

Тестування в ігробудуванні

Лекція №5

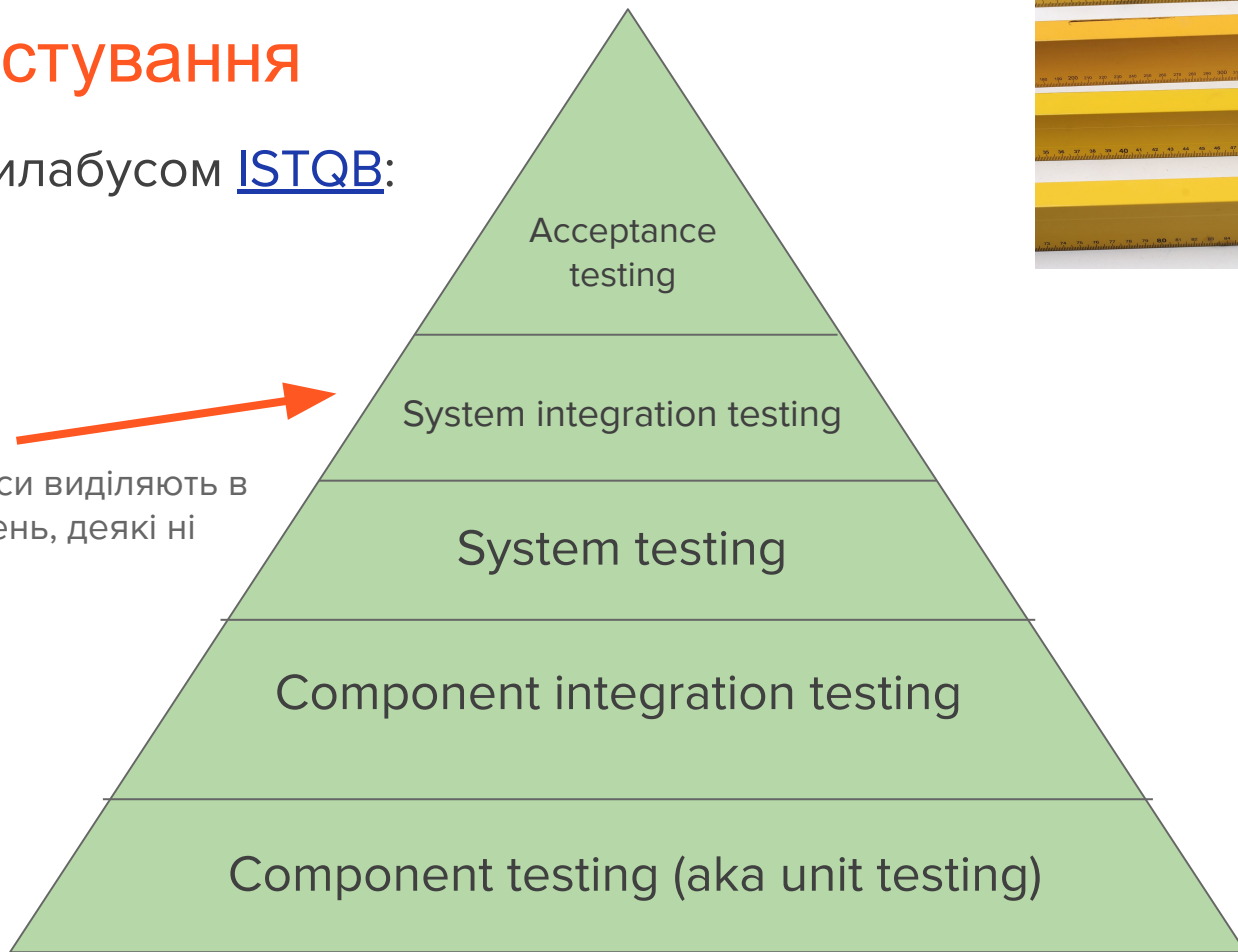
Тема: Види тестування та методика їхнього виконання

Питання лекції

1. Класифікація тестування.
2. Методика проведення різних видів тестування.
3. Автоматизація тестування. Інструменти для автоматизації тестування.

Рівні тестування

Згідно з си́лабусом [ISTQB](#):



*Деякі ресурси виділяють в окремий рівень, деякі ні



Класифікація тестування

Тестування класифікується за різними критеріями:

- За метою тестування
- За знанням будови системи
- За часом проведення тестування
- За ступенем підготовки до тестування
- За станом системи
- За ступенем автоматизації
- ...
- ...



Класифікація тестування



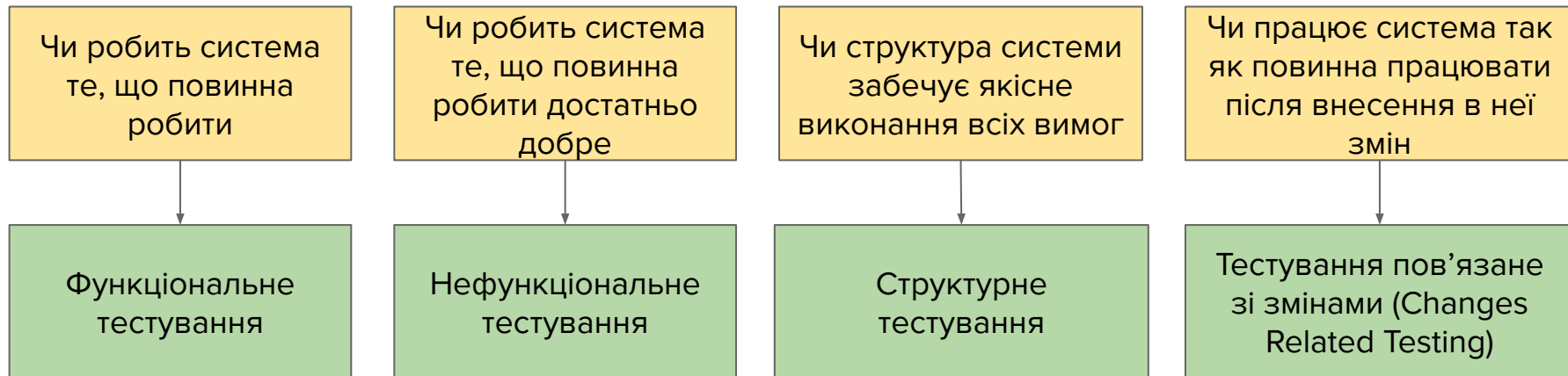
Спроба візуалізувати можливі види тестування

Класифікація тестування

За метою тестування.

Тут основне питання: Що саме ми хочемо перевірити?

Відповідями можуть бути наступні варіанти:



Класифікація тестування

За метою тестування.



Функціональне тестування передбачає аналіз функціональних характеристик додатка та перевірку на невідповідності між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і бізнес-вимоги. Функціональне тестування імітує фактичне використання системи.

Нефункціональне тестування направлено на перевірку тих аспектів ПЗ, які можуть бути описані в документації, але не відносяться до функцій програмних продуктів. включає тестування нефункціональних вимог системи, таких як продуктивність, безпека, масштабованість, зручність використання, надійність тощо.

Структурне тестування направлене на перевірку внутрішньої структури елементів системи тобто архітектури різноманітних різновидів програмного забезпечення. Його ще називають тестуванням “скляної” або “білої” скриньки.

Тестування, пов'язане зі змінами — це вид тестування програмного забезпечення, який перевіряє, як зміни впливають на існуючу функціональність, і включає такі методи, як регресійне тестування, повторне тестування (Retest, confirmation testing),

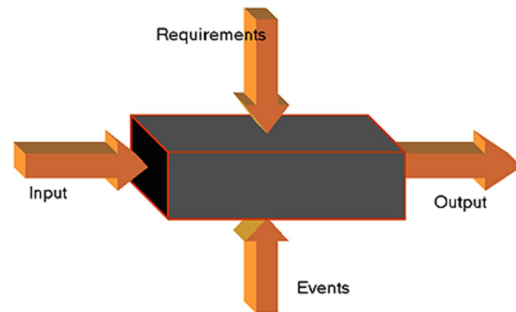
Класифікація тестування

Функціональне тестування

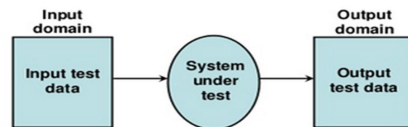
Базується на основі функціональних вимог (специфікації, інших видів вимог) і передбачає перевірку виконання програмою описаних вимог або розуміння можливих варіантів використання системи тестувальником.

Основні задачі функціонального тестування:

- Визначення ключових функцій / операцій системи, що знаходиться під тестуванням
- Визначення змінних / вхідних даних, що використовуються системою при її роботі, та визначення границь цих змінних
- Визначення змінних оточення / обладнання, що можуть вплинути на роботу системи, що знаходиться під тестування



Functional Testing



Класифікація тестування

Нефункціональне тестування

Термін нефункціональне тестування описує тести (перевірки), які необхідні для вимірювання характеристик системи і програмного забезпечення, що можуть бути визначені кількісно по тій чи іншій шкалі, наприклад, час відгуку для тестування продуктивності.

- **Тестування навантаження** – Load testing – як правило, проводиться з метою визначення поведінки ПЗ під очікуваним рівнем навантаження.
- **Тестування продуктивності** – Performance testing – перевірка швидкості роботи ПЗ або його окремих функцій.
- **Тестування сумісності** – Compatibility testing – тестування системи під час роботи в різних середовищах: «залізо», софт частина тощо.
- **Тестування безпеки** – Security testing – проводиться для відповіді на питання «Чи є додаток безпечним/захищеним чи ні?».
- І ще багато інших “-ity testinгів”

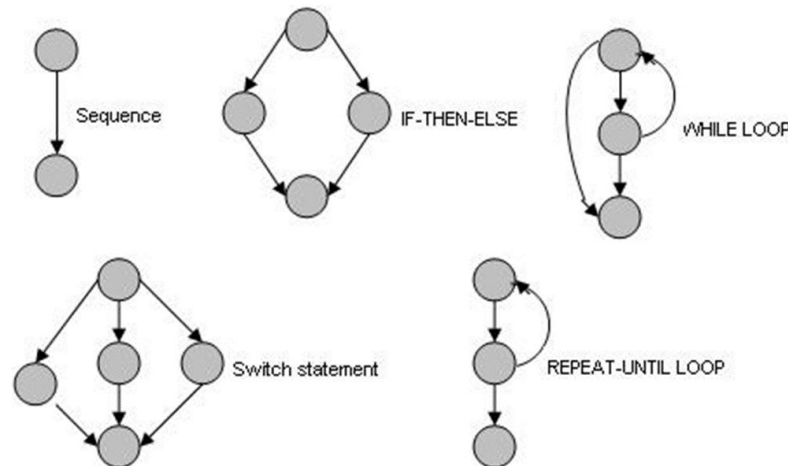
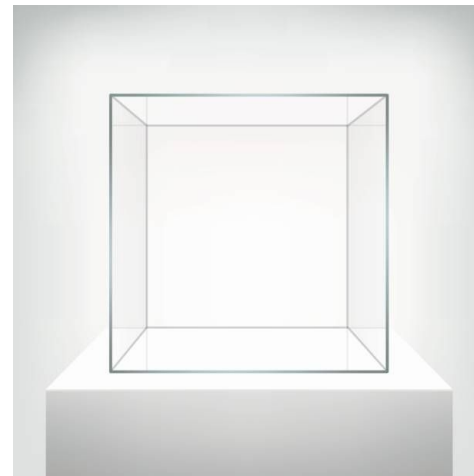
Класифікація тестування

Структурне тестування

Структурне тестування направлено на тестування структури системи або компонента. Цей вид тестування, як правило, відносять до тестування «білого» або «скляного», та «сірого» ящиків, оскільки ми перевіряємо, що відбувається всередині системи або додатка.

Методи структурного тестування:

- **Рядкове покриття (Statement Coverage)** – перевірка застосування усіх операторів в програмі на використання (хоча б один раз).
- **Покриття шляху (Path Coverage)** – метод тестування, призначений для задоволення критеріїв охоплення кожного логічного шляху через програму.
- **Покриття рішення (Branch Coverage)** – перевіряє, чи має кожна умова розгалуження для програми дійсні чи хибні значення;
- **Покриття умови (Condition Coverage)** – метод, схожий на Branch Coverage, основна відмінність полягає в перевірці стану покриття для умовних і неумовних гілок.



Класифікація тестування

Тестування пов'язане зі змінами

При тестуванні змін в системі дуже важливо зрозуміти різницю та межу між поняттями регресійне тестування (Regression testing) та повторне тестування (Retesting).

- Регресійне тестування (Regression testing) проводиться з метою перевірки працездатності функціоналу, що існує, та перевірки на відсутність сторонніх помилок після оновлення системи (внесення правок або доповнень в систему).
- Повторне тестування (Retesting) – проводиться для підтвердження виправлення помилки та роботи даного функціоналу.

Regression testing	Retesting
Регресійне тестування виконується тільки при додаванні нової функціональності ПЗ або істотній зміні функціоналу системи.	Ретест виконується в тому ж оточенні й з тими ж даними, але на новому білді.
Регрес можна проводити паралельно з повторним тестуванням (Retesting).	Повторне тестування має вищий пріоритет та має бути виконано до регресійного.
Тест-кейси можуть бути автоматизовані.	Тест-кейси не можуть бути автоматизовані.
В рамках регресійного тестування тест-кейси, які були відмічені раніше як «Passed», повинні бути перевірені повторно.	В рамках повторного тестування (ретест) перевіряються тест-кейси тільки зі статусом «Failed».

Класифікація тестування

Класифікація за іншими критеріями

За знанням будови системи:

- Black box
- White (Glass) box
- Gray box

За станом системи:

- Статичне
- Динамічне

За часом проведення тестування:

- Alpha
- Beta
- Acceptance

За ступенем автоматизації:

- ручне
- автоматизоване

За ступенем підготовленості до тестування:

- формальне (на основі документації)
- дослідницьке / інтуїтивне (Ad Hoc)

Методика проведення різних видів тестування.

Функціональне тестування.

Для проведення функціонального тестування доцільно використовувати техніки **тест-дизайну**.

Тест-дизайн – це один з початкових етапів процесу тестування ПЗ, на якому плануються і проєктуються тестові випадки (тест-кейси) відповідно до критеріїв якості, вимог до проєкту і цілей тестування. Головною метою тест-дизайну є покриття тестами всього функціоналу, використовуючи при цьому мінімальну кількість тестів.

Методика проведення різних видів тестування.

Функціональне тестування.

Еквівалентне розбиття (Equivalence Partitioning)

Визначення: поділ вхідних даних на класи, де всі значення в межах кожного класу поведуться однаково.

Алгоритм:

1. Визначити допустимі та недопустимі діапазони вводу.
2. Розділити їх на класи еквівалентності.
3. Вибрати по одному представнику з кожного класу.



Аналіз граничних значень (Boundary Value Analysis)

Визначення: тестування значень на межах допустимих діапазонів, де найчастіше виникають помилки.

Алгоритм:

1. Знайти мінімальні та максимальні межі вводу.
2. Вибрати тести для: мін-1, мін, мін+1, макс-1, макс, макс+1.

Методика проведення різних видів тестування.

Функціональне тестування.

Таблиця рішень (Decision Table Testing)

Визначення: техніка для тестування комбінацій умов та дій у системі.

Алгоритм:

1. Визначити всі умови та можливі дії.
2. Скласти таблицю з усіма комбінаціями.
3. Для кожної комбінації визначити очікуваний результат.
4. Створити тести на кожне правило.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
New customer (15%)	T	T	T	T	F	F	F	F
Loyalty card (10%)	T	T	F	F	T	T	F	F
Coupon (20%)	T	F	T	F	T	F	T	F
Actions								
Discount (%)	X	X	20	15	30	10	20	0

Методика проведення різних видів тестування.

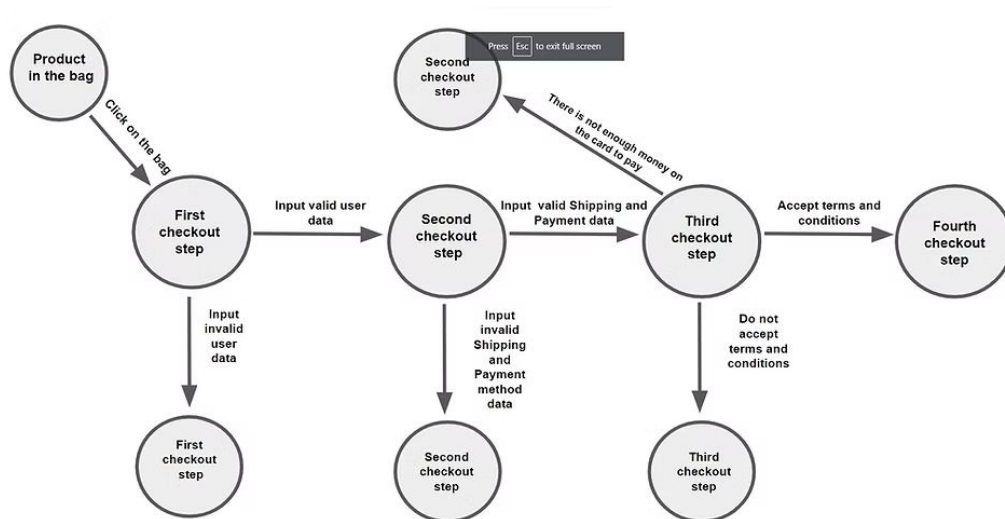
Функціональне тестування.

Тестування переходів станів (State Transition Testing)

Визначення: перевірка коректності роботи системи при переході між станами.

Алгоритм:

1. Побудувати діаграму станів системи.
2. Визначити всі переходи та тригери.
3. Створити тести для допустимих переходів.
4. Додати негативні тести для недопустимих переходів.



Методика проведення різних видів тестування.

Нефункціональне тестування. Тестування продуктивності

1. Визначення цілей тестування

Мета: наприклад, оцінити швидкодію, стабільність, масштабованість та ресурсоспоживання системи.

Формуються ключові метрики:

- Час відгуку (response time),
- Пропускна здатність (throughput),
- Рівень використання ресурсів (CPU, пам'ять, диск, мережа),
- Кількість одночасних користувачів.
- тощо

2. Планування тестування

- Визначаються сценарії використання.
- Задається рівень навантаження (кількість користувачів, частота запитів).
- Встановлюються критерії успішності (наприклад, час відповіді ≤ 2 секунди для 95% запитів).



Методика проведення різних видів тестування.

Нефункціональне тестування. Тестування продуктивності

3. Підготовка тестового середовища

- Тестове середовище має бути наближеним до системних вимог.
- Налаштовуються сервери, бази даних, мережеве обладнання.
- Налаштовуються системи моніторингу (Unity Profiler, Unreal Insights тощо).

4. Планування тестування

- Сценарії відображають реальні дії користувачів.
- Створюються скрипти для інструментів навантаження (Apache JMeter, Gatling, k6, Locust).
- Враховується як середнє, так і пікове навантаження.

5. Виконання тестування

Виконується пробний запуск для перевірки скриптів.

Проводяться основні види перевірок:

- Load Testing – при типовому навантаженні.
- Stress Testing – при навантаженні, що перевищує розрахункове.
- Endurance Testing – тривала робота системи при стабільному навантаженні.

Навантаження підвищується поступово (ramp-up).

Автоматизація тестування

Автоматизація тестування — це використання спеціальних інструментів і скриптів для автоматичного виконання тестів та перевірки результатів без ручного втручання.

Мета:

- Зменшення часу і вартості регресійного тестування.
- Підвищення повторюваності й точності тестів.
- Можливість швидкого запуску великої кількості тестів.

Види автоматизації тестування:

- Автоматизація тестування коду (Code-driven testing)
- Автоматизація тестування графічного інтерфейсу користувача (Graphical user interface testing)
- Автоматизація тестування API



Автоматизація тестування

Процес впровадження

1. Вибір кандидатів для автоматизації (стабільні, часто повторювані сценарії).
2. Вибір інструментів (залежно від типу застосунку: веб, мобільний, API).
3. Розробка тестових скриптів.
4. Інтеграція в CI/CD пайплайн.
5. Підтримка та оновлення автотестів.

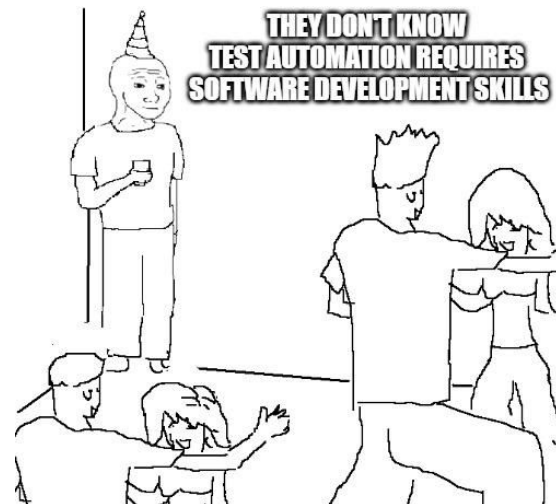
Популярні інструменти

UI: [Selenium](#), [Playwright](#), [Cypress](#).

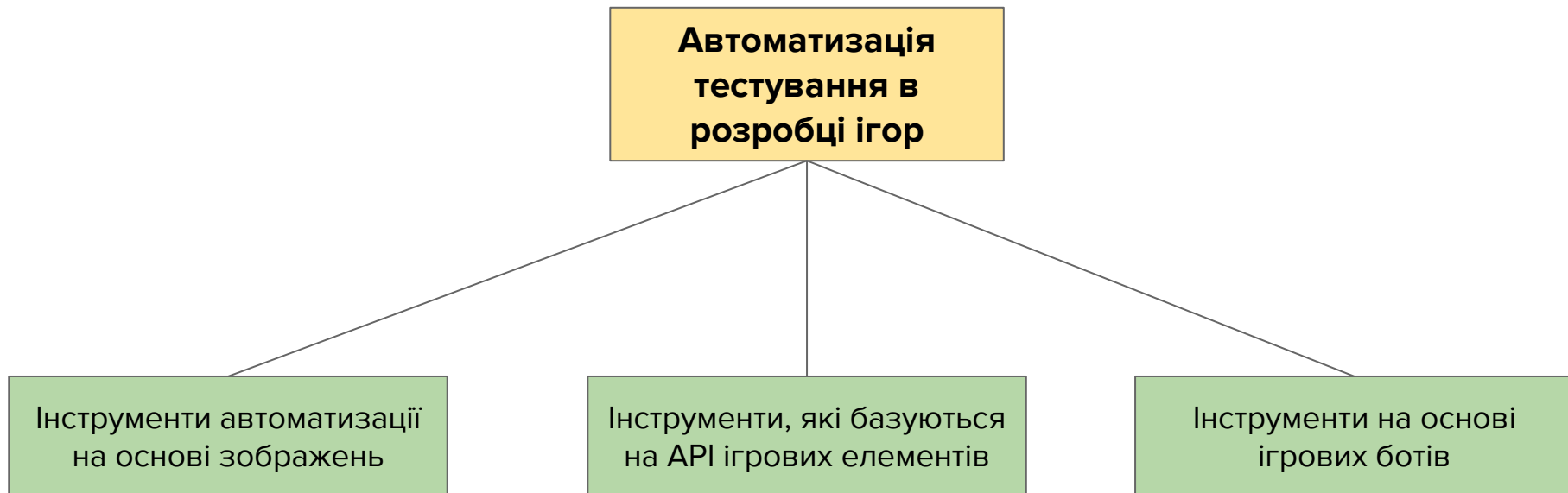
API: Postman, RestAssured.

Performance: JMeter, Gatling, k6.

CI/CD інтеграція: Jenkins, GitHub Actions, GitLab CI.



Автоматизація тестування в ігробудуванні



Автоматизація тестування в ігробудуванні

Інструменти автоматизації на основі зображень використовують методи розпізнавання зображень для взаємодії з елементами гри. Вони роблять знімки екрана, аналізують зображення та ідентифікують певні елементи на основі їхнього візуального вигляду.

Переваги:

Незалежність від інтерфейсу користувача

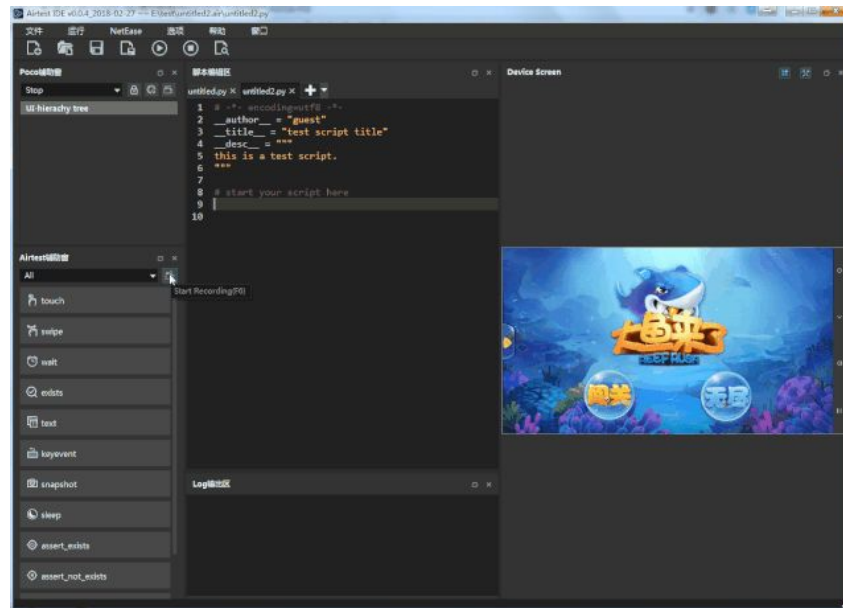
Візуальна гнучкість

Недоліки:

Крихкість

Вплив на продуктивність

Інструменти: Sikuli, Zap Test, [Airtest](#)



Автоматизація тестування в ігробудуванні

Інструменти, які базуються на відкритих API ігрових елементів встановлюють певний канал зв'язку між інструментом автоматизації та грою, використовуючи певні протоколи API. Вони обмінюються повідомленнями та командами для взаємодії з грою в режимі реального часу.

Переваги:

Взаємодія в реальному часі

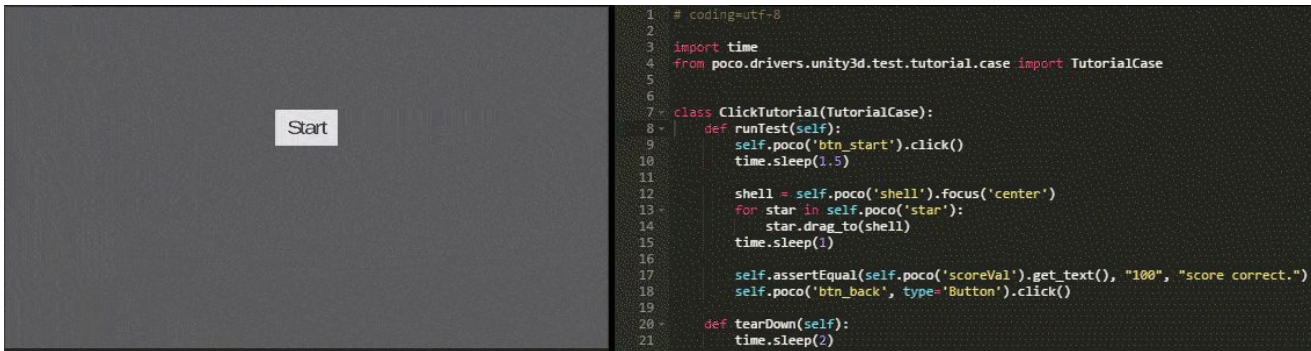
Точне керування

Недоліки:

Вимоги до інтеграції

Залежність від API гри:

Інструменти: [Game Driver](#), [Poco Driver](#), [AltTester](#)



Автоматизація тестування в ігробудуванні

Інструменти на основі ігрових ботів використовують різні методи, такі як алгоритми штучного інтелекту або системи на основі правил, для імітації поведінки гравців та автоматизації ігрових дій. Вони спрямовані на відтворення людської взаємодії в ігровому середовищі.

Переваги:

Гнучкість та адаптивність: Інструменти автоматизації ігрових ботів можуть бути розроблені для адаптації до різних ігрових сценаріїв та динамічного реагування на зміни в ігровому середовищі.

Можливості ШІ: Розширені інструменти ігрових ботів можуть включати алгоритми ШІ для навчання та покращення ігрових стратегій.

Недоліки:

Поріг входу: Розробка ефективних ігрових ботів часто вимагає досвіду в галузі ШІ або систем на основі правил, а також глибокого розуміння ігрової механіки.

Реалізація, специфічна для гри: Ігрових ботів може знадобитися адаптувати або налаштовувати для кожної гри, що робить їх менш універсальними для використання в різних іграх.

Інструменти: [Modl.ai](https://modl.ai)

Корисні ресурси

1. https://istqb.org/wp-content/uploads/2024/11/ISTQB_CTFL_Syllabus_v4.0.1.pdf
2. <https://coggle.it/diagram/We-qF3iayQABgwGL/t/testing>
3. <https://qagroup.com.ua/publications/vydy-testuvannya-ta-vidminnosti-mizh-nymy/>
4. <https://dou.ua/forums/topic/40666/>
5. <https://dou.ua/forums/topic/44882/>
6. <https://training.qatestlab.com/blog/technical-articles/review-the-types-of-testing/>
7. <https://www.ixiegaming.com/blog/comprehensive-review-game-test-automation-tools/>
8. <https://www.ixiegaming.com/blog/leveraging-pocodriver-for-game-test-automation/>

Дякую за увагу!