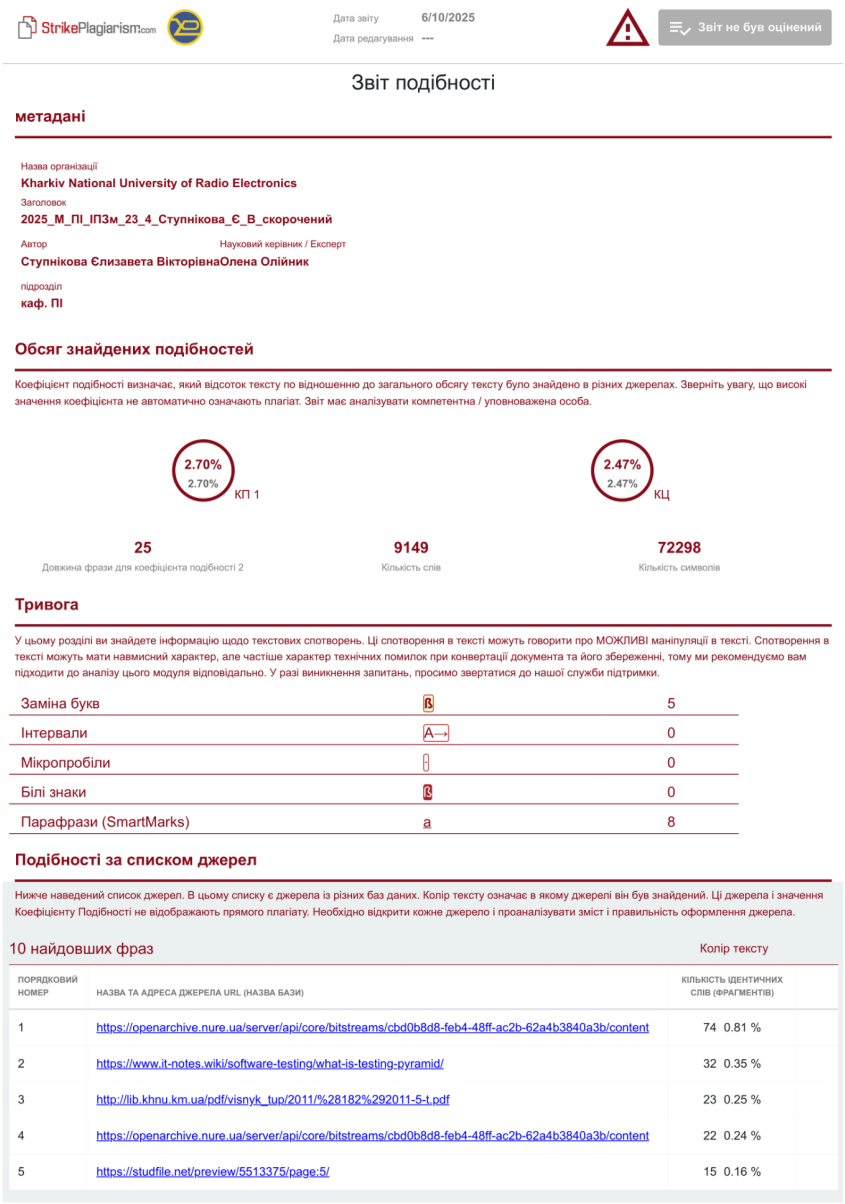
ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/10/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗм_23_4_Ступнікова_Є_В_скорочений
Автор
Ступнікова Єлизавета Вікторівна
Науковий керівник / Експерт
Олена Олійник
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

2.70% 2.70% КП 1

2.47% 2.47% КЦ

25 Довжина фрази для коефіцієнта подібності 2

9149 Кількість слів

72298 Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		5
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	8

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/server/api/core/bitstreams/cbd0b8d8-feb4-48ff-ac2b-62a4b3840a3b/content	74 0.81 %
2	https://www.it-notes.wiki/software-testing/what-is-testing-pyramid/	32 0.35 %
3	http://lib.khnu.km.ua/pdf/visnyk_tup/2011/%28182%292011-5-t.pdf	23 0.25 %
4	https://openarchive.nure.ua/server/api/core/bitstreams/cbd0b8d8-feb4-48ff-ac2b-62a4b3840a3b/content	22 0.24 %
5	https://studfile.net/preview/5513375/page:5/	15 0.16 %

6	https://openarchive.nure.ua/server/api/core/bitstreams/cbd0b8d8-feb4-48ff-ac2b-62a4b3840a3b/content	14 0.15 %
7	https://openarchive.nure.ua/server/api/core/bitstreams/cbd0b8d8-feb4-48ff-ac2b-62a4b3840a3b/content	13 0.14 %
8	https://openarchive.nure.ua/server/api/core/bitstreams/cbd0b8d8-feb4-48ff-ac2b-62a4b3840a3b/content	11 0.12 %
9	https://openarchive.nure.ua/server/api/core/bitstreams/cbd0b8d8-feb4-48ff-ac2b-62a4b3840a3b/content	9 0.10 %
10	https://www.it-notes.wiki/software-testing/what-is-testing-pyramid/	8 0.09 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (2.70 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://openarchive.nure.ua/server/api/core/bitstreams/cbd0b8d8-feb4-48ff-ac2b-62a4b3840a3b/content**	**156 (8) 1.71 %**
2	**https://www.it-notes.wiki/software-testing/what-is-testing-pyramid/**	**48 (3) 0.52 %**
3	http://lib.khnu.km.ua/pdf/visnyk_tup/2011/%28182%292011-5-t.pdf	28 (2) 0.31 %
4	https://studfile.net/preview/5513375/page:5/	15 (1) 0.16 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

ВСТУП

В розробці програмного забезпечення вирішальне значення має здатність постачати високоякісні продукти швидше за конкурентів. Автоматизація тестування може допомогти компаніям отримати конкурентну перевагу, однак правильний вибір інструментів чи платформи є критичним фактором успіху. Шукаючи найкращі рішення серед наявних, які критерії слід враховувати та які постачальники вважаються лідерами у сфері інструментів автоматизованого тестування?
Метою даної роботи є глибоке порівняння інструментів автоматизованого тестування веб-додатків та їх впливу на ефективність процесу тестування. В сучасному світі цифрових технологій веб-додатки стали невід'ємною частиною бізнес-процесів майже в кожній галузі, від фінансів та охорони здоров'я до роздрібної торгівлі та освіти. Це зумовлює необхідність забезпечення їх належної якості, надійності та продуктивності.
Актуальність питань ефективності веб-додатків обумовлена стрімким розвитком інтернет-технологій та зростаючими вимогами користувачів до швидкодії, зручності та функціональності. Сьогодні навіть незначні затримки у відображенні контенту чи помилки в роботі веб-додатків можуть призвести до втрати користувачів та, як наслідок, зниження конкурентоспроможності компанії. Автоматизоване тестування стає критично важливим компонентом у забезпеченні якості веб-додатків, дозволяючи своєчасно виявляти дефекти, підвищувати швидкість розробки та

ДОДАТОК Б

Презентаційні слайди для захисту кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

«Дослідження інструментів для автоматизованого тестування веб-додатків та їх впливу на ефективність процесу тестування»

Ступнікова Є.В. ІПЗм-23-4
Науковий керівник: проф. Лєсна Н.С.

20 червня 2025

1

Дослідження

Об'єкт дослідження - інструменти для автоматизованого тестування веб-додатків, їх вплив на процесні аспекти ефективності, зокрема повний вартісний цикл, стабільність тестового середовища, вплив на кількість нестабільних тестів, економічну ефективність всього проекту.

Методи:

- Порівняльний експеримент з однаковими E2E-сценаріями
- Запуск тестів у повторюваних циклах (10 разів кожен)
- Імітація хаотичних умов (мережеві затримки, HTTP 500)
- Chaos Engineering
- Фіксація флакі-тестів та помилок

2

Огляд літератури (аналогів)

Теорія надійності програмного забезпечення

Метрики якості (MTTR, Defect Density, Escaped Defects Rate)

Chaos Engineering (Netflix, Gremlin) — як метод підвищення стійкості систем

Виявлені прогалини:

– У більшості публікацій відсутні практичні метрики типу MTTR, флакі-індекс, нестабільність при збоях

– Недостатньо робіт, які порівнюють інструменти в умовах нестабільної інфраструктури або продакшн-аналогів

– Переважання уваги до швидкості виконання замість якості в реальному сценарії

Постановка задачі

Метою даної роботи є глибоке порівняння інструментів автоматизованого тестування веб-додатків та їх впливу на ефективність процесу тестування. В сучасному світі цифрових технологій веб-додатки стали невід'ємною частиною бізнес-процесів майже в кожній галузі, від фінансів та охорони здоров'я до роздрібної торгівлі та освіти. Це зумовлює необхідність забезпечення їх належної якості, надійності та продуктивності.

Визначення нового показника ефективності.

Методологія

Методи: Використано порівняльний аналіз, експериментальне тестування, Chaos Engineering, аналіз флакі-тестів, метрики ефективності (EDR, MTTR, TCO).

MTTR (Mean Time To Repair) - середній час, необхідний для відновлення роботи системи або усунення збою після його виникнення.

Інструменти:

– Selenium (Java + Cucumber),

– Cypress (JavaScript),

– Playwright (TypeScript),

– Katalon Studio (Groovy),

– TestComplete (Groovy/VBScript).

Архітектура система для проведення експериментального дослідження

Компоненти системи:

Цільовий веб-додаток

- Реальний публічний ресурс, що симулює e-commerce середовище

Тестові сценарії

- Ідентичні E2E-тести, реалізовані для 5 інструментів (Cypress, Playwright, Selenium, Katalon Studio, TestComplete)

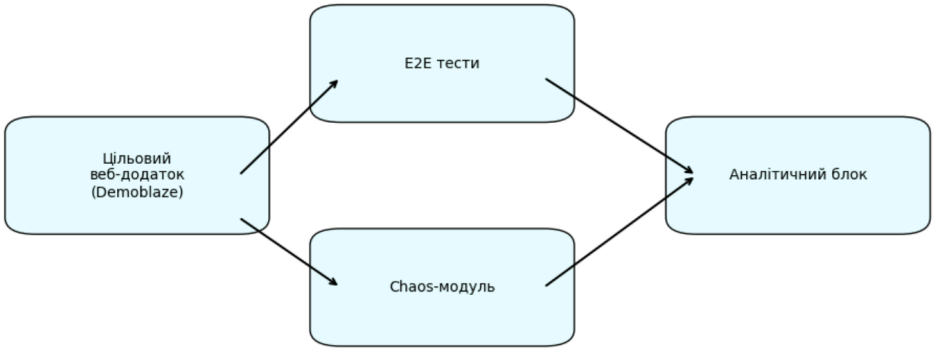
Chaos-модулі

- Додавання затримок, HTTP-помилок, нестабільності в мережу або інтерфейс для перевірки стійкості

Опис програмного забезпечення

Тести для кожного інструменту (Cypress, Playwright, Selenium, Katalon Studio, TestComplete).

– Інтеграція Chaos-параметрів: для деяких сценаріїв застосовано мережеву затримку, фальшиві відповіді (HTTP 500) тощо.



– Збір та логування метрик: результат кожного тесту фіксується — статус, час, помилки, флакі-поведінка.

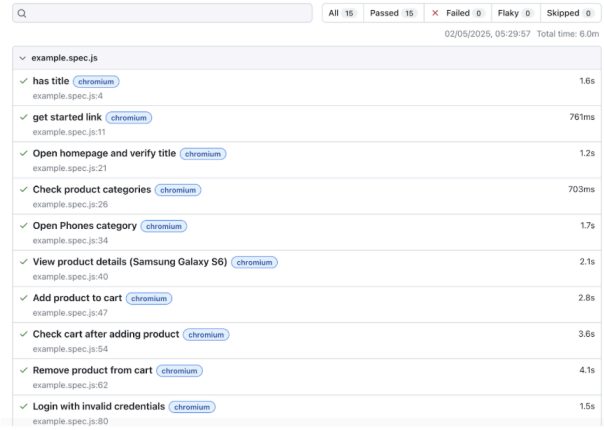
– Аналітичний модуль: розрахунок MTTR, Chaos Success Rate, побудова Парето-ефективного набору та оцінка за адитивною моделлю.

Зміст проведеного експерименту

Умови експерименту:

- 10 ітерацій запуску (150 виконань тестів)

- штучне введення аномалій (Chaos Engineering):



Запуск Playwright тестів

Аномалія	Реалізація	Інструменти
Затримки мережі (200–5000 мс)	Додавання штучних затримок.	Selenium WebDriverWait
HTTP-500 помилки (20% запитів)	Мокування API-відповідей за допомогою MockServer або WireMock.	MockServer
Скидання cookies	Видалення випадкових cookies під час тесту.	Selenium: driver.manage().deleteAllCookies()

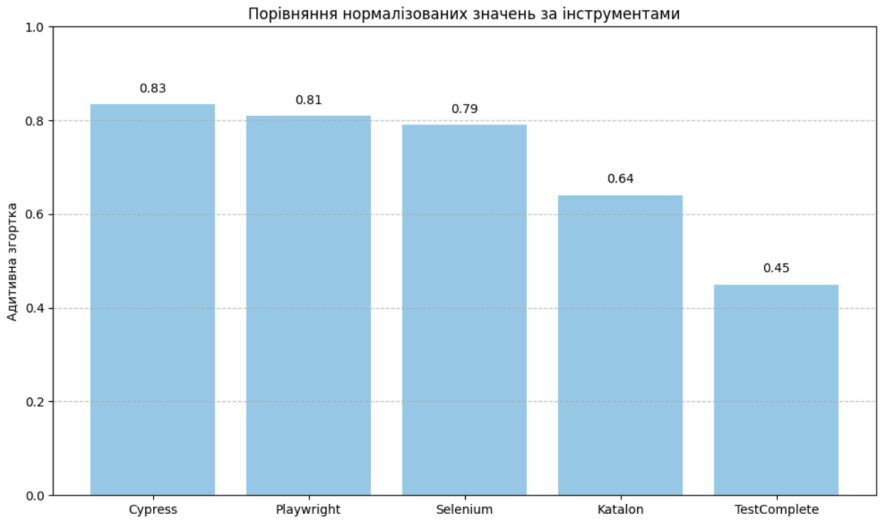
Опис аномалій

```
    Thread.sleep((long) (Math.random() * 5000) + 1000);

    if (Math.random() < 0.2) {
        MockServer.mockResponse("POST", "/login", 500, "Internal Server
Error");
    }

    if (Math.random() < 0.3) {
        driver.manage().deleteAllCookies();
    }
```

Приклад інтеграції аномалій

У рамках експериментального Chaos-тестування було реалізовано сценарії, спрямований на перевірку стабільності автоматизованих тестів при навмисному створенні аномальних умов. Метою було оцінка поведінки тестів за умов нестабільної мережі та випадкових збоїв на стороні сервера.

Результати експерименту



Графік візуалізації результатів адитивної згортки

Аналіз отриманих результатів

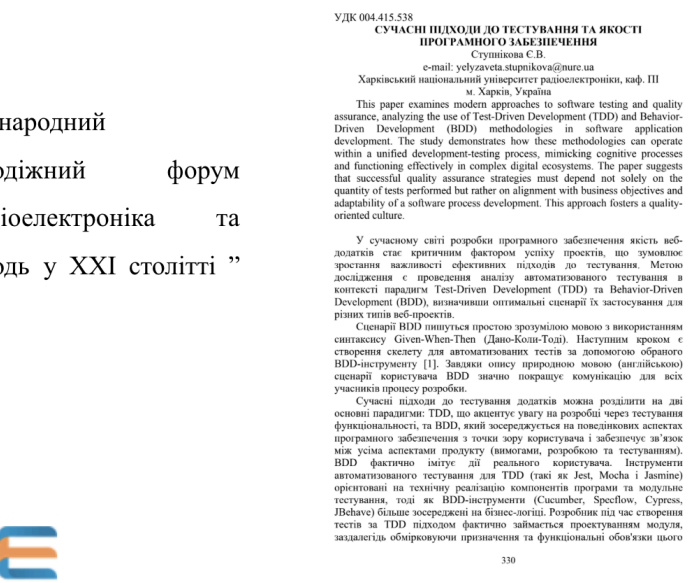
Критерій/ інструмент	Katalon Studio	Selenium	Cypress	TestComplete	Playwright
MTTR	32	32	29	34.7	31
Flaky Rate	13	13	13	13	13
Chaos Experiment Success Rate	2	3	2	2	2

Таблиця 4.3.1 – Характеристики інструментів автоматизації тестування

Критерій/ інструмент	Katalon Studio	Selenium	Cypress	TestComplete	Playwright
Сумісність з ОС	Кросплатформенний	Кросплатформенний	Windows, Linux, macOS	Windows	Кросплатформенний
Кросбраузерність	Chrome, Firefox, Safari, Edge	Кросбраузерний	Chrome, Edge	Chrome, Firefox, Edge	Chrome, Webkit, Firefox
Якість документації	Обширна документація	Обширна документація	Обширна документація	Комерційна, повна	Обширна документація
Економічна ефективність	Безкоштовний	Безкоштовний	Безкоштовний	Платний	Безкоштовний
Популярність у спільноті	340 зірок на GitHub	31.2 тис. зірок на GitHub	47.6 тис. зірок	-	73.3 тис. зірок
Мова скриптів	Groovy, Java	Java, C#, Perl, Python,	JavaScript	Vbscript, C, jscript, C++, delphi	JavaScript, TypeScript

Публікація результатів

Міжнародний молодіжний форум “Радіоелектроніка та молодь у XXI столітті ” 2025



УДК 004.415.538

СУЧАСНІ ПІДХОДИ ДО ТЕСТУВАННЯ ТА ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Ступнікова Є.В.
e-mail: yelyzaveta.stupnikova@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This paper examines modern approaches to software testing and quality assurance, analyzing the use of Test-Driven Development (TDD) and Behavior-Driven Development (BDD) methodologies in software application development. The study demonstrates how these methodologies can operate within a unified development-testing process, mimicking cognitive processes and functioning effectively in complex digital ecosystems. The paper suggests that successful quality assurance strategies must depend not solely on the quantity of tests performed but rather on alignment with business objectives and adaptability of a software process development. This approach fosters a quality-oriented culture.

У сучасному світі розробки програмного забезпечення якість веб-додатків стає критичним фактором успіху проектів, що зумовлює зростання важливості ефективних підходів до тестування. Метою дослідження є проведення аналізу автоматизованого тестування в контексті парадигм Test-Driven Development (TDD) та Behavior-Driven Development (BDD), визначивши оптимальні сценарії їх застосування для різних типів веб-проектів.

Сценарії BDD пишуться простою зрозумілою мовою з використанням синтаксису Given-When-Then (Дано-Коли-Тоді). Наступним кроком є створення скелету для автоматизованих тестів за допомогою обраного BDD-інструменту [1]. Завдяки опису природною мовою (англійською) сценарії користувача BDD значно покращує комунікацію для всіх учасників процесу розробки.

Сучасні підходи до тестування додатків можна розділити на дві основні парадигми: TDD, що акцентує увагу на розробці через тестування функціональності, та BDD, який зосереджується на поведінкових аспектах програмного забезпечення з точки зору користувача і забезпечує зв'язок між усіма аспектами продукту (вимогами, розробкою та тестуванням). BDD фактично імітує дії реального користувача. Інструменти автоматизованого тестування для TDD (такі як Jest, Mocha і Jasmine) орієнтовані на технічну реалізацію компонентів програми та модульне тестування, тоді як BDD-інструменти (Cucumber, SpecFlow, Cypress, JBehave) більше зосереджені на бізнес-логіці. Розробник під час створення тестів за TDD підходом фактично займається проектуванням модуля, заздалегідь обмірковуючи призначення та функціональні обов'язки цього

модуля. У такий спосіб він визначає архітектуру та відповідальність програмного компонента ще до початку його безпосередньої розробки. Оптимальна архітектура бази даних, у свою чергу, забезпечує легку підтримку та розвиток системи [2].

Проблема полягає в неправильному сприйнятті цих методологій як взаємовиключних альтернатив, а не як взаємодоповнюючих інструментів. Складні системи визначаються великою розмірністю, містять значну кількість елементів та зв'язків між ними [3]. Працівники які фокусуються виключно на TDD, часто досягають технічної досконалості коду, але ризикують створити продукт, що не відповідає реальним потребам користувачів. Натомість, прихильники чистого BDD можуть забезпечити відповідність бізнес-вимогам, але страждати від нижчої технічної якості.

З однієї сторони, тестування кінцевої поведінки користувача є надважливим. Але чи можливо ефективно перевірити поведінку системи без ґрунтовного тестування її функціональних компонентів? Тож BDD і TDD ні в якому разі не мають розглядатися як конкуруючі чи взаємовиключні, а навпаки, як взаємодоповнюючі підходи. BDD фокусує увагу на критичних бізнес-функціях, а TDD дозволяє розставити пріоритети на рівні компонентів, що разом скорочує час тестування. Унікальність цього симбіозу полягає в тому, що він дозволяє одночасно вирішувати два фундаментальні питання розробки: "Чи правильно ми будуємо систему?" та "Чи правильну систему ми будуємо?"

У результаті можна зробити висновок, що ефективне тестування веб-додатків потребує інтеграції підходів TDD та BDD, оскільки кожен з них фокусується на важливих аспектах розробки: технічній якості та бізнес-логіці. TDD забезпечує стабільність коду, тоді як BDD допомагає забезпечити відповідність продукту вимогам користувача і бізнесу.

Список використаних джерел:

1. Schults C. What are TDD and BDD and how to use them?!. The Pair Programming Blog. URL: https://blog.coscreen.co/blog/tdd-and-bdd/ (дата звернення: 02.03.2025).

2. Чуприна А. С. Дослідження моделей та архітектурних рішень в базах даних Web-технологій / А. С. Чуприна, А. С. Штельма // Поліграфічні, мультимедійні та web-технології : тези доп. IX Міжнар. наук.-техн. конф., 14-18 травня 2024 р. – Т. 1. – Харків: ТОВ «Друкарня Мадрид», 2024. – С. 193-194.

3. Пономаренко О. Програмна платформа для оцінювання ефективності агрегації структурної моделі складних систем / О. Пономаренко, В. Горбачов // Сучасний стан наукових досліджень та технологій в промисловості. – 2023. – № 3(25). – С. 79–87.

Підсумки

Проведено комплексне дослідження п’яти інструментів для автоматизованого тестування веб-додатків:

Cypress, Playwright, Selenium, Katalon Studio, TestComplete

- Розроблено однакові E2E-сценарії, які виконувались з хаотичними умовами та фіксацією результатів
- Запропоновано нові метрики оцінки MTTR, модифіковано Chaos Experiment Success Rate, Flaky Index.
- Застосовано багатокритеріальну модель оцінки з урахуванням технічних та економічних факторів
- Побудовано Парето-множину та здійснено адитивну згортку з ваговими коефіцієнтами

MTTR (Mean Time To Repair) — це не просто технічна метрика, а радше філософія виживання в умовах хаосу, помилок і продакшн-кризи.

Дякую за увагу!

ДОДАТОК В

Текст наукової публікації за темою кваліфікаційної роботи

УДК 004.415.538

СУЧАСНІ ПІДХОДИ ДО ТЕСТУВАННЯ ТА ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Ступнікова Є.В.
e-mail: yelyzaveta.stupnikova@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This paper examines modern approaches to software testing and quality assurance, analyzing the use of Test-Driven Development (TDD) and Behavior-Driven Development (BDD) methodologies in software application development. The study demonstrates how these methodologies can operate within a unified development-testing process, mimicking cognitive processes and functioning effectively in complex digital ecosystems. The paper suggests that successful quality assurance strategies must depend not solely on the quantity of tests performed but rather on alignment with business objectives and adaptability of a software process development. This approach fosters a quality-oriented culture.

У сучасному світі розробки програмного забезпечення якість веб-додатків стає критичним фактором успіху проектів, що зумовлює зростання важливості ефективних підходів до тестування. Метою дослідження є проведення аналізу автоматизованого тестування в контексті парадигм Test-Driven Development (TDD) та Behavior-Driven Development (BDD), визначивши оптимальні сценарії їх застосування для різних типів веб-проектів.

Сценарії BDD пишуться простою зрозумілою мовою з використанням синтаксису Given-When-Then (Дано-Коли-Тоді). Наступним кроком є створення скелету для автоматизованих тестів за допомогою обраного BDD-інструменту [1]. Завдяки опису природною мовою (англійською) сценарії користувача BDD значно покращує комунікацію для всіх учасників процесу розробки.

Сучасні підходи до тестування додатків можна розділити на дві основні парадигми: TDD, що акцентує увагу на розробці через тестування функціональності, та BDD, який зосереджується на поведінкових аспектах програмного забезпечення з точки зору користувача і забезпечує зв'язок між усіма аспектами продукту (вимогами, розробкою та тестуванням). BDD фактично імітує дії реального користувача. Інструменти автоматизованого тестування для TDD (такі як Jest, Mocha і Jasmine) орієнтовані на технічну реалізацію компонентів програми та модульне тестування, тоді як BDD-інструменти (Cucumber, Specflow, Cypress, JBehave) більше зосереджені на бізнес-логіці. Розробник під час створення тестів за TDD підходом фактично займається проектуванням модуля, заздалегідь обмірковуючи призначення та функціональні обов'язки цього

Рисунок В.1 – Міжнародний молодіжний форум "Радіоелектроніка та молодь у XXI столітті " 2025. Перша сторінка

модуля. У такий спосіб він визначає архітектуру та відповідальність програмного компонента ще до початку його безпосередньої розробки. Оптимальна архітектура бази даних, у свою чергу, забезпечує легку підтримку та розвиток системи [2].

Проблема полягає в неправильному сприйнятті цих методологій як взаємовиключних альтернатив, а не як взаємодоповнюючих інструментів. Складні системи визначаються великою розмірністю, містять значну кількість елементів та зв'язків між ними [3]. Працівники які фокусуються виключно на TDD, часто досягають технічної досконалості коду, але ризикують створити продукт, що не відповідає реальним потребам користувачів. Натомість, прихильники чистого BDD можуть забезпечити відповідність бізнес-вимогам, але страждати від нижчої технічної якості.

З однієї сторони, тестування кінцевої поведінки користувача є надважливим. Але чи можливо ефективно перевірити поведінку системи без ґрунтовного тестування її функціональних компонентів? Тож BDD і TDD ні в якому разі не мають розглядатися як конкуруючі чи взаємовиключні, а навпаки, як взаємодоповнюючі підходи. BDD фокусує увагу на критичних бізнес-функціях, а TDD дозволяє розставити пріоритети на рівні компонентів, що разом скорочує час тестування. Унікальність цього симбіозу полягає в тому, що він дозволяє одночасно вирішувати два фундаментальні питання розробки: "Чи правильно ми будуємо систему?" та "Чи правильну систему ми будуємо?"

У результаті можна зробити висновок, що ефективне тестування веб-додатків потребує інтеграції підходів TDD та BDD, оскільки кожен з них фокусується на важливих аспектах розробки: технічній якості та бізнес-логіці. TDD забезпечує стабільність коду, тоді як BDD допомагає забезпечити відповідність продукту вимогам користувача і бізнесу.

Список використаних джерел:

1. Schults C. What are TDD and BDD and how to use them?!. The Pair Programming Blog. URL: https://blog.coscreen.co/blog/tdd-and-bdd/ (дата звернення: 02.03.2025).

2. Чуприна А. С. Дослідження моделей та архітектурних рішень в базах даних Web-технологій / А. С. Чуприна, А. С. Штельма // Поліграфічні, мультимедійні та web-технології : тези доп. IX Міжнар. наук.-техн. конф., 14-18 травня 2024 р. – Т. 1. – Харків: ТОВ «Друкарня Мадрид», 2024. – С. 193-194.

3. Пономаренко О. Програмна платформа для оцінювання ефективності агрегації структурної моделі складних систем / О. Пономаренко, В. Горбачов // Сучасний стан наукових досліджень та технологій в промисловості. – 2023. – № 3(25). – С. 79–87.

Рисунок В.2 – Міжнародний молодіжний форум “Радіоелектроніка та молодь у XXI столітті ” 2025. Друга сторінка

ДОДАТОК Г

Експертний Висновок результатів Перевірки кваліфікаційної роботи на відповідність оформлення Вимог ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-4 (група)

Ступнікова Єлизавета
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
7.1.21	Заголовки підрозділів, пунктів і підпунктів звіту потрібно друкувати з абзацного відступу з великої літери без крапки в кінці.	13, далі за текстом
	7.3 Нумерація сторінок звіту	
	7.5 Рисунки	
	7.6 Таблиці	
7.6.8	Назву таблиці друкують з великої літери і розміщують над таблицею з абзацного відступу.	30, далі за текстом
7.6.9	Якщо рядки або колонки таблиці виходять за межі формату сторінки, таблицю поділяють на частини, розміщуючи одну частину під іншою або поруч, чи переносять частину таблиці на наступну сторінку. У кожній частині таблиці повторюють її головку та боковик. У разі поділу таблиці на частини дозволено її головку чи боковик заміняти відповідно номерами колонок або рядків, нумеруючи їх арабськими цифрами в першій частині таблиці. Слово «Таблиця » подають лише один раз над першою частиною таблиці. Над іншими частинами таблиці з абзацного відступу друкують «Продовження таблиці » або «Кінець таблиці ____ » без повторення її назви.	30
	7.7 Переліки	
7.7.4	Текст кожної позиції переліку треба починати з малої літери з абзацного відступу відносно попереднього рівня підпорядкованості.	31, далі за текстом
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
7.10.6	Пояснення познак, які входять до формули чи рівняння, треба подавати безпосередньо під формулою або рівнянням у тій послідовності, у якій їх наведено у формулі або рівнянні. Пояснення познак треба подавати без абзацного відступу з нового рядка, починаючи зі слова «де» без двокрапки. Познаки, яким встановлюють визначення чи пояснення, рекомендовано ви-рівнювати у вертикальному напрямку.	58, далі за текстом
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

Рисунок Г.1 – Висновок результатів Перевірки кваліфікаційної роботи. Перша сторінка

6.2.2	Якщо додатки є продовженням тексту основної частини звіту, нумерація сторінок додатків — це продовження нумерації сторінок звіту. Кожний додаток повинен мати заголовок, який друкують вгорі малими літерами з першої великої симетрично до тексту сторінки. Над заголовком, але посередині рядка, друкують слово «ДОДАТОК» і відповідну велику літеру української абетки, крім літер Ґ, Є, З, І, І, Й, О, Ч, Ь, яка позначає додаток. Текст кожного додатка починають з наступної сторінки.	66, далі за текстом

Експерт _____ Вадим НЕЧВОЛОД

(підпис) (прізвище, ініціали)

10.06.2025

Рисунок Г.2 – Висновок результатів Перевірки кваліфікаційної роботи. Друга сторінка