори;
– інтерфейс командного рядка для виконання тестів;
– провайдери даних – генератори наборів даних, для тестування елементів скрипта, використовуючи різні вхідні параметри;
– підтримка тестування коду, що працює з базою даних;
– тестування винятків;
– підтримка так званих фіктивних об'єктів;
– генератори звітів[19].

Виконаємо порівняльний аналіз даних інструментів. Напишемо функцію обчислення факторіалу для всіх інструментів модульного тестування. Складемо графік виконання даної функції в залежності від часу (рисунок 2.1).

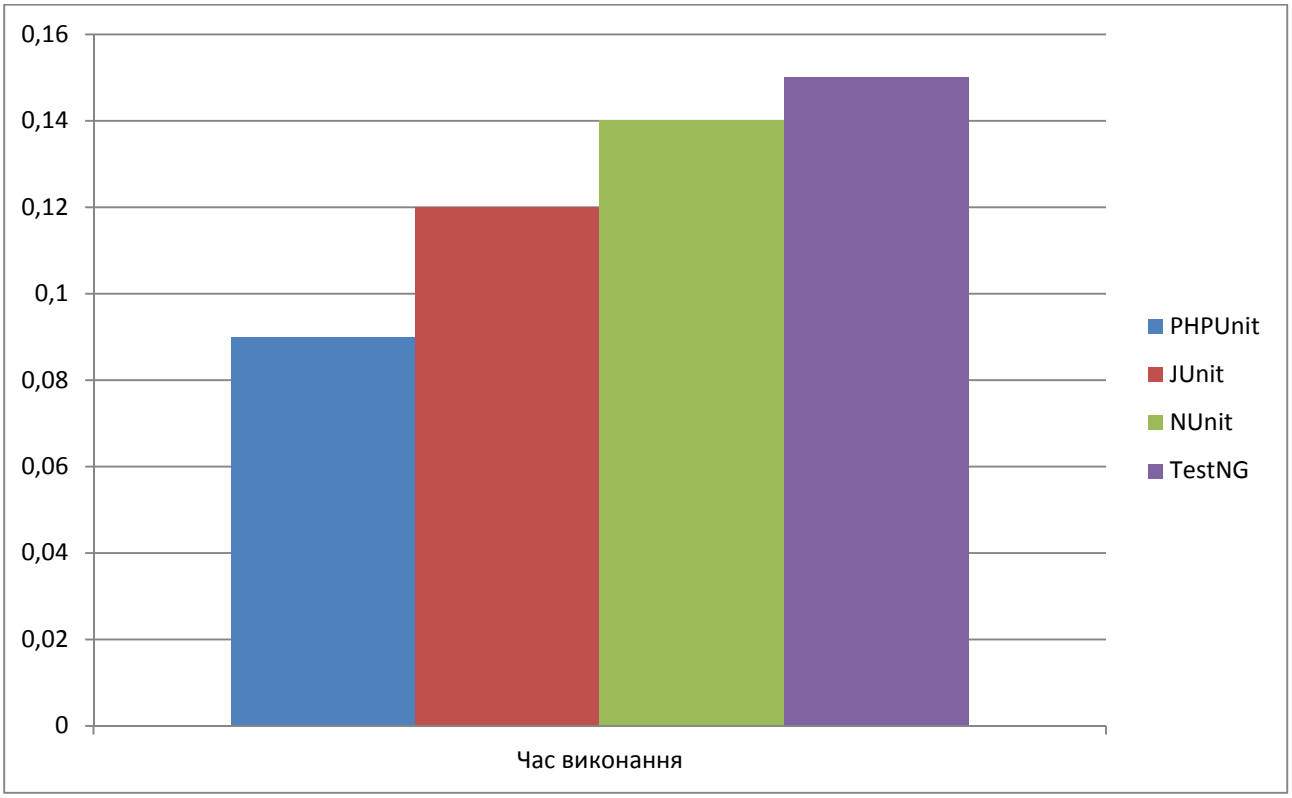

Рисунок 2.1 – Статистика виконання функції обчислення факторіалу інструментами модульного тестування в залежності від часу

Як видно із рисунку PHPUnit є найшвидшим. Отже, саме його будемо використовувати при написанні модульних тестів.

2.3 Алгоритми та інструменти функціонального тестування веб-додатків

Забезпечення якості програмного забезпечення є невід'ємною частиною життєвого циклу розробки, і з ускладненням тестованих програм з'явилася необхідність автоматизувати і контролювати процес тестування. багато великих компанії, що займаються розробкою програмного забезпечення мають власні напрацювання в даній області. Яскравим прикладом є компанія HewlettPackard Company (HP) має великий набір рішень для тестування програмного забезпечення[21]. З виділенням процесу забезпечення якості як окремої частини розробки програмного забезпечення зріс і попит на різні інструменти для тестування[22,23].

На сьогодні існує велика кількість інструментів для тестування програмного забезпечення: ресурси, відокремлені для тестування чи розробки програмного забезпечення, містять опис до кількох сотень різних інструментів[24,25]. Звичайно при такій кількості різних рішень виникає питання на рахунок раціонального вибору інструментів для тестування, але у випадку тестування систем, які мають ряд специфічних особливостей, вибір варто робити враховуючи всі особливості архітектури системи, а також підсистем, що входять у неї. Список інструментів тестування, які реалізовують найбільш розповсюджені тестові підходи, достатньо обширний. Тому варто провести аналіз інструментів тестування, що найчастіше використовуються.

Дослідивши інструменти для функціонального тестування, проаналізувавши різні ресурси[26,27,28], а також ознайомившись зі статистикою пошукових запитів по даній темі, можна виділити десять інструментів: п’ять комерційних та п’ять з відкритим вихідним кодом. Список інструментів по яким була зібрана статистика:

- Комерційні інструменти (менш привілейовані до використання):
    1) HP Quick Test Professional (HP QTP);
    2) SilkTest;
    3) Telerik TestStudio;
    4) TestComplete;
    5) Ranorex.
- Інструменти з відкритим вихідним кодом (більш привілейовані до використання):
    1) Selenium;
    2) AutoIt;
    3) SoapUI;
    4) Sahi;
    5) Watir.

На рисунку 2.2 зображена статистика пошукових запитів на рахунок комерційних інструментів тестування, а на рисунку 2.3 – список інструментів тестування з відкритим вихідним кодом.

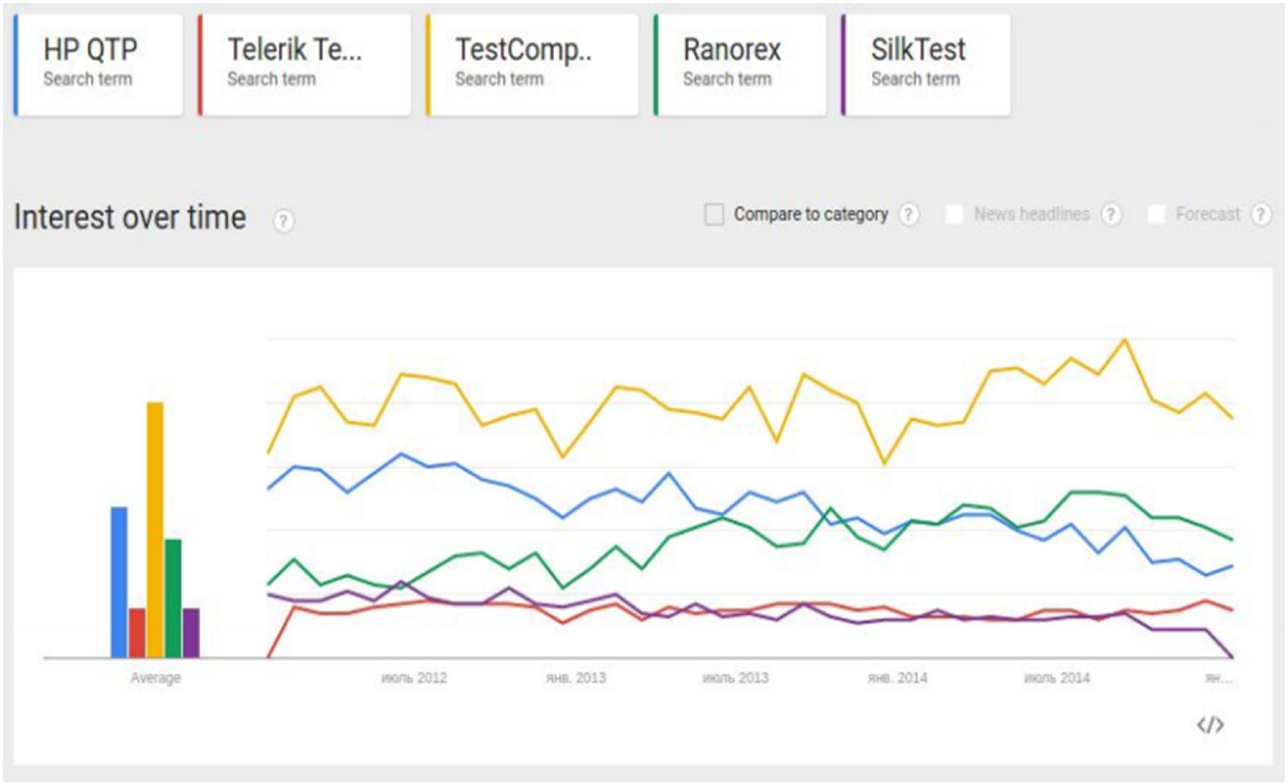

Рисунок 2.2 – Статистика пошукових запитів на рахунок комерційних інструментів тестування, сформована за допомогою сервісу Google Trends

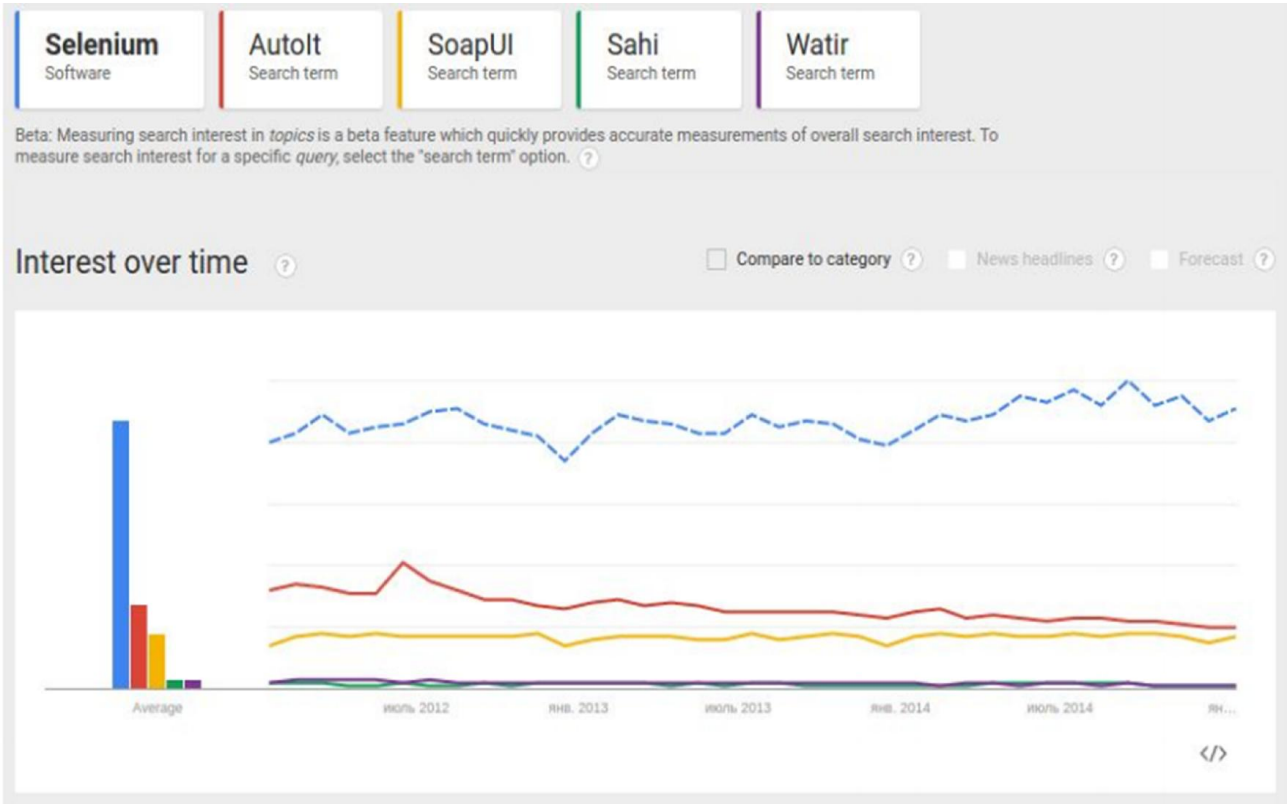

Рисунок 2.3 – Статистика пошукових запитів на рахунок інструментів тестування з відкритим вихідним кодом, сформована за допомогою сервісу Google Trends

На основі зібраних даних було вирішено вибрати п’ять з них, а саме:

- Selenium[29];
- HP Quick Test Professional (HP QTP)[30];
- Telerik TestStudio[31];
- TestComplete[32];
- Ranorex[33].

Selenium – це набір інструментів, що призначений для автоматизації веб-браузерів на різних платформах. Selenium може автоматизувати безліч різноманітних браузерів на різних платформах, використовуючи різні мови програмування і інтегруючись з різними тестовими фреймворками. Найбільш популярною областю застосування Selenium є автоматизація тестування веб-додатків. Однак за допомогою Selenium можна автоматизувати будь-які інші рутинні дії, що виконуються через браузер. Розробка Selenium підтримується виробниками популярних браузерів. Вони адаптують браузери для більш тісної інтеграції з Selenium, а іноді навіть реалізують вбудовану підтримку Selenium в браузері. Selenium є центральним компонентом цілого ряду інших інструментів і фреймворків автоматизації. Selenium підтримує десктопні та мобільні браузери. Selenium дозволяє розробляти сценарії автоматизації практично на будь-якій мові програмування. За допомогою Selenium можна організовувати розподілені стенди, що складаються з сотень машин з різними операційними системами і браузерами, і навіть виконувати сценарії в хмарах[20].

Selenium WebDriver – це надійний фреймворк автоматизації, здатний працювати з будь-яким браузером. Він забезпечує великий тестовий набір, що включає тести з досить складною логікою поведінки і перевірок.

Selenium WebDriver – набір бібліотек для різних мов програмування що дозволяють управляти браузером з програми, написаної на цій мові програмування.

Схема взаємодій між різними інструментами проекту Selenium зображена на рисуноку 2.4.

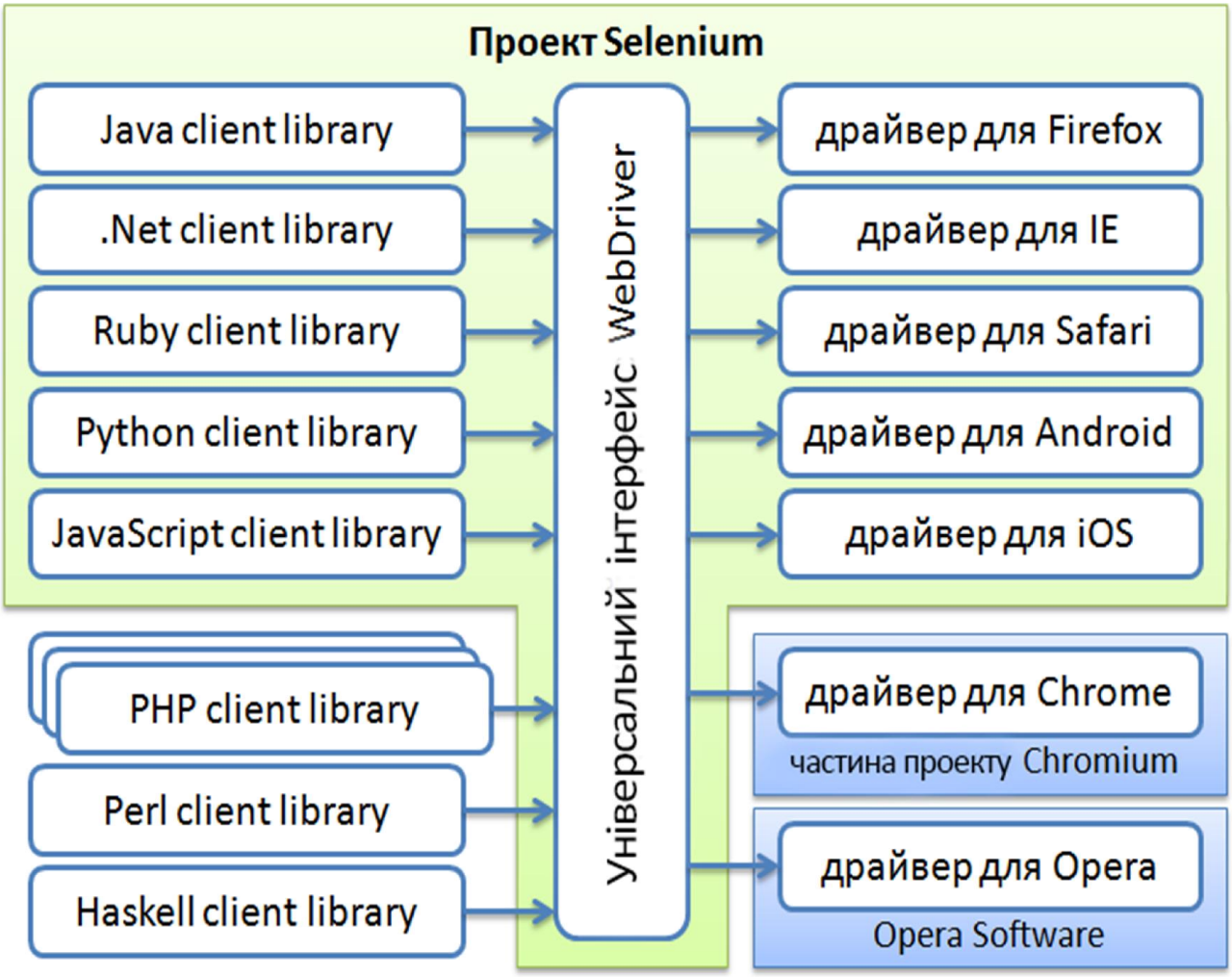

Рисунок 2.4 – Схема можливостей Selenuim WebDriver

В рамках проекту Selenium розробляються драйвери для браузерів Firefox, Internet Explorer і Safari, а також драйвери для мобільних браузерів Android і iOS. Драйвер для браузера Google Chrome розробляється в рамках проекту Chromium, а драйвер для браузера Opera (включаючи мобільні версії) розробляється компанією Opera Software. Тому вони формально не є частиною проекту Selenium, поширюються і підтримуються незалежно. Але логічно, звичайно, можна вважати їх частиною сімейства продуктів Selenium. Аналогічна ситуація і з клієнтськими бібліотеками - в рамках проекту Selenium розробляються бібліотеки для мов Java, .Net (C #), Python, Ruby, JavaScript[20].

HP QuickTest Professional (QTP) – один з провідних інструментів автоматизації функціонального тестування, є флагманським продуктом компанії HP в своїй лінійці. Для розробки автоматизованих тестів QTP використовує мову VBScript, яка підтримує такі технології: Windows

Presentation Foundation, Web services, Macromedia Flex, Ajax, Delphi, .NET, J2EEWeb, Visual Basic, ActiveX, Java, Oracle, SAP Solution, TE , PowerBuilder, Siebel, PeopleSoft, VisualAge, Stingray. Компанія HP рекомендує використання QTP в інтеграції з HP Quality Center для встановлення зв'язку тестів з вимогами, зберігання тестів, управління їх запуском, формування звітів[23].

У великій степені його популярність обумовлена наявністю в ньому рекордера для користувача активності, який дозволяє записати дії користувача і перетворити їх в скрипт. Об'єкти, з якими взаємодіє користувач, автоматично ідентифікуються QTP і зберігаються в спеціальне сховище – репозиторій. При збереженні в репозиторій, QTP автоматично зберігає ідентифікаційні властивості об'єкта, але робить це не завжди правильно. Наприклад, якщо на веб-сторінці присутні кілька таблиць (навіть якщо у кожної з них є свій ID), QTP ідентифікує їх за порядковими номерами. Такий спосіб ідентифікації об'єктів викликає проблеми при програванні автотестів. Більш того, багато об'єктів взагалі не потрапляють в репозиторій під час запису. Це викликано багатьма причинами, найбільш частими з яких є складна верстка або верстка із застосуванням DIV-ів[36].

QTP – це програмне забезпечення для автоматизованого тестування, призначеного для тестування різних програмних додатків і середовищ. QTP виконує функціональне і регресійне тестування за допомогою призначеного для користувача інтерфейсу, такі як рідний GUI або веб-інтерфейс. Працює шляхом ідентифікації об'єктів в інтерфейсі програми або веб-сторінки і виконанням необхідних операцій (наприклад, клацань миші або події клавіатури); також може захопити властивості об'єкта, як ім'я або ID. QTP використовує мову сценаріїв VBScript, щоб визначити процедуру тестування та маніпулювати об'єктами і елементами управління тестової програми.

Хоча QTP зазвичай використовується для інтерфейс направленої автоматизації test case, він може також автоматизувати деякі test case на основі випробувань, таких як операції файлової системи, тестування баз даних або тестування веб-сервісів.

Обробка винятків. HP QTP управляє обробкою виняткових ситуацій з використанням сценаріїв відновлення; мета полягає в тому, щоб продовжити виконання тестів, якщо відбувається непередбачена помилка. Оскільки QTP захоплює простір пам'яті тестованих додатків, деякі винятки можуть привести до того, що QTP мусить припинити виконання і бути невідновлюваним.

Тестування орієнтоване на дані. HP QTP підтримує тестування керованих даними. Наприклад, дані можуть бути виведені у вигляді таблиці даних для повторного використання в інших місцях. Тестування орієнтоване на дані реалізоване у вигляді книги Microsoft Excel, що можна отримати з QTP. HP QTP має два типи таблиць даних: глобальний лист даних і лист даних дій (локальних). Кроки тесту можуть зчитувати дані з цих таблиць даних для того, щоб управляти змінними даних в тестовій програма і для перевірки очікуваного результату.

Автоматизація і складні об'єкти користувальницького інтерфейсу. HP QTP може не розпізнати індивідуальні об'єкти користувальницького інтерфейсу і інші складні об'єкти. Користувачі можуть визначати ці типи об'єктів як віртуальні об'єкти. QTP не підтримує віртуальні об'єкти для аналогового запису або запису в режимі низького рівня.

QTP має можливість розширення за допомогою окремих надбудов для ряду середовищ розробки, які не підтримуються поза збіркою. Надбудови включають підтримку Web, .NET, Java і Delphi[37].

Telerik WebUI Test Studio – це інструмент для автоматизації функціонального тестування веб-додатків (ASP.NET, Silverlight). Являє собою розширення до Microsoft Visual Studio. До головних переваг програми можна віднести:

– підтримка Silverlight;

– даний інструмент автоматизації відмінно справляється з AJAX запитами будь-якої складності;

– немає проблем при написанні тестів з RadControls компонентами (елементи управління сторінкою) Telerik. Деякі RadControls елементи досить

важко піддаються автоматизації, наприклад Selenium на ранніх версіях не міг працювати з випадаючим списком і календарем;

– WebUI Test Studio має зручний редактор тестів;

– інтеграція з іншими засобами тестування. Доступні всі можливості Visual Studio;

– на хорошому рівні реалізована базова функціональність будь-якого розумного сучасного засоби автоматизації (Test Recorder, Elements Explorer, DOM Explorer, генерація коду, підтримка декількох браузерів і т.д.).

Мінусів у WebUI Test Studio кілька, по-перше він вміє працювати тільки з ASP.NET, Silverlight додатками. Також, варто відзначити, що в Visual Studio 2010 з'явилися в міру не дороблені Coded UI Tests, які надають приблизно такий же набір функціональності для автоматизації функціонального тестування, а з ліцензією навіть можна тестувати Silverlight додатки[38].

TestComplete – це автоматизований засіб тестування, що дозволяє створювати тести для Windows додатків, web серверів і web сторінок.

TestComplete використовується для автоматизації різних типів тестування програмного забезпечення для .NET, Java, Visual C++, Visual Basic, Delphi, C++ Builder, web сторінок та інших додатків. Інструмент призначений для розробників програмного забезпечення і відділів тестування для того, щоб зменшити тимчасові витрати, необхідні на ручне тестування.

У TestComplete існує спеціальний механізм, який полегшує створення скриптів для тестування додатків. Суть механізму полягає в наступному: тестувальник виконує необхідні дії, які автоматично записуються в файл. Даний файл, який доступний для перегляду і редагування, і при запуску, повторить всі ті операції, які були попередньо виконані. Важливою обставиною є той факт, що записані тести можуть бути змінені пізніше[22].

Основні характеристики TestComplete:

– Тестування Windows- і web-додатків. Програма не залежить від типів додатків і засобів розробки. Тести працюють для програм, створених на C #, C ++, Delphi, Java і будь-яких інших мовах.

– Кросбраузерність web-тестування. Створення тестів в одному браузері і запуск в іншому підтримуваному без якої-небудь зміни тесту.

– Тестування додатків HTML5. Автоматизація тестування HTML5-контенту, включаючи нові теги, web-форми і пов'язаний з ними вміст JavaScript і CSS3.

– Вибір типу тестування. TestComplete дозволяє вибирати з різних типів тестування: тестів графічного процесора, розподілених тестів і тестів під управлінням даних.

– Тестування ключових слів. Незалежні від скриптів тести ключових слів і простий користувальницький інтерфейс дозволяють швидко запускати створення автоматизованих перевірок.

– Функціональне тестування RIA-додатків. Перевірка надійності і функціоналу додатків HTML5, Flash, Flex, AIR і Silverlight.

– Інтегрована підтримка сторонніх рішень. Інтеграція з JIRA, Axosoft, Bugzilla і т.д. Підтримка найбільш популярних елементів управління користувацьким інтерфейсом, включаючи extJS 4, JavaFX2, Developer Express, Infragistics, Rogue Wave, Telerik, .NET.

Ranorex – це повноцінна сфера розробки автоматизації тестування GUI. Ranorex також є набором інструментів і бібліотек для написання тестів, призначених для тестування настільних, веб-орієнтованих і мобільних додатків.

Дане середовище надає наступні можливості:

– Підтримка динамічно генерованих графічних елементів управління (контролів) за допомогою GUI Object Repository. Для доступу до елементів використовується мова XPath, тому вибрати елемент можна згенерувавши за допомогою вбудованого рекордера відповідний запит.

– Гнучка і настроювана система пошуку контролів. У Ranorex XPath вирази називаються RanoreXPath. Якщо елемент не має унікальної назви, то звернутися до нього можна в одну дію, знаючи XPath вираз для нього.

– Проста підтримка тестів, що базується на даних (Data Driven Testing). У Ranorex також є можливість розробляти Data Driven-тести, але виключаючи формат XML для даних.

– Можливість розробляти свої модулі (фреймворки) і використовувати їх при розробці тестів на C #. Ranorex надає своє середовище розробки – Ranorex Studio, яке засноване на SharpDevelop і тому має дуже схожий з Visual Studio інтерфейсом і принциплм роботи. У Ranorex зручний інспектор елементів і можливість створювати тести практично тільки кліками мишки. Крім цього, є можливість розробляти додаткові програмні модулі, які потім будуть використані для створення більш складних тестових сценаріїв.

– Підтримка запуску тестів на сервері безперервної інтеграції (TeamCity). Ranorex, в свою чергу, вміє компілювати Test Suit в виконуваний файл, який потім можна просто запустити як окреме завдання збірки TeamCity.

– Генерація інформативних звітів по результату прогону тестів. Середовище розробки Ranorex надає прекрасні звіти з ілюстраціями і поясненнями. Однак, для цього тести необхідно запускати з самої Ranorex Studio. Скомпільовані ж тести після прогону повертають звіт Ranorex XML, який потім можна конвертувати в xUnit.

– Можливість інтеграції тестів з тест-кейсами системи тест-менджменту (TMS). У Ranorex Studio тестовий проект збирається в exe-файл. Тобто, при поточній реалізації утиліти немає можливості використовувати його в якості джерела тест-кейсів.

– Простота вивчення та використання тестувальниками. Ranorex спеціально створений як середовище тестувальника. З його допомогою зручно створювати, запускати і переглядати результати тестів. Крім того, після розробки кількох прикладів для тестового додатка, проблем з пошуком динамічних елементів не виникає. Ranorex також успішно працював з додатками на .NET WinForms.

Перевіримо можливість запуску тестів за розкладом і з прив’язкою до системних подій.

Практика використання планувальників для організації розкладу виконання різних додатків не нова і відображається у багатьох програмних продуктах. Всі проаналізовані інструменти мають можливість роботи за розкладом. Наприклад, Telerik TestStudio, TestComplete і HP QTP дозволяють виконувати тести в запланований час без додаткових плагінів чи настройок. А

розклад в Selenium і Ranorex можна реалізувати за допомогою серверів безперервної інтеграції. Прив'язку до системних подій можна здійснити за допомогою зовнішніх плагінів та утиліт. Наприклад, за допомогою Skybot Scheduler можна організувати запуск тестів без жорсткої прив'язки до часу, а по будь-якій події[35].

Незважаючи на велику кількість способів реалізації прив'язки виконання тестів до часу чи системним подіям, всі способи мають свої недоліки. Ними є складана конфігурація додаткових утиліт і відносно невеликий функціонал вбудованих інструментів планування.

Перевіримо можливість одночасного виконання тестів.

Всі переглянуті інструменти дозволяють виконувати тестові сценарії паралельно в декількох тестових середовищах, що дозволяє розподілити навантаження при тестуванні, а також скорочувати час прогону незалежних тестів.

Наприклад, для вибраних комерційних інструментів дана функція включена в поставку за замовчуванням. У випадку із Selenuim необхідна інтеграція із зовнішніми модулями. Так, дане рішення в поєднанні з бібліотекою TestNG дозволяє виконувати кілька тестів паралельно, але з деякими обмеженнями:

- необхідна складна мережева архітектура типу клієнт-сервер;
- тести фактично будуть виконуватися в різних середовищах.

Однак з одночасним виконанням тестових скриптів все трохи складніше, тому що для всіх інструментів, тест є найменшою неподільною одиницею автоматизації. Звичайно, можна організувати тестовий скрипт як псевдопаралельний, але це суттєво ускладнює роботу тестувальників, а у випадку з тестуванням графічного інтерфейсу стає практично неможливим.

Також слід проаналізувати ці інструменти щодо загальних вимог:

- можливість взаємодії з системою за допомогою графічного інтерфейсу;
- можливість взаємодії з системою через API;
- тестування Back-End;
- гнучкість і універсальність тестових скриптів;

– простота використання.

У Test Complete тести можуть створюватися в обох форматах за допомогою вбудованих редакторів. Цей інструмент підтримує такі мови, як: VBScript, JScript, DelphiScript і інші. Quick Test Professional дозволяє записувати дії користувача в тестованому додатку. Ці дії зберігаються у вигляді скриптів, які можуть бути виконані в подальшому і можуть бути представлені в двох форматах: як у вигляді коду на VBScript (expert view), так і в якості набору послідовних кроків з діями (keyword view).

Selenium IDE також надає можливість записувати дії користувача, в даному випадку, у вигляді пар Command - Target, які можуть бути експортовані в декількох форматах: HTML, Java, Groovy, C#, Python і інші. У Ranorex і Test Studio використовується Keyword-driven підхід.

Незважаючи на те, що всі розглянуті інструменти є потужними і комплексними рішеннями, вони розроблені з нахилом на тестування широко використовуваних програм і додатків. Дана спрямованість, безумовно важлива на ринку програмного забезпечення, але це значно ускладнює їх конфігурацію для більш специфічних систем проведення розрахунків і клірингу, тим самим зменшуючи ефективність процесу тестування і підвищуючи витрати на забезпечення якості програмного забезпечення (час, навчання, налаштування середовищ та інструментів)[39].

Виконаємо порівняльний аналіз досліджених інструментів. Напишемо функціональний тест для сторінки vidfall.com для кожного проаналізованго інструменту. Задля точніших даних часу виконання тесту виконаємо кожен тест по 5 разів. З отриманої інформації складемо порівняльний графік часу виконання даного тесту

Як визно із рисунку 2.5 інструмент Selenium WebDriver є найшвидшим, тобто виконується він дуже мало часу. Також Hewlett Packard QuickTest Professional має гарні показники, але він є нестабільним (виконання тесту показує не однаковий час). TestComplete показав найгірші показники часу виконання тестової функції.

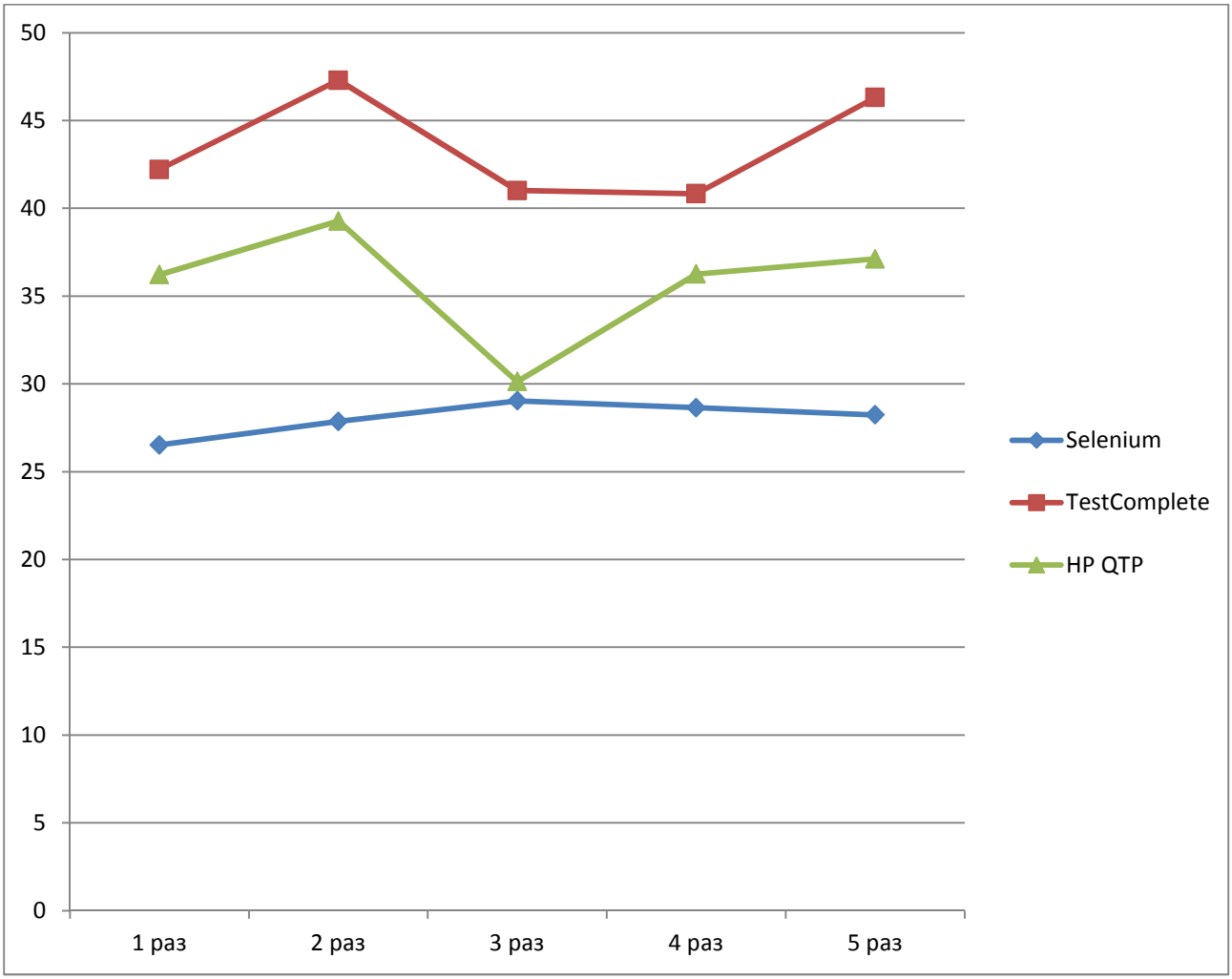

Рисунок 2.5 – Статистика виконання функціонального тесту для сторінки vidfall.com в залежності від часу

Отже, з проведено аналізу виберемо найкращий інструмент для написання функціональних тестів – Selenium WebDriver.

2.4 Засоби тестування прикладного програмного інтерфейсу веб-додатку

Прикладний програмний інтерфейс (API – англ. Application Programming Interface) дозволяє здійснювати зв'язок і обмін даними між двома окремими програмними системами. Система програмного забезпечення, що реалізовує API, містить функції (підпрограми), які можуть бути виконані за допомогою іншої системи програмного забезпечення.

Тестування API повністю відрізняється від тестування графічного інтерфейсу і в основному концентрується на шарі бізнес-логіки архітектури програмного забезпечення. Це тестування не буде концентруватися на зовнішньому вигляді додатків.

Замість використання стандартних даних, що вводяться користувачем (клавіатурою) і виходів, в API тестуванні, використовується програмне забезпечення для відправки викликів до API, отримується вихід, і записується відповідь системи.

API тестування потребує взаємодії додатку з API. Для того, щоб протестувати API, потрібно:

– використовувати інструмент тестування для управління API.

– написати код, щоб перевірити API.

Тест-кейси API тестування засновані на:

– Значення, що повертається засноване на вхідному стані. Це відносно легко перевірити, так як вхідні дані можуть бути визначені і результати можуть бути автентифіковані.

– Неповернення результату. Коли немає значення для повернення, поведінка API в системі може бути перевірена.

– Викликання іншого API / події / переривання. Якщо вихід з API викликає якусь подію або переривання, то ці події і переривання будуть відслідковані.

– Оновлення структури даних. Оновлення структури даних буде мати певний результат або вплив на систему, і це має бути автентифіковано.

– Зміна певних ресурсів. Якщо API-виклик змінює деякі ресурси, то воно повинно бути підтверджено шляхом доступу до відповідних ресурсів.

Основи підходу до тестування API:

– розуміння функціональності API програми і чітке визначення сфери охоплення програми;

– застосування методів тестування, такі як класи еквівалентності, аналіз крайового значення і помилки вгадування, написання тестів для API;

– вхідні параметри для API повинні бути заплановані і визначені належним чином;

– виконання тестових прикладів і порівняння очікуваного і фактичного результатів.

Різниця між тестуванням API і модульним тестуванням представлена у таблиці 2.3[40].

Таблиця 2.3 – Порівняльна таблиця модульного і API тестування

Модульне тестування	Тестування API
Зазвичай виконують розробники	Зазвичай виконують тестувальники
Тестуються окремі функції	Тестується вся функціональність системи
Розробник може звертатися до вихідного коду	Тестувальник не може звертатися до вихідного коду
Включене тестування графічного інтерфейсу	Тестуються лише API функції
Тестуються лише базові функції	Тестуються усі функціональні питання
Обмежене застосування	Широка сфера застосування
Зазвичай проходить до початку роботи	Проходить після створення побудови

Оскільки API і модульне тестування обидва направлені на вихідний код, подібні інструменти для тестування можуть бути використані для тестування обох:

- SOAPUI;
- Runscope;
- Postman with jetpacks;
- Postman with Newman;
- PHPUnit+Curl;
- Cfix;
- Check;
- CTESK;
- dotTEST;

– Eclipse SDK tool.

Отже, в ході аналізу було розглянуто найпопулярніші на сьогоднішній день інструменти тестування веб-додатків. Було досліджено їх продуктивність на прикладі роботи з реальними даними в існуючій системі, заміряно їх швидкодію та характеристики, та побудовано графіки залежностей характеристик. Аналізуючи графіки, було зроблено висновки про слабкі і сильні сторони методів і зроблений висновок про їх ефективність. Було розглянуто перспективи і можливості застосування технологій безперервної інтеграції. Були вибрані конкретні інструменти і сервіси, які необхідні для програмної системи тестування веб-додатків.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ МУЛЬТИКРИТЕРІАЛЬНОГО ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Налаштування системи безперервної інтеграції проекту

Систему тестування будемо реалізовувати на базі системи Vidfall. Даний веб-сервіс написаний на мові програмування PHP.

Отже, для об’єднання усіх видів тестування і безперервної розгортки та безперервного автоматичного тестування системи налаштуємо сервер безперервної інтеграції Jenkins. Завдяки цьому ми і отримаємо суцільну систему тестування веб-додатку.

Після успішної установки стартова сторінка Jenkins виглядатиме, як показано на рисунку 3.1.

Рисунок 3.1 – Панель налаштувань Jenkins після установки

Перейдемо до створення проекту. Для цього вибираємо пункт меню «Новий Item», вводимо назву проекту (VF) та його тип, в даному випадку – freestyle project (рисунок 3.2). Проект VF буде призначений для unit тестів.

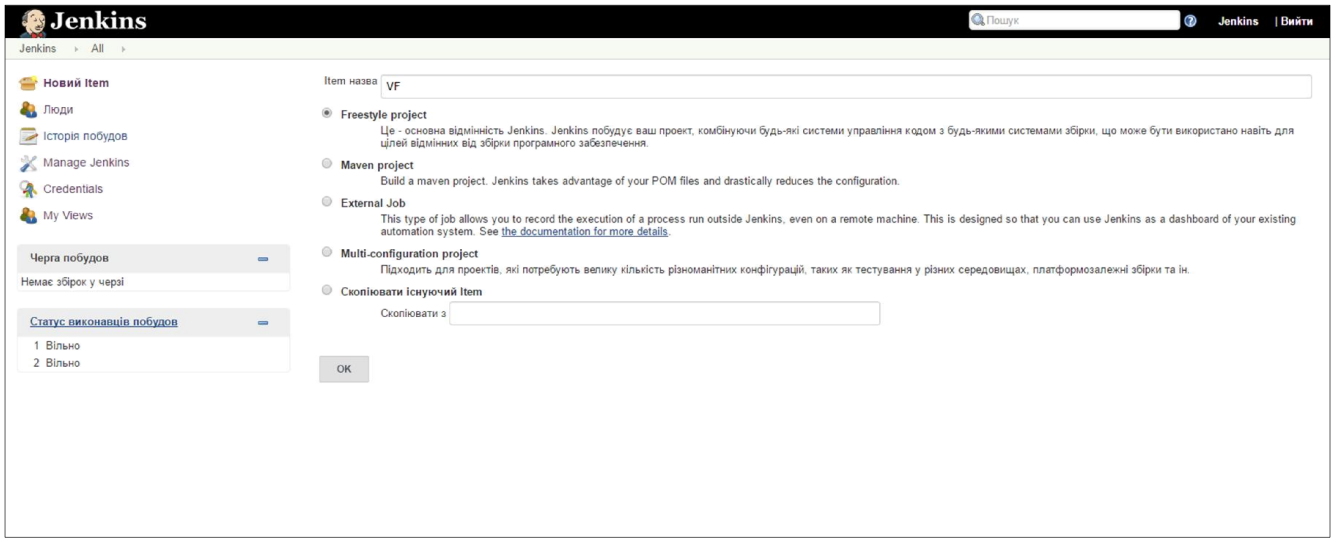

Рисунок 3.2 – Панель створення нового проекту

На сторінці налаштування проекту міститься наступна інформація:

– Project name: ім'я проекту; при його зміні також змінюється ім'я елементу – VF;

– Description: опис завдання;

– Strategy: стратегія ведення журналу, кількість журналів – Log Rotation;

– Parameterized: визначення змінних проекту. Існують змінні різного типу (параметр файлу, параметр тексту, параметр рядка і т.д.);

– Where: обмежує область виконання проекту;

– Advance configuration: додаткові специфікації для управління способом збірки проекту.

Плагіни, вибрані для установки, впливають на категорії і функції, які будуть доступні в проекті. Ось деякі можливі категорії і функції:

– Source code management: інструмент управління вихідним кодом – Git.

– Build triggers: метод запуску збірки – віддалено.

– Build: збірка – це найбільш важлива частина проекту. Вказуються точні дії на рахунок запуску проекту. Зазвичай це команди DOS для Windows або Shell для Linux. В нашому випадку це дві команди:

cd www/protected/ && echo 'yes' | php yiic.php migrate up -env=TEST_STAGING

cd www/protected/tests && /home/jenkins/vendor/bin/phpunit --log-junit report/unitreport.xml --coverage-html report --coverage-clover report/clover.xml unit/models

– Post-build actions: дії, що виконуються після збірки. Як правило, це відправлення повідомлень електронної пошти, запуск інших збірок або публікація звітів про результати. Команда:

unset AWS_ACCESS_KEY_ID && unset AWS_SECRET_KEY && cd /var/build && vendor/phing/phing/bin/phing -Denv=staging -Dbranch=dev -Dmessage="Jenkins $BUILD_DISPLAY_NAME"

Збережений проект знаходиться в списку на сторінці http://dev.vidfall.com:8080/job/VF/.

Jenkins дозволяє запускати збірку проекту вручну або автоматично. Існують різні механізми запуску збірки. Якщо вибрати автоматичну збірку, то при налаштуванні проекту можна встановити значення параметра Build Triggers (рисунок 3.3). Можливі варіанти:

– збірка проекту після складання інших проектів: після налаштування проекту можна визначити, чи потрібно після цього проекту збирати інші проекти. Цей варіант обирається, якщо проект залежить від інших проектів;

– дистанційний запуск збірки (наприклад, зі сценарію): збірка проекту запускається з іншої системи або з іншого вузла. Наприклад, збірку проекту можна викликати по електронній пошті або направити запит на складання зі сценарію;

– періодична збірка: створення графіку періодичної збірки проектів відповідно до конфігурації;

– опитування SCM: цей варіант збирає проект шляхом внесення змін до початкового коду. В цьому випадку вказують, як часто Jenkins повинен опитувати систему управління версіями. При наявності змін вихідного коду виконується складання проекту.

Рисунок 3.3 – Параметри запуску збірки

Мета безперервного розгортання – забезпечити ефективний і високоякісний процес розробки, випробування, розгортання і запуску програмного забезпечення в виробництво. При безперервному розгортанні кожну зміну в будь-якій частині системи програмного забезпечення (на рівні інфраструктури, додатку або налаштуванню даних) постійно переноситься в виробничу середу за допомогою спеціального конвеєра розгортання.

Для реалізації безперервного розгортання потрібно швидкий автоматичний процес розгортання наборів змін. Процес розгортання складається з декількох кроків. Стандартний процес полягає в наступному:

– внесення змін у код;
– виконання збірки інструментами управління вихідним кодом;
– здійснення автоматичного тестування;
– встановлення збірки.

Процес безперервного розгортання представлений на рисунку 3.4.

Топологія системи безперервного розгортання заключається в тому, що на відміну від традиційного процесу розгортки: розробник фіксує набір змін на сервері управління вихідним кодом, а потім інший сервер збірки виконує збірку – в процесі безперервної розгортки після додання Jenkins зявляється ведуча машина Jenkins. Ведуча частина Jenkins використовує, наприклад Git, для завантаження вихідного коду і запусає виконання збірки. Середовища розробки, тестування і виробництва грають роль ведених машин Jenkins. Ними управляє ведуча машина Jenkins, а вони виконують проекти по встановленню. Тестові середовища виконують проект функціонального тестування.

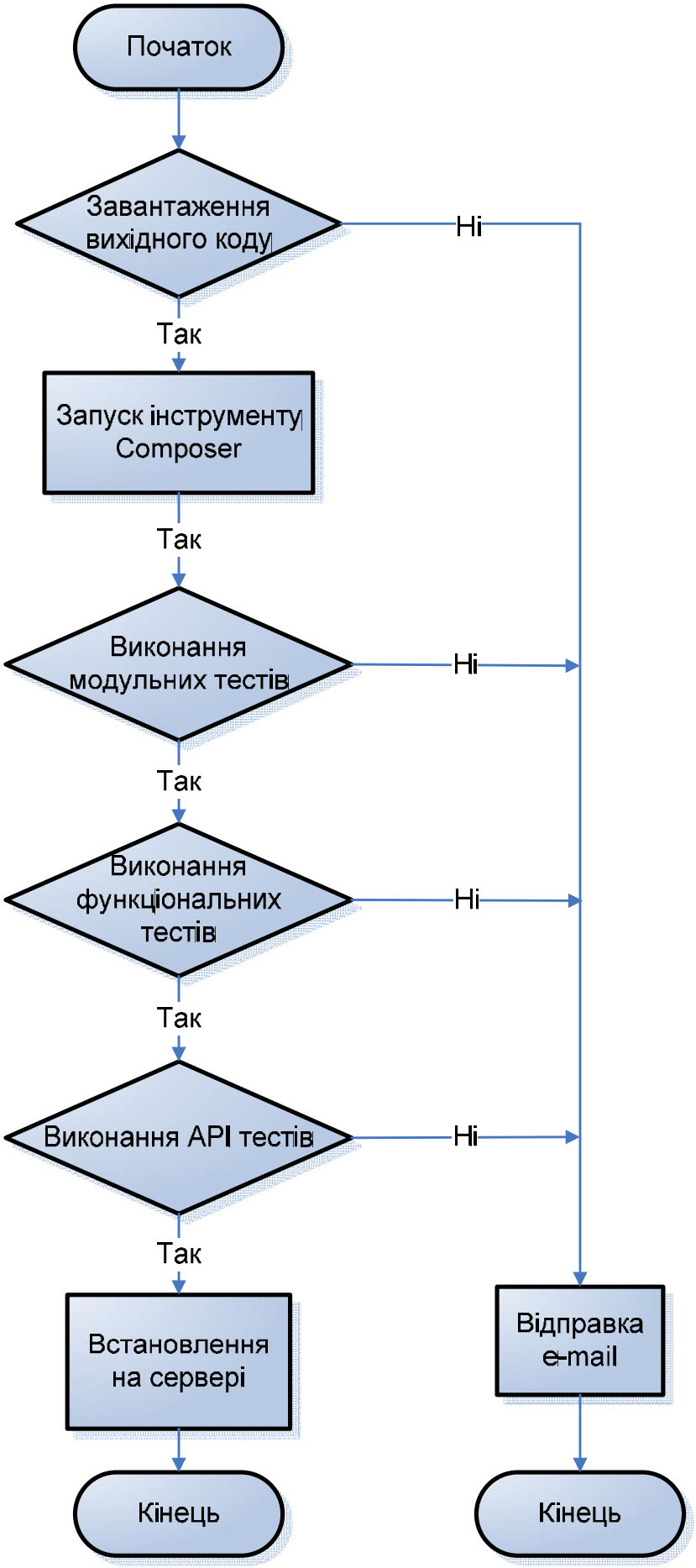

Рисунок 3.4 – Блок-схема алгоритму побудови проекту

В призначений час стартує побудова проекту в Jenkins. Побудова проекту складається із:

1. Оскільки на проект використовує інструмент управління вихідним кодом Git, потрібно виконати команду git checkout для отримання всього коду проекту.

2. Для того, щоб спростити підключення всіх необхідних для тестування бібліотек використаємо інструмент Composer – менеджер пакетів прикладного програмування для мови програмування PHP, який забезпечить єдиний формат для управління залежностями та потрібними бібліотеками[41]. Тобто спочатку описуємо бібліотеки, від яких залежить код. Далі виконуємо команду Composer install. І тепер Composer автоматично знайде потрібні версії вказаних бібліотек для всього проекту, завантажить їх і встановить в папку проекту.

3. Наступним кроком є виконання модульних тестів – PHPUnit.

4. Далі виконуються функціональні тести – Selenium.

5. І останнім тестуються API функції системи.

6. Якщо тести пройшли, побудова була успішною, то Jenkins створює zip-архів, який розгортає і встановлює збірку на сервері.

Якщо сталась помилка, то виконання побудови зупиняється і позначається як неуспішне. Також надсилається e-mail сповіщення адміністратору, що побудова не пройшла.

Головна сторінка Jenkins при успішних побудовах проекту матиме вигляд, як зображено на рисунку 3.5.

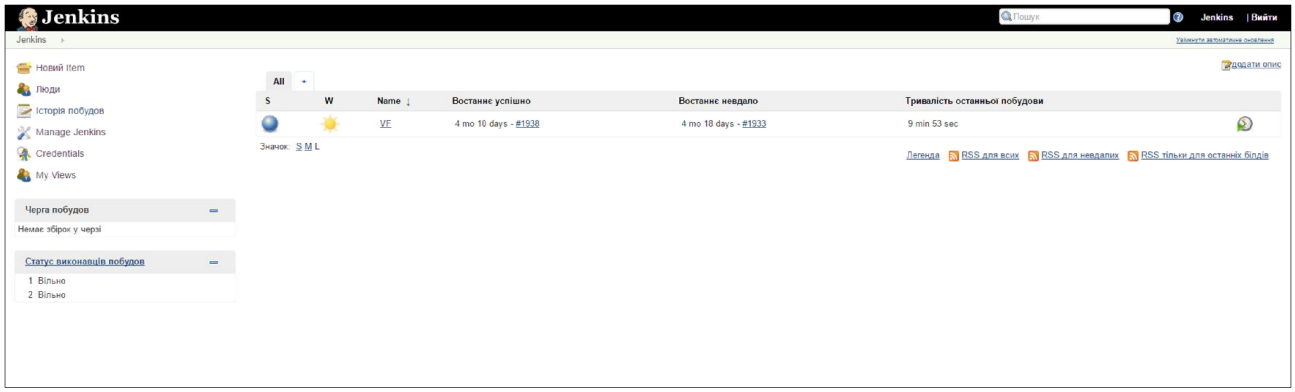

Рисунок 3.5 – Проект VF на головній сторінці Jenkins

3.2 Програмна реалізація функціональних тестів веб-додатку

Для написання Selenium тестів спочатку обов'язково потрібно підключити необхідні бібліотеки.

Оператор package, повідомляє транслятору, в якому пакеті повинні визначатися класи, які є в цьому файлі. Пакети задають набір роздільних просторів імен, в яких зберігаються імена класів. Якщо оператор package не вказано, класи потрапляють в безіменне простір імен, що використовується за умовчанням. Тому вказуємо необхідний шлях:

package com.wv.vf.ui;

Після оператора package, але до будь-якого визначення класів у вихідному Java-файлі, може бути присутнім список операторів import. Пакети є хорошим механізмом для відділення класів один від одного, тому всі вбудовані в Java класи зберігаються в пакетах. Тому наступними будуть команди import для всіх потрібних елементів[42].

WebDriver – це найменування ключового інтерфейсу, з використанням якого повинні писатися тести, однак існує кілька реалізацій цього інтерфейсу. По замовчуванню використаємо браузер Mozilla Firefox. Відповідно драйвер який забезпечить роботу із ним – Firefox Driver. Далі підключимо клас для створення його екземпляру:

import org.openqa.selenium.firefox.FirefoxDriver;

Основний клас для тестування:

import org.openqa.selenium.*;

Також потрібно підключити всі необхідні класи для роботи з функціями:

- Action – import org.openqa.selenium.interactions.Action;
- Actions – import org.openqa.selenium.interactions.Actions;
- Wait – import org.openqa.selenium.support.ui.WebDriverWait;
- Select – import org.openqa.selenium.support.ui.Select;
- основні функції Java – import org.junit.*;
- randomString() – import java.util.Random;

– implicitlyWait (TimeUnit) – import java.util.concurrent.TimeUnit;
– DesiredCapabilities() – import org.openqa.selenium.remote.DesiredCapabilities;

Оголосимо статичною бібліотеку для виведення даних:

import static org.openqa.selenium.OutputType.*;

Для роботи із тегами мови розмітки сторінки HTML підключимо необхідну бібліотеку:

import org.openqa.selenium.htmlunit.HtmlUnitDriver;

Наступним блоком коду є підключення потрібних об’єктів дати, тексту, патернів і функцій роботи з файлом, зі вхідним потоком даних, із виключеннями в блоці try-catch, і властивостями та часом на Java:

```
import org.apache.commons.io.FileUtils;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Date;
import java.text.*;
import java.util.Properties;
import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
```

У проекті Vidfall, який ми тестуємо, є системно-важлива функція авторизації користувача. Оскільки без неї (або у випадку некоректної роботи даної фунції) неможлива подальша робота із системою, виконаємо функціональне тестування функцій входу і виходу із системи. Також впродовж тесту перевіримо реєстрацію користувача. Сторінка авторизації користувача зображена на рисунку 3.6. Цифрами позначені основні елементи сторінки, які потрібно протестувати:

1 – поле для введення електронної пошти;

2 – поле для введення відповідного паролю;

3 – кнопка для входу у систему.

Рисунок 3.6 – Сторінка авторизації користувача у Vidfall

За перевірку коректності роботи даної сторінки відповідає тест-кейс testLoginLogout.java.

Для того, щоб вказати у файлі, що даний метод є тестовим його потрібно проанотувати – @Test, після чого даний метод можна буде запускати в окремому потоці для проведення тестування.

Анотації вдають із себе дескриптори, що включаються в текст програми, і використовуються для зберігання метаданих програмного коду, необхідних на різних етапах життєвого циклу програми. Інформація, що зберігається в анотаціях, може використовуватися відповідними обробниками для створення необхідних допоміжних файлів або для маркування класів, полів і т.д[43].

Структура даного тест-кейсу на Java виглядає наступним чином, як представлено на рисунку 3.7.

Рисунок 3.7 – Структура коду тестового файлу testLoginLogout.java

Анотація @Before вказує на те, що метод, що знаходиться безпосередньо після неї, буде виконуватися перед кожним тестованим методом @Test[43].

Оскільки Java підтримує спільне використання коду установки тестового оточення, то до того, як почне виконуватися метод тестування авторизації, буде викликаний шаблонний метод setUp ().

Отже, в блоці @Before пишемо шаблонний метод setUp (), де видаляємо усі куки (інформація, яка збережена в браузері, якщо користувач вже відвідував сайт):

```
@Before
   public void setUp()
   {  driver.manage().deleteAllCookies();
      driver.manage().timeouts().implicitlyWait(25, TimeUnit.SECONDS);    }
```

Анотація @Test (expected = Exception.class) – вказує на те, що в даному тестовому методі навмисно очікується Exception[43].

Відповідно в даному блоці пишемо основну функцію тесту, де спочатку налаштовуємо роботу із Jenkins:

```
        try {
           String OS = System.getProperty("os.name").toLowerCase();
           if(OS.indexOf("win") >= 0) {
              input = new FileInputStream("config.properties");
           } else{
              input = new FileInputStream("config-jenkins.properties");  }
           prop.load(input);
        } catch (IOException ioe) {
           ioe.printStackTrace(); }
```

Тепер можна писати самі команди емулювання користувача: перехід на головну сторінку проекту Vidfall, реєстрація, авторизація та вихід із системи.

Анотація @After вказує на те, що метод буде виконуватися після кожного тестованого методу @Test[43].

Як тільки метод тестування завершить свою роботу – викликаємо інший шаблонний метод – tearDown (), де програма просто виходить із системи. Причому його виклик не залежить від того чи успішно завершився тест чи ні.

Отже, блок @After виглядає наступним чином:

```
@After
   public void tearDown()
   {   driver.quit();  }
```

Сама функція перевірки авторизації користувача реалізована через його реєстрацію, а тоді логінацію із збереженими електронною адресою та паролем у попередній функції.

Функція реєстрації починається із натискання на кнопку «SIGN UP» (рисунок 3.6) інструментом click (), що емалює натискання клавішою мишки користувачем на будь-який вказаний елемент :

```
driver.findElement(By.cssSelector("a.sign_up")).click();
```

Тоді у відповідне поле вводимо (інструмент sendKeys () – вводить вказаний текст у відповідне поле) випадкову електронну адресу, що задаємо наступною функцією Random (), створює змінну із 25 знаків (прописні і рядкові букви англійського алфавіту і 10 цифр). У наступне поле вводимо пароль, що аналогічно задається функцією Random ()[44], і повторюємо це із наступним полем підтвердження паролю. В кінці натискаємо кнопку підтвердження:

```
driver.findElement(By.id("email")).sendKeys(email);
driver.findElement(By.id("password")).sendKeys(password);
driver.findElement(By.id("password_again")).sendKeys(password);
driver.findElement(By.cssSelector("a.btn-private-register")).click();
```

Інструмент cssSelector () вибирає на елемент на сторінці, в якого є відповідний клас, що вказується після знаку «.» (крапка). В даному випадку вибраний елемент «a», в якого клас – btn-private-register.

В результаті програма відкриває сторінку успішної реєстрації користувача (рисунок 3.8).

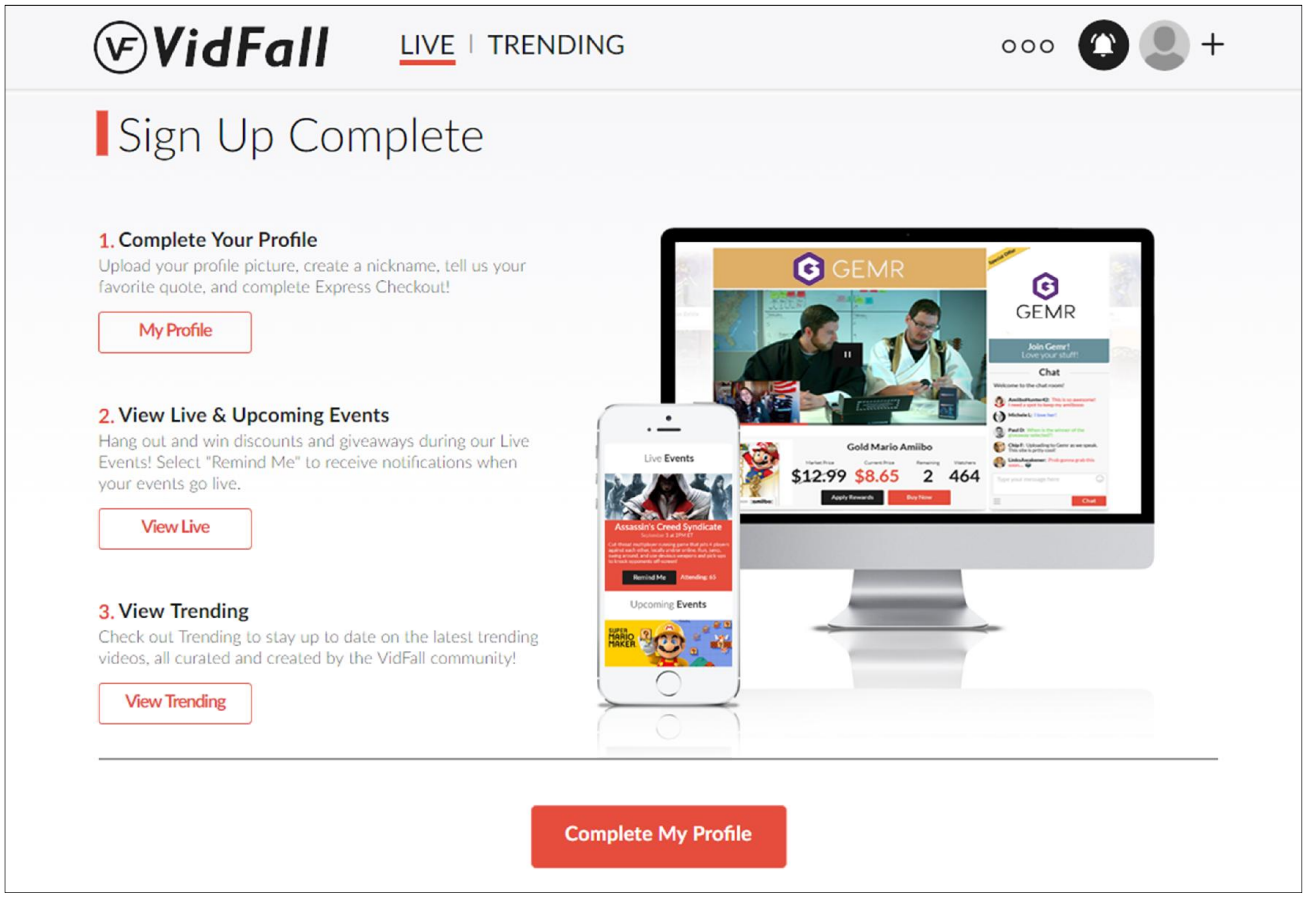

Рисунок 3.8 – Результат виконання функції реєстрації

Тоді програма виходить із аканту користувача обираючи даний пункт із випадаючого меню, що знаходиться у правому верхньому куті (рисунок 3.7). Шлях до пункту заданий це методом xpath ()[45], де в ньому прописане конкретне розташування елементів сторінки і їх властивостей:

WebElement subElement = driver.findElement(By.xpath("//a[contains (@href,'/main/logout/')]"));

Зареєструвавши конкретного користувача програма заходить на головну сторінку проекту, натискає на кнопку «SIGN IN» і заповнює форму збереженими електронною поштою та паролем:

driver.findElement(By.xpath("//div[@class='header-btns-v5']/a")).click();

driver.findElement(By.id("username")).sendKeys(email);

driver.findElement(By.xpath("(//input[@id='password'])[2]")).sendKeys(password);

driver.findElement(By.cssSelector("a.btn-private-login")).click();

Перевірка коректності роботи функції авторизації відбувається через перевірку наявності імені користувача на сторінці профайлу та порівняння на відповідність його із збереженим іменем (перша частина електронної пошти). Реалізовано через основну функцію тестування Assert.assertTrue () – чи твердження, що ім’я користувача за заданим шляхом відображається, є вірним (правдою)[46]:

Assert.assertTrue(driver.findElement(By.xpath("//input[contains(@value, '" + nickname + "')]")).isDisplayed());

В кінці програма закриває браузер. Результат проходження тесту представлений у консольному виводі (рисунок 3.9).

```
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:compileTestJava
:processTestResources UP-TO-DATE
:testClasses
:test

com.wv.vf.ui.testLoginLogout > testLoginAndLogout STARTED

com.wv.vf.ui.testLoginLogout > testLoginAndLogout PASSED
```

Рисунок 3.9 – Успішне виконання тесту testLoginLogout.java

3.3 Програмна реалізація модульного тестування веб-додатку

Кожен unit тест будемо писати до окремого файлу. Для прикладу, візьмемо файл User.php. В ньому є функція getProfilePercentage (), яка обраховує кількість відсотків заповнення профайлу користувача інформацією. Код функції, яку будемо тестувати:

```
public function getProfilePercentage($abolute = true) {
    $percentage = 0;
    if ($this->image) $percentage += 15;
```

```
if ($this->fname) $percentage += 5;
if ($this->lname) $percentage += 5;
if ($this->nickname) $percentage += 5;
if ($this->quote) $percentage += 5;
if (SettingsInterest::model()->findAllByAttributes(array('user_id' => $this->id)))
{  $percentage += 15;   }
if ($this->expressCheckoutIsAvailable()) $percentage += 25;
if ($this->email_confirmed) $percentage += 25;
if($abolute) {
   return $percentage;
} else {
   $returnPersentage = 0;
   $range = [25, 50, 75, 100];
   foreach ($range as $key=>$val) {
      if($key < 3 && $percentage >= $range[$key] && $percentage <
$range[$key+1]) {
         $returnPersentage = $range[$key];
      } elseif($percentage == 100) {
         $returnPersentage = 100;  } }
   return $returnPersentage; }  }
```

Функція в залежності від параметру повертає абсолютне (параметр = true) або відносне (параметр = false) значення. В першому випадку параметр переданий функції за умовчанням true, тобто в тесті ми нічого не передаємо. В другому випадку передаємо функції параметр false. Абсолютне значення обраховується сумуванням результатів перевірки існування всіх властивостей класу. Для відносного значення є наперед визначені: 0, 25, 50, 75, 100. З них повертається одне, до рівня якого досягло абсолютне значення.

На початку ми витягуємо із фікстур необхідного користувача, дані якого не заповнені. Він і буде екземпляром тестованого класу[47,48].

Першою є властивість «image», при існуванні якої змінна $percentage має збільшити своє значення на 15. Це реалізовано через функцію перевірки

assertEquals (очікуваний_результат, фактичний_результат) – чи відповідає значення першого аргументу (в даному випадку 15) другому (значення, яке повертає тестована функція):

```
$user->image = 'http://b.com/a.jpg';
$this->assertEquals(15, $user->getProfilePercentage());
$this->assertEquals(0, $user->getProfilePercentage(false));
```

В останньому рядку ми перевірили чому рівне відносне значення, яке обрахувала функція.

Наступною є перевірка властивості «fname», тоді значення має збільшитись на 5, тобто тепер абсолютний результат функції має бути 20, а відносний – 0:

```
$user->fname = 'fname';
$this->assertEquals(20, $user->getProfilePercentage());
$this->assertEquals(0, $user->getProfilePercentage(false));
```

Аналогічно властивості «fname» перевіряються наступні властивості класу.

При існуванні «lname», сума збільшується на 5, в підсумку дорівнює 25 і абсолютний, і відносний результати:

```
$user->lname = 'lname';
$this->assertEquals(25, $user->getProfilePercentage());
$this->assertEquals(25, $user->getProfilePercentage(false));
```

При існуванні «nickname», сума збільшується на 5, в підсумку абсолютний результат буде 30, а відносний – 25:

```
$user->nickname = 'nickname';
$this->assertEquals(30, $user->getProfilePercentage());
$this->assertEquals(25, $user->getProfilePercentage(false));
```

Якщо існує «quote», сума збільшується на 5, в підсумку абсолютний результат дорівнює 35, а відносний знову ж таки 25:

```
$user->quote = 'favorite_quote';
$this->assertEquals(35, $user->getProfilePercentage());
$this->assertEquals(25, $user->getProfilePercentage(false));
```

При встановлених інтересах «SettingsInterest», сума збільшується на 15, тому абсолютний і відносний результати будуть рівні 50:

```
$settingsInterest = new SettingsInterest();
$settingsInterest->user_id = 11;
$settingsInterest->settings_type_id = 1;
$settingsInterest->save();
$this->assertEquals(50, $user->getProfilePercentage());
$this->assertEquals(50, $user->getProfilePercentage(false));
```

Коли функція expressCheckoutIsAvailable() вертатиме значення true, сума збільшується на 25, абсолютний і відносний результати мають бути 75:

```
$address = new Address();
$address['fname'] = 'fname';
$address['lname'] = 'lname';
$address['address'] = 'address 25';
$address['city'] = 'city';
$address['country'] = 'USA';
$address['state'] = 'Texas';
$address['zip'] = '46000';
$address['home_phone'] = '46000234234';
$address['type'] = Address::TYPE_SHIPPING;
$address['user_id'] = 11;
$address->save();
$user->last4 = '0000';
$user->stripe_id = 'cus_3123424234';
$this->assertEquals(75, $user->getProfilePercentage());
$this->assertEquals(75, $user->getProfilePercentage(false));
```

При існуванні «email_confirmed», сума збільшується на 25, в підсумку і абсолютний результат, і відносний дорівнюватимуть 100:

```
$user->email_confirmed = true;
$this->assertEquals(100, $user->getProfilePercentage());
$this->assertEquals(100, $user->getProfilePercentage(false));
```

Результат виконання тесту UserTest.php представлений на рисунку 3.10.

```
PHPUnit 3.7.21 by Sebastian Bergmann.

Configuration read from C:\xampp\htdocs\wasabi\vidfall\www\protected\tests\phpunit.xml

....................................................

Time: 25.66 seconds, Memory: 10.75Mb

OK (52 tests, 206 assertions)
```

Рисунок 3.10 – Результат успішного виконання тесту UserTest.php

Для того, щоб перевірити наскільки вихідний код є відтестований переглянемо міру Code Coverage (покриття коду)[49]. На головній сторінці Jenkins перейдемо у проект та видеремо інструмент «Clover HTML Report».

На сторінці висвітлена статистика покриття коду всього проекту (рисунок 3.11). Окремо виділені методи, константи і загальні дані.

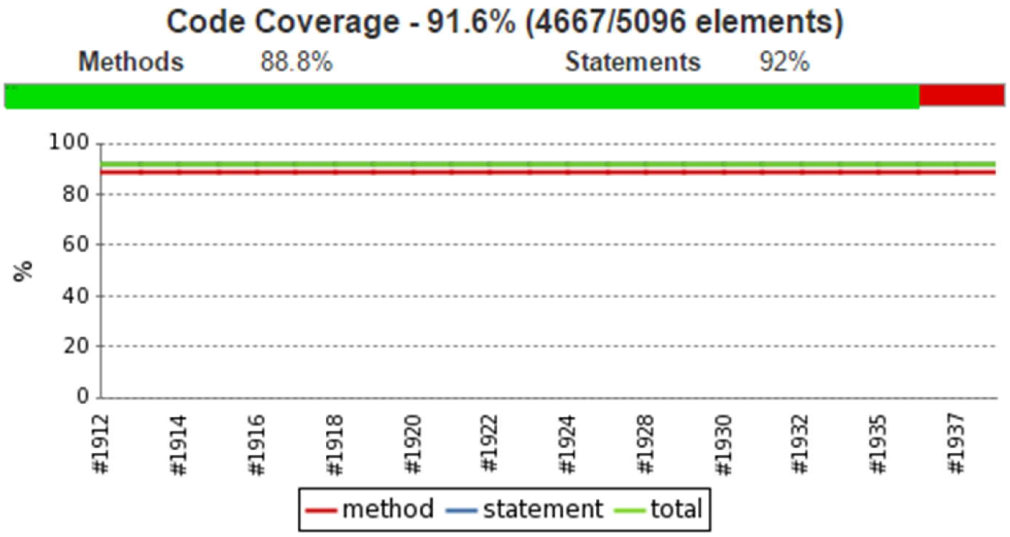

Рисунок 3.11 – Статистика покриття коду проекту Vidfall модульними тестами

Переглянемо докладну інформацію на рахунок кожного файлу. На рисунку 3.12 представлена таблиця з конкретними файлами, для яких написані модульні тести.

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		90.46%	1602 / 1771		90.39%	348 / 385		78.12%	50 / 64
Address.php		80.00%	20 / 25		80.00%	4 / 5		0.00%	0 / 1
Brand.php		100.00%	6 / 6		100.00%	4 / 4		100.00%	1 / 1
Config.php		100.00%	11 / 11		100.00%	4 / 4		100.00%	1 / 1
ContentVideo.php		100.00%	12 / 12		100.00%	7 / 7		100.00%	1 / 1
Error.php		100.00%	2 / 2		100.00%	2 / 2		100.00%	1 / 1
Inventory.php		98.10%	103 / 105		91.67%	11 / 12		0.00%	0 / 1
Notification.php		96.15%	25 / 26		90.00%	9 / 10		0.00%	0 / 1
Order.php		96.74%	89 / 92		85.71%	12 / 14		0.00%	0 / 1
Product.php		99.21%	126 / 127		95.83%	23 / 24		0.00%	0 / 1
Question.php		100.00%	36 / 36		100.00%	11 / 11		100.00%	1 / 1
SpecialOffer.php		100.00%	5 / 5		100.00%	3 / 3		100.00%	1 / 1
Statistics.php		100.00%	5 / 5		100.00%	3 / 3		100.00%	1 / 1
Subscribe.php		90.91%	20 / 22		75.00%	3 / 4		0.00%	0 / 1
User.php		86.35%	430 / 498		85.07%	57 / 67		0.00%	0 / 1
View.php		100.00%	2 / 2		100.00%	2 / 2		100.00%	1 / 1
AppleWatchForm.php		100.00%	4 / 4		100.00%	2 / 2		100.00%	1 / 1
AskOrderInfoForm.php		100.00%	7 / 7		100.00%	2 / 2		100.00%	1 / 1
ChangePassForm.php		100.00%	26 / 26		100.00%	3 / 3		100.00%	1 / 1
ContactForm.php		100.00%	8 / 8		100.00%	2 / 2		100.00%	1 / 1
ContestEntryForm.php		100.00%	6 / 6		100.00%	1 / 1		100.00%	1 / 1
CouponForm.php		100.00%	15 / 15		100.00%	3 / 3		100.00%	1 / 1
ExpressCheckoutForm.php		77.78%	14 / 18		66.67%	2 / 3		0.00%	0 / 1
InviteSocialForm.php		86.21%	50 / 58		66.67%	4 / 6		0.00%	0 / 1
LoginForm.php		75.86%	22 / 29		66.67%	2 / 3		0.00%	0 / 1
RecoverPassForm.php		100.00%	12 / 12		100.00%	3 / 3		100.00%	1 / 1
RequestDemoForm.php		100.00%	11 / 11		100.00%	2 / 2		100.00%	1 / 1
RewardCodeForm.php		100.00%	15 / 15		100.00%	2 / 2		100.00%	1 / 1
SavePassForm.php		100.00%	4 / 4		100.00%	1 / 1		100.00%	1 / 1
SaveSettingsForm.php		100.00%	55 / 55		100.00%	2 / 2		100.00%	1 / 1
SaveUserForm.php		100.00%	40 / 40		100.00%	4 / 4		100.00%	1 / 1
SendInviteForm.php		100.00%	3 / 3		100.00%	1 / 1		100.00%	1 / 1
UsePointsForm.php		100.00%	2 / 2		100.00%	1 / 1		100.00%	1 / 1

Рисунок 3.12 – Таблиця із файлами, для яких є тести

3.4 Програмна реалізація тестування прикладного програмного інтерфейсу веб-додатку

API тести будемо реалізовувати для кожної окремої функції. В API тестах реалізовується метод тестування «чорного ящика», тобто ми знаємо, що подається на вхід функції, і знаємо яким має бути результат. Базуючись на цьому принципі реалізуємо API тест для функції userChangePassword () – зміна паролю користувача[50].

На вхід функції поступають три параметри:

– auth_token – спеціальний унікальний код для API авторизації користувача, який відкриває доступ до його даних. За замовчуванням він вже переданий функції;

– password – дійсний пароль користувача;

– password_new – новий пароль користувача.

На виході ми отримуємо відповідь сервера у форматі JSON – текстовий формат обміну даними між комп'ютерами. В ньому міститься код стану HTTP – відповідь сервера при запиті по протоколу HTTP, змінна, що повідомляє про успішне чи неуспішне виконання функції, а в разі помилки – повідомлення з чим виникли проблеми[51]. Якщо сервер працює справно, то код завжди буде 200 (OK) – стандартна відповідь про успішне виконання HTTP запиту. Як і в модульних тестах використаємо ті ж самі фікстури.

Зробимо негативну перевірку – викличемо функцію не ввівши потрібні дані:

```
$response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '', 'password_new' => '']);
```

І перевіримо чи фактичний результат відповідає очікуваному :

```
$this->assertEquals($response['http_code'], '200');
$this->assertEquals($response['data']['success'], false);
$this->assertEquals($response['data']['errors']['general'][0], "EVF_required_params_were_not_passed");
```

Тепер попробуємо ввести не правильний пароль, в результаті має бути повідомлення про помилку і її тип:

```
$response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '000000', 'password_new' => '222222']);
this->assertEquals($response['http_code'], '200');
$this->assertEquals($response['data']['success'], false);
$this->assertEquals($response['data']['errors']['password'][0], "Incorrect password");
```

І в кінці виконаємо позитивне тестування– введемо всі правильні дані:

$response = $this->_doCall('/user/changePassword', 'test1', false, ['password' => '111111', 'password_new' => '222222']);

$this->assertEquals($response['http_code'], '200');

$this->assertEquals($response['data']['success'], true);

Виконання тесту і його результат представлені на рисунку 3.13.

```
PHPUnit 3.7.21 by Sebastian Bergmann.

Configuration read from C:\xampp\htdocs\wasabi\vidfall\www\protected\tests\phpunit.xml

..........

Time: 3.75 seconds, Memory: 7.00Mb

OK (10 tests, 13 assertions)
```

Рисунок 3.13 – Результат успішного виконання API тесту UserControllerTest.php

Отже, в даному розділі було розроблено алгоритм роботи проекту системи тестування веб-додатків. Написано програмний код з використанням різних технологій та мов програмування для функціонування системи. Здійснено реалізацію системи тестування в реальному проекті. Доведено надійність та продуктивність системи мультикритеріального тестування веб-додатків.

ВИСНОВКИ

1. Було розглянуто види веб-додатків, їх класифікацію та призначення. Розглянуто вразливі місця додатків відносно класифікації. Також технології їх створення та інструменти.

2. В ході дослідження було вияснено, що в часи розвитку інтернет технологій надійність веб-додатків є необхідністю. Було розглянуто основні методи тестування програмних засобів, для задоволення цієї необхідності.

3. Було розглянуто найпопулярніші на сьогоднішній день інструменти тестування веб-додатків відносно кожного методу, досліджено їх продуктивність на прикладі роботи з реальними даними в існуючій системі, заміряно їх швидкодію та характеристики, та побудовано графіки залежностей характеристик. Аналізуючи графіки, зроблено висновки про слабкі і сильні сторони методів і зроблений висновок про їх ефективність.

4. Було розглянуто перспективи і можливості застосування технологій безперервної інтеграції, як інструменту створення цілих проектів тестування.

5. Також були вибрані конкретні інструменти і сервіси, які необхідні для програмної системи тестування веб-додатків.

6. Розроблено алгоритм роботи проекту системи тестування веб-додатків, написано програмний код з використанням різних технологій та мов програмування для функціонування системи і здійснено реалізацію системи тестування в реальному проекті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Totallink. Терміни та поняття. [Електронний ресурс]: А. О. Береца. – 2012. – Режим доступу : http://totallink.cv.ua/%D1%82%D0%B5%D1%80%D0%BC%D1%96%D0%BD%D0%B8-%D1%82%D0%B0-%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D1%82%D1%8F/

2. 2Websecurity. Веб безпека. [Електронний ресурс]: Р. І. Ромашин. – 2015. – Режим доступу : http://websecurity.com.ua/security/chapter1/

3. 3Megabook. Web-приложения. [Електронний ресурс]: А. А. Орлов. – 2011. – Режим доступу : http://megabook.ru/article/Web-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F

4. 4AJAX: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/AJAX

5. 5JavaScript: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/JavaScript

6. 6DHTML: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/DHTML

7. XMLHttpRequest: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/XMLHttpRequest

8. Labaka – обучение программированию. Структура веб-приложения. [Електронний ресурс]: В. А. Солонин. – 2011. – Режим доступу : http://labaka.ru/likbez/struktura-veb-prilozheniya

9. IEEE Guide to Software Engineering Body of Knowledge (SWEBOK) / IEEE Computer Society. – IEEE Computer Society Press, 2004. – 204p.

10. ISO 8402:1994 Quality management and quality assurance. – International Organization for Standardization, 1994. – 39p.

11. Савин. Р. Тестирование Дот Ком, или Пособие по жесткому обращению багами в интернет-стартапах / Р. Савин. – Дело, 2007. – 312 с.

12. DOU. Тестирование. Фундаментальная теорія. [Електронний ресурс]: Gennadii Mishchevskii. – 2015. – Режим доступу : https://dou.ua/forums/topic/13389/

13. Тестування програмного забезпечення. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F

14. Керування конфігурацією. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BA%D0%BE%D0%BD%D1%84%D1%96%D0%B3%D1%83%D1%80%D0%B0%D1%86%D1%96%D1%94%D1%8E

15. Wiki. Сравнение систем управления конфигураций. [Електронний ресурс]: Д.Р. Туляков. – 2016. – Режим доступу : http://www.dtulyakov.ru/ci-cd-cm.html

16. ORDINATUS. 5 инструментов непрерывной интеграции. [Електронний ресурс]: А.Р Рубенов. – 2016. – Режим доступу : http://ordinatus.ru/5-instrumentov-nepreryvnoj-integracii/

17. Comparison of continuous integration software. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software

18. DevColibri. Тестирование с помощью TestNG в Java. [Електронний ресурс]: А.Р. Барчук. – 2013. – Режим доступу : http://devcolibri.com/1528

19. WebForMyself. Тестирование кода с PHPUnit. [Електронний ресурс]: О.В. Тотко. – 2014. – Режим доступу : https://webformyself.com/testirovanie-koda-s-phpunit/

20. Selenium / WebDriver. [Електронний ресурс] // Selenium – свободный ресурс для автоматизация веб-приложений через браузер. – Режим доступу : http://selenium2.ru/

21. Хабрахабр. Что такое Selenium? [Електронний ресурс]: А.В. Баранцев. – 2012. – Режим доступу : https://habrahabr.ru/post/152653/

22. PVS-Studio. TestComplete. [Електронний ресурс]: Р.В. Воронов. – 2014. – Режим доступу : http://www.viva64.com/ru/t/0089/

23. HP QuickTest Professional. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://ru.wikipedia.org/wiki/HP_QuickTest_Professional

24. List of Testing Tools. [Електронний ресурс] // guru99 – professional courses. – Режим доступу : http://www.guru99.com/list-of-testing-tools.html

25. Test Tools [Електронний ресурс] // Ministry Of Testing – co-creating smarter testers. – Режим доступу : http://www.ministryoftesting.com/resources/softwaretesting-tools/

26. 10 REGRESSION/FUNCTIONAL WEB TESTING TOOLS [Електронний ресурс] // TECH BLOG. – Режим доступу : http://www.hurricanesoftwares.com/10-regressionfunctional- web-testing-tools/

27. 5 Best Test Automation Tools [Електронний ресурс] // automated-360 blog . – Режим доступу : http://automated- 360.com/automation-tools/5-best-test-automation-tools

28. Category:Software testing tools [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://en.wikipedia.org/wiki/Category:Software_testing_tools

29. What is Selenium? [Електронний ресурс] // Selenium HQ. – Режим доступу: http://docs.seleniumhq.org/

30. HP Unified Functional Testing software [Електронний ресурс] // Web-site of Hewlett-Packard Company. – Режим доступу : http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA4- 8360ENW.pdf

31. Test Studio [Електронний ресурс] // Web-site of Telerik. – Режим доступу : http://www.telerik.com/teststudio

32. TestComplete Platform: Testing for Desktop, Mobile, & Web Applications [Електронний ресурс] // Web-site of Smartbear. – Режим доступу : http://smartbear.com/product/testcomplete/overview/

33. Automated Testing of Desktop. Web. Mobile. [Електронний ресурс] // Ranorex Company. – Режим доступу : http://www.ranorex.com/

34. Самойлов, В. Д. Модельное конструирование компьютерных приложений / В. Д. Самойлов. – К.: Наукова думка. – 2007. – 198 с.

35. Skybot Job Scheduler [Електронний ресурс] // HelpSystems issues. – Режим доступу : http://www.helpsystems.com/skybot/products/skybot-scheduler

36. Хабрахабр . Descriptive Programming в QuickTest Pro. [Електронний ресурс]: В.В. Ягилюк. – 2009. – Режим доступу : https://habrahabr.ru/post/69138/

37. HP QuickTest Professional. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://en.wikipedia.org/wiki/HP_QuickTest_Professional

38. Bugs Catcher. Telerik WebUI Test Studio. [Електронний ресурс]: В.Н. Лувин. – 2011. – Режим доступу : http://bugscatcher.net/archives/817

39. Хабрахабр. CodedUI или Ranorex? Автоматизация функционального тестирования .NET приложений. [Електронний ресурс]: В.В. Джигда. – 2013. – Режим доступу : https://habrahabr.ru/company/2gis/blog/188302/

40. API testing. [Електронний ресурс] // guru99 – professional courses. – Режим доступу : http://www.guru99.com/api-testing.html

41. Хабрахабр. Composer — менеджер зависимостей для PHP. [Електронний ресурс]: Р.И. Ружков. – 2012. – Режим доступу : https://habrahabr.ru/post/145946/

42. Программирование на языке JAVA. [Електронний ресурс]: А.В. Картузов. – 2013. – Режим доступу : http://www.mstu.edu.ru/study/materials/java/

43. DevColibri. Unit тестирование с JUnit. [Електронний ресурс]: А.Р. Шуляк. – 2013. – Режим доступу : http://devcolibri.com/864

44. Brusilovsky, P. Methods and techniques of adaptive hypermedia / P. Brusilovsky // User Modeling and User-Adapted Interaction. – 1996. – № 6 (2-3). – P. 87-129.

45. International Forum of Educational Technology & Society [Електронний ресурс]. – Режим доступу : http://ifets.ieee.org/.

46. International Journal of Artificial Intelligence in Education (IJAIED) [Електронний ресурс]. – Режим доступу : http://aied.inf.ed.ac.uk/.

47. Титенко, С. В. FreshKnowledge – система управління навчальним Веб-контентом на семантичному рівні / С. В. Титенко, О. О. Гагарін // VII международная конференция «Интеллектуальный анализ информации ИАИ-2007», Киев, 15-18 мая 2007г. : Сб. тр. / Ред. кол. : С. В. Сирота (гл.ред.) и др. – К.: Просвіта, 2007. – С. 342-352.

48. FreshKnowledge CMS – онтологічно-орієнтована система керування контентом, розроблена здобувачем [Електронний ресурс]. – Режим доступу : http://www.freshknowledge.net.

49. De Bra, P. Web-based educational hypermedia / P. De Bra // Book chapter in: Data Mining in E-Learning / [edited by C. Romero and S. Ventura]. – Universidad de Cordoba, Spain, WIT Press., 2006. – P. 3-17.– ISBN 1-84564-152-3.

50. Encyclopedia of Artificial Intelligence. 2nd ed. / [S. C. Shapiro editor.]. – New York: John Wiley & Sons, 1992. – Volume 1. – p. 434.

51. Діренко, І. С. Система управління вмістом веб-ресурсів на основі онтологічно-документного моделювання / І. С. Діренко, О. В. Олецький // Теоретичні та прикладні аспекти побудови програмних систем. Матеріали міжнародної конференції TAAPSD'2006. Київ, грудень 2006 р. – С.171-176.

52. Система мультикритеріального тестування веб-додатків / В.Р. Лабо, К.М. Березька // Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп’ютерні інформаційні технології». – Тернопіль: ТНЕУ. – 2016 – с. 95