

Концидайло Андрій Михайлович

Quality Assurance Engineer
asp_kam1@student.ztu.edu.ua
@AndyFox96

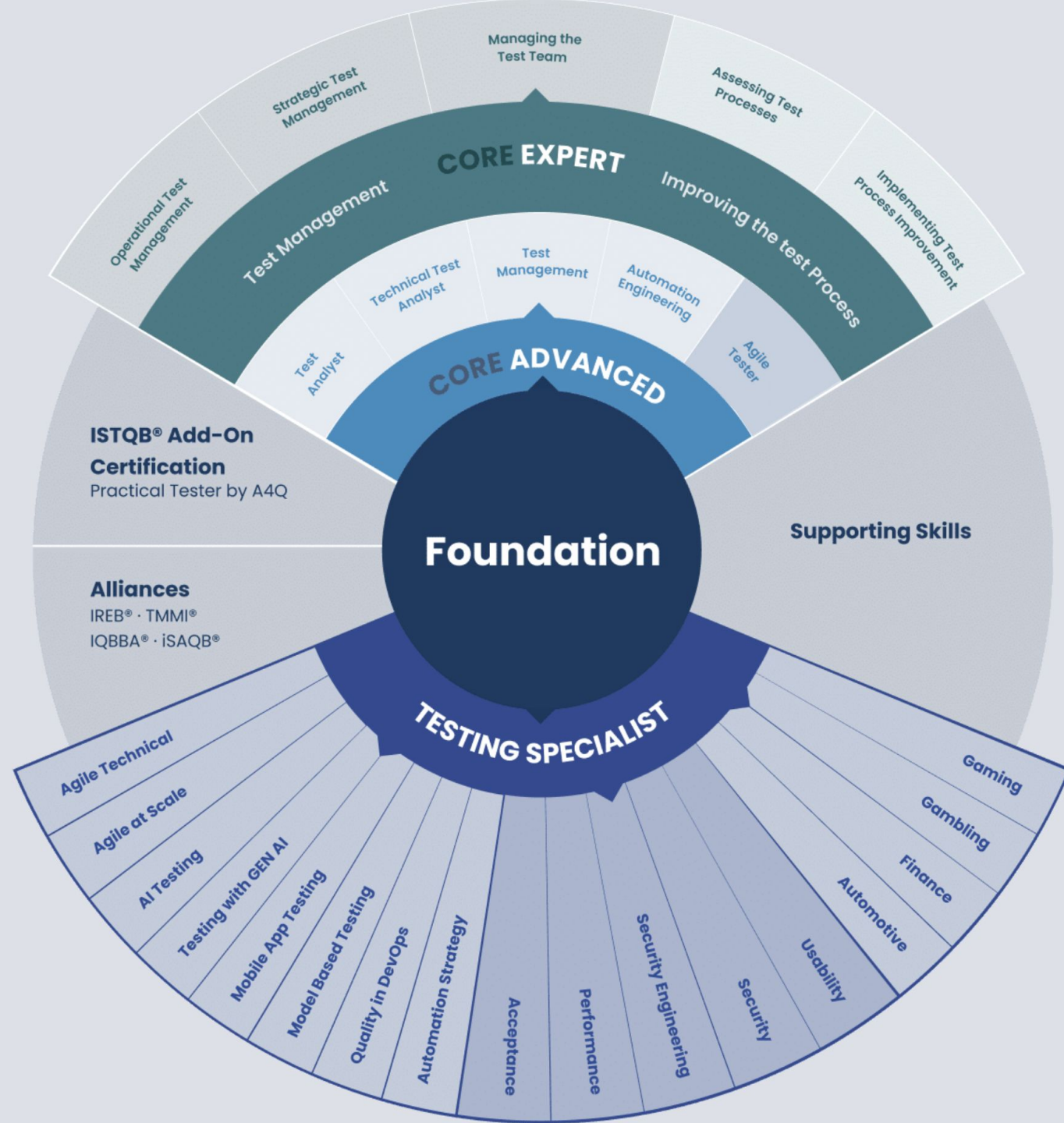


**Оцінювання
студента.**



Теми опрацювання на курсі

- Test Analyst
- Technical Test Analyst
- Test Management
- Test Automation Engineering
- Agile Tester



Розділ 1: Основи тестування

Що таке тестування, чому воно необхідне та ключові
принципи, які керують якістю ПЗ.

Питання до Розділу 1 (8 питань)

Теорія та визначення

- ❓ Які типові цілі тестування в залежності від контексту проєкту?
- ❓ У чому полягає ключова відмінність між тестуванням та відлагодженням (debugging)?
- ❓ Чому тестування є необхідним та як воно впливає на загальний успіх проєкту?
- ❓ Як співвідносяться поняття Quality Assurance (QA) та Quality Control (тестування)?

Принципи та процес

- ❓ Поясніть причинно-наслідковий зв'язок: помилка (error) → дефект (defect) → відмова (failure).
- ❓ Назвіть 7 принципів тестування. У чому полягає парадокс "пестициду"?
- ❓ Які основні активності входять до стандартного процесу тестування (від планування до завершення)?
- ❓ У чому полягають переваги та ризики високого рівня незалежності тестувальників?

Що таке тестування?

Суть тестування

Набір активностей для виявлення дефектів та оцінки якості робочих продуктів. Це не лише "запуск коду", а інтелектуальний процес, що включає статичне та динамічне тестування.

Верифікація vs Валідація

- **Верифікація:** Чи відповідає система заданим вимогам? (Чи правильно ми робимо продукт?)
- **Валідація:** Чи задовольняє система реальні потреби користувачів? (Чи правильний продукт ми робимо?)

Типові цілі тестування



Оцінка та Виявлення

Оцінка робочих продуктів (вимог, дизайну, коду). Провокування відмов для знаходження прихованих дефектів.



Зниження ризиків

Забезпечення необхідного покриття тестами та зменшення рівня ризику неналежної якості програмного забезпечення.



Довіра та Відповідність

Перевірка відповідності контрактним та правовим нормам. Побудова довіри стейкхолдерів до системи.

Тестування vs Відлагодження (Debugging)

Це два абсолютно різні процеси в життєвому циклі розробки.

- **Тестування:** Процес, що викликає відмови системи (через динамічне тестування) або безпосередньо знаходить дефекти (статичне тестування).
- **Відлагодження:** Процес, який виконують розробники. Включає відтворення відмови, діагностику (пошук дефекту) та його виправлення.
- **Підтверджувальне тестування:** Проводиться тестувальником після відлагодження, щоб переконатися, що дефект дійсно виправлено.



Помилка, Дефект та Відмова



Помилка (Error / Mistake)



Дефект (Defect / Bug)



Відмова (Failure)

Забезпечення якості (QA) vs Тестування

Quality Assurance (QA)

Процесно-орієнтований, **превентивний** підхід.
Фокусується на впровадженні та покращенні процесів.

- Спрямоване на запобігання дефектам.
- Забезпечує дотримання правильних процесів.
- Відповідальність всієї команди проекту.

Тестування (як частина QC)

Продукто-орієнтований, **корективний** підхід.
Фокусується на виявленні дефектів у готовому продукті.

- Спрямоване на виявлення дефектів та відмов.
- Перевіряє якість конкретного тест-об'єкта.
- Результати тестування використовуються QA для покращення процесів.

7 Принципів тестування (1-4)



1. Тестування показує наявність дефектів

Тестування може довести наявність дефектів, але не їх відсутність.



2. Вичерпне тестування неможливе

Протестувати "все" неможливо.
Використовуйте ризики та пріоритети для фокусування.



3. Раннє тестування економить час та гроші

Shift Left: знаходження багів на ранніх етапах запобігає їх накопиченню.



4. Дефекти скупчуються разом Принцип

Парето: невелика кількість модулів містить більшість критичних дефектів.

7 Принципів тестування (5-7)



5. Тести зношуються (Парадокс пестициду)

Повторення тих самих тестів перестає знаходити нові баги. Тести потрібно оновлювати.



6. Тестування залежить від контексту

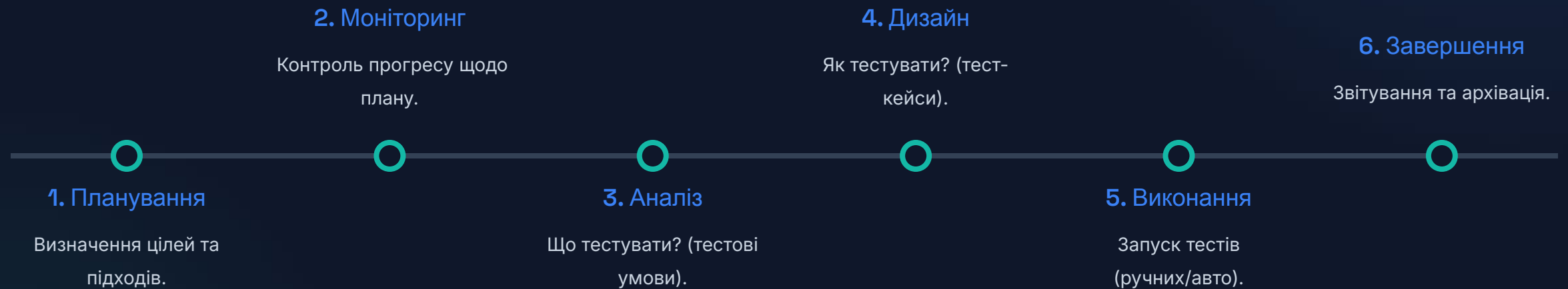
Тестування банківського ПЗ та e-commerce вимагають різних підходів та пріоритетів.



7. Омана щодо відсутності дефектів

Виправлення всіх багів не гарантує успіх, якщо система не відповідає бізнес-потреbam.

Стандартний процес тестування



Ролі в тестуванні

Test Management

Бере на себе відповідальність за процес тестування в цілому.

- ▷ Планування тестування.
- ▷ Моніторинг та контроль.
- ▷ Завершення тестування та звітування.
- ▷ В Agile ці завдання може виконувати вся команда.

Testing

Фокусується на інженерних (технічних) аспектах тестування.

- ▷ Аналіз бази тестування.
- ▷ Дизайн тест-кейсів та процедур.
- ▷ Реалізація та налаштування середовища.
- ▷ Виконання тестів та репорт дефектів.

Незалежність тестування

4

Рівні незалежності

Від відсутності до повної незалежності

Певна ступінь незалежності робить тестувальника ефективнішим завдяки відсутності когнітивних упереджень автора коду. Рівні варіюються від тестування самим розробником (немає незалежності) до залучення зовнішніх аудиторів або сторонніх компаній (дуже висока незалежність).

Недоліки незалежності: Може викликати ізоляцію, проблеми в комунікації та зняття з розробників відповідальності за якість.



Розділ 2: Тестування протягом SDLC

Інтеграція тестування в життєвий цикл розробки, підходи
Test-First та DevOps.

Питання до Розділу 2 (8 питань)

SDLC та Підходи

- ❓ Як обрана модель життєвого циклу розробки (SDLC) впливає на процес тестування?
- ❓ Які хороші практики тестування залишаються незмінними для будь-якої моделі SDLC?
- ❓ У чому суть підходів розробки Test-First (TDD, ATDD, BDD)?
- ❓ Який вплив має впровадження методології DevOps на тестування та автоматизацію?

Рівні та Типи

- ❓ Що означає концепція "Зсув вліво" (Shift-Left) та як її реалізувати на практиці?
- ❓ Опишіть різницю між компонентним, інтеграційним, системним та приймальним рівнями.
- ❓ Чим функціональне тестування відрізняється від нефункціонального (з наведенням прикладів)?
- ❓ У яких випадках застосовується підтверджувальне (Confirmation), а в яких — регресійне (Regression) тестування?

Вплив SDLC на тестування

Секвенційні моделі (Waterfall, V-model)

Тестувальники залучаються до огляду вимог на ранніх етапах. Динамічне тестування відбувається лише на пізніх стадіях, коли є готовий код.

Ітеративні/Agile моделі

Часті поставки вимагають швидкого зворотного зв'язку. Статичне та динамічне тестування виконується на всіх рівнях у кожній ітерації. Великий фокус на автоматизації.

Хороші практики тестування в будь-якій SDLC

✓ Кожній активності з розробки відповідає активність з тестування.

 Різні рівні тестування мають конкретні, різні цілі для уникнення дублювання.

 Аналіз та дизайн тестів починаються паралельно з відповідною фазою розробки (принцип раннього тестування).

 Тестувальники залучаються до огляду робочих продуктів щойно з'являються їх перші чернетки (підтримка Shift Left).

Підходи Test-First



TDD (Test-Driven Development)

Направляє написання коду через тести. Спершу пишуться тести, потім код, який задовольняє ці тести, після чого відбувається рефакторинг.



ATDD (Acceptance Test-Driven)

Тести створюються на основі критеріїв приймання ще до початку розробки функціоналу (спільна робота команди).



BDD (Behavior-Driven)

Описує бажану поведінку простою мовою (Given/When/Then), зрозумілою для бізнесу, що автоматично перетворюється на тести.

DevOps та Тестування

DevOps об'єднує розробку та експлуатацію. Сприяє автономії, швидкому фідбеку та автоматизації (CI/CD).

- Швидкий зворотний зв'язок щодо якості коду.
- CI підтримує "зсув вліво" (Shift Left) через обов'язкові авто-тести перед злиттям коду.
- Мінімізація ризиків регресії завдяки великому масштабу автоматизації.
- **Ризики:** налаштування CI/CD пайплайн вимагає часу, знань та інфраструктури. Ручне дослідницьке тестування все ще необхідне.

Applying **CI/CD DevOps**
in Legacy Modernization Projects



Концепція Shift Left (Зсув вліво)



Зсув вліво означає, що тестування повинно виконуватись раніше в життєвому циклі. Це не означає, що пізніми етапами тестування можна знехтувати.



— ISTQB CTFL v4.0.1 (Принцип Раннього Тестування)

Рівні тестування



Компонентне (Unit)

Тестування ізольованих модулів або класів. Зазвичай виконується розробниками за допомогою unit-фреймворків.



Інтеграційне

Тестування інтерфейсів. Буває інтеграція компонентів (між модулями) та інтеграція систем (із зовнішніми сервісами/API).



Системне & Приймальне

Системне: перевірка всієї системи (наскрізні сценарії). Приймальне: валідація, UAT, перевірка готовності для бізнесу.

Види тестування

Функціональне тестування

Оцінює функції, які компонент або система повинні виконувати. Це тестування **"ЩО"** повинна робити система.

- ▷ Функціональна повнота.
- ▷ Функціональна коректність.
- ▷ Функціональна доречність.

Нефункціональне тестування

Оцінює атрибути якості. Це тестування того, **"ЯК ДОБРЕ"** поводить ся система.

- ▷ Продуктивність (Performance).
- ▷ Зручність використання (Usability).
- ▷ Безпека (Security).
- ▷ Надійність та Портативність.

Тестування змін

Підтверджувальне (Confirmation)

Або re-testing. Проводиться, щоб підтвердити, що оригінальний дефект був успішно виправлений розробником.

- Запуск тест-кейсів, які раніше впали.
- Додавання нових тестів для покриття змін у коді.

Регресійне (Regression)

Підтверджує, що зміни в коді (фікси або нові фічі) не спричинили появу нових дефектів у незмінюваних частинах системи.

- Часто виконується повторно.
- Ідеальний кандидат для автоматизації (CI/CD).

Тестування супроводу (Maintenance)



Модифікації: Заплановані покращення, релізи, виправлення критичних багів (hot fixes). Потребує аналізу впливу (Impact Analysis) для виявлення ризиків регресії.



Оновлення або Міграція: Перехід на нову платформу, операційну систему або зміна бази даних. Включає тестування конвертації даних.



Виведення з експлуатації (Retirement): Тестування архівування даних, особливо якщо існують жорсткі вимоги до тривалого зберігання та можливого відновлення.

Розділ 3: Статичне тестування

Аналіз коду та документації без запуску програми, процес
рецензування.

Питання до Розділу 3 (7 питань)

Цінність та Відмінності

- ❓ Які типи робочих продуктів (work products) можуть бути перевірені за допомогою статичного тестування?
- ❓ У чому полягає основна цінність та економічна вигода статичного тестування?
- ❓ Які типи дефектів можна знайти за допомогою статичного тестування, але дуже важко виявити динамічним?
- ❓ Чому ранній та частий зворотний зв'язок від стейкхолдерів є критично важливим для якості?

Процес Рецензування

- ❓ З яких послідовних активностей складається формальний процес рецензування (Review Process)?
- ❓ Які ролі (Author, Moderator, Reviewer, Scribe, Manager) беруть участь у рецензуванні та які їхні обов'язки?
- ❓ Чим відрізняються неформальне рецензування, walkthrough, технічне рецензування та інспекція (Inspection)?

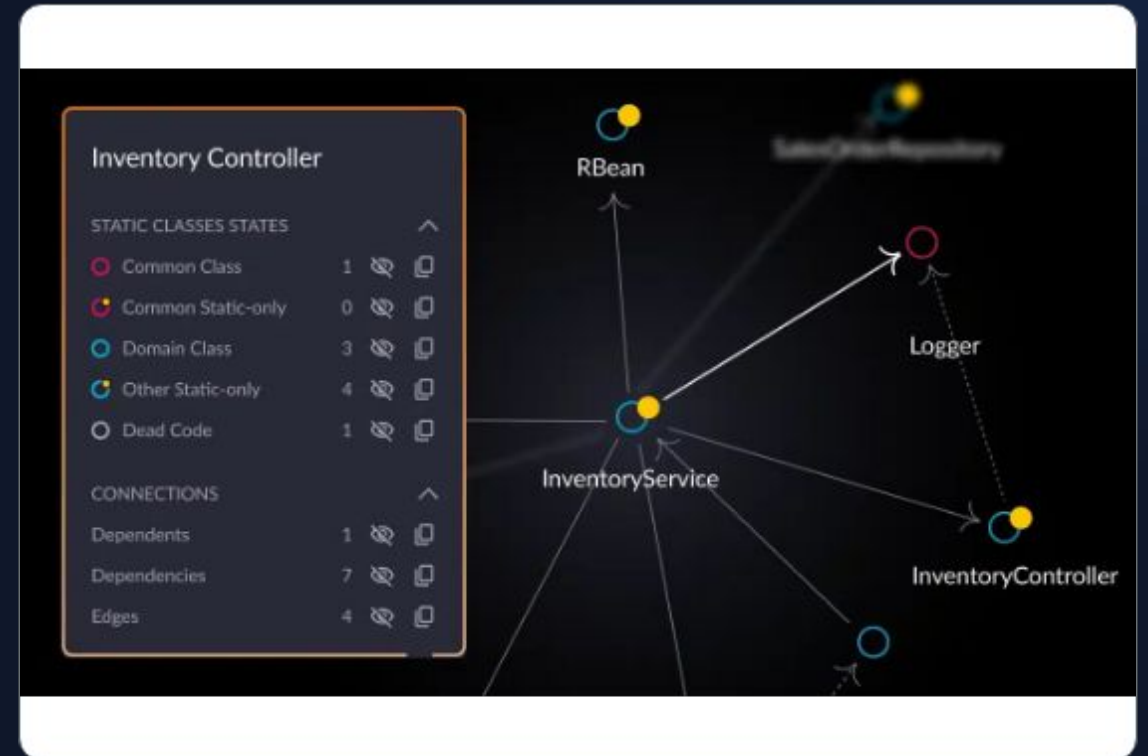
Статичне vs Динамічне тестування

Статичне тестування: Оцінка робочих продуктів без виконання коду. Ключове для підходу "Shift Left".

- Знаходить дефекти напрому (неоднозначності, пропущені вимоги, архітектурні хиби).
- Надзвичайно економічно вигідне на ранніх етапах.

Динамічне тестування: Запуск ПЗ для пошуку відмов (failures).

- Виявляє дефекти, які проявляються лише під час роботи системи (напр., витоки пам'яті).
- Результатом є відмови, які потребують подальшого відлагодження.



Робочі продукти для статичного тестування



Документація

Специфікації вимог, контракти, плани проєктів, тест-плани, користувацькі історії (User Stories), критерії приймання.



Дизайн та Архітектура

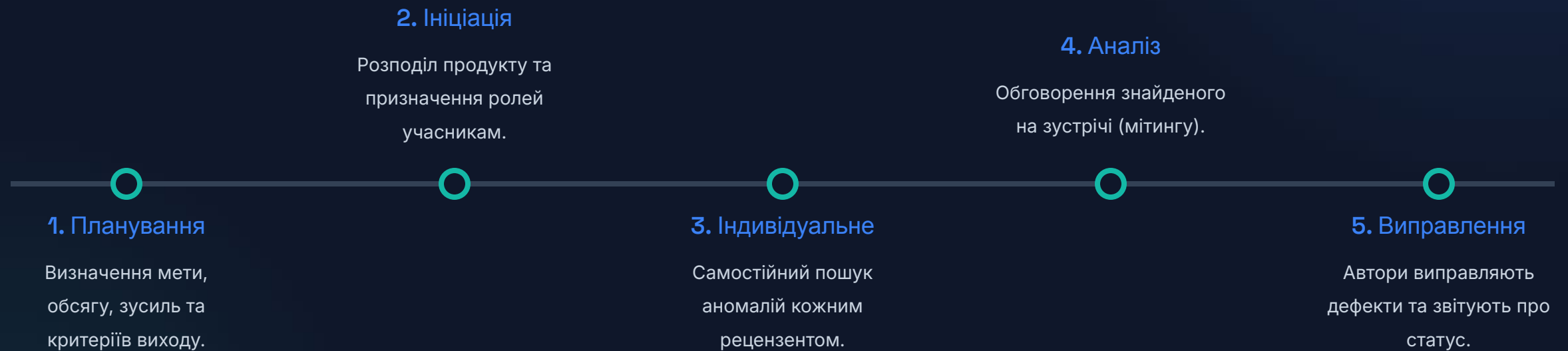
Архітектурні специфікації, моделі баз даних, макети інтерфейсів (UI/UX).



Вихідний код

Аналіз коду за допомогою інструментів статичного аналізу (пошук вразливостей безпеки, недосяжного коду, перевірка синтаксису).

Процес рецензування (Review Process)



Ролі в процесі рецензування



Автор (Author)

Створює документ та виправляє його після рецензування. Не може бути модератором чи лідером інспекції.



Модератор (Moderator)

Фасилітує мітинги, слідкує за часом та створює безпечну атмосферу для висловлення думок.



Рецензент (Reviewer)

Шукає аномалії (може бути тестувальником, розробником, бізнес-аналітиком або експертом).

Типи рецензування

Тип	Опис та формальність
Неформальне (Informal)	Немає формального процесу. Мета: швидкий пошук аномалій (напр. Pair Programming).
Проходження (Walkthrough)	Ведеться Автором. Мета: навчання, збір ідей, досягнення консенсусу щодо документу.
Технічне (Technical Review)	Виконується технічними фахівцями, ведеться Модератором. Вирішення тех. проблем.
Інспекція (Inspection)	Найбільш формальний тип. Ведеться Модератором, збір метрик, суворі критерії входу/виходу.

Розділ 4: Аналіз та дизайн тестів

Категорії технік тестування, чорний ящик, білий ящик та
підходи на основі досвіду.

Питання до Розділу 4 (9 питань)

Техніки Чорного ящика

- ? За якими ознаками техніки поділяються на Black-box, White-box та Experience-based?
- ? Як розрахувати 100% покриття для Еквівалентного розбиття (Equivalence Partitioning)?
- ? У чому різниця між 2-value BVA та 3-value BVA при аналізі граничних значень?
- ? Для яких сценаріїв найбільш доцільно використовувати Таблиці рішень (Decision Table Testing)?
- ? Що означають критерії покриття "All states" та "Valid transitions (0-switch)" у State Transition Testing?

Інші техніки та Колаборація

- ? Поясніть різницю між Statement Coverage та Branch Coverage.
- ? Чи гарантує 100% Branch Coverage виконання 100% Statement Coverage (і навпаки)?
- ? У чому різниця між техніками Error Guessing та Exploratory Testing?
- ? Як підхід "3 C's" (Card, Conversation, Confirmation) допомагає у написанні User Stories та Acceptance Criteria?

Категорії технік тестування



Black-box (Чорний ящик)

Базуються на специфікаціях (вимогах) без знання внутрішньої структури коду. Стійкі до рефакторингу коду.



White-box (Білий ящик)

Базуються на внутрішній структурі, архітектурі та коді. Потребують технічних знань для написання.



Experience-based (На основі досвіду)

Використовують знання, інтуїцію та досвід тестувальника. Доповнюють інші формальні техніки.

Еквівалентне розбиття (Equivalence Partitioning)



Поділяє дані на класи (партиції) з очікуванням, що всі значення в одному класі обробляються програмою однаково.



Валідні класи: значення, які система повинна приймати.



Невалідні класи: значення, які система повинна відхиляти (тестуються ізольовано для уникнення маскування дефектів).



Для 100% покриття необхідно вибрати хоча б одне значення з кожної партиції (валідної та невалідної).

Аналіз граничних значень (BVA)

Застосовується лише для **впорядкованих** партицій. Баги часто ховаються саме на межах (через неправильне використання операторів $<$, $>$, $<=$, $>=$).

- **2-value BVA:** Береться сама границя та найближче сусіднє значення поза цією границею (з суміжної партиції). Приклад для $[10-20]$: 9, 10, 20, 21.
- **3-value BVA:** Береться сама границя та обидва сусідні значення з обох сторін. Є більш суворим і знаходить більше дефектів. Приклад для межі 10: 9, 10, 11.



Таблиці рішень (Decision Table Testing)

Ідеально підходять для тестування складної бізнес-логіки з багатьма комбінаціями умов (T - True, F - False).

Умови / Дії	Правило 1	Правило 2	Правило 3
Користувач авторизований?	T	T	F
Баланс > 0?	T	F	- (Не має значення)
Результат: Дозволити покупку	X		
Результат: Показати помилку		X	X

Тестування переходів станів (State Transition)

Модель станів

Використовується, коли система може перебувати в різних станах, і поведінка залежить від поточного стану (напр., банкомат, статус замовлення).

- **Стан (State):** Поточний статус об'єкта.
- **Перехід (Transition):** Зміна стану.
- **Подія (Event):** Тригер, що викликає перехід.

Рівні покриття

- **All States Coverage:** Відвідати всі можливі стани системи. Найслабкіший рівень.
- **Valid Transitions (O-switch):** Пройти всі дозволені (валідні) переходи хоча б один раз. Найбільш популярний.
- **All Transitions Coverage:** Перевірити всі валідні та спробувати виконати всі невалідні переходи (тестування негативних сценаріїв).

Техніки білого ящика (White-box)

100%

Branch покриває Statement

Statement & Branch Coverage

Покриття операторів (Statement Coverage): Вимірює відсоток виконаних рядків коду під час тестів. 100% означає, що кожен рядок виконано хоча б раз, але це не гарантує перевірку всієї логіки умов.

Покриття гілок (Branch Coverage): Перевіряє всі можливі результати рішень (if-else: True та False). Є значно сильнішою метрикою.

Примітка: Досягнення 100% покриття гілок автоматично гарантує 100% покриття операторів (але не навпаки).

Техніки на основі досвіду



Error Guessing

Вгадування помилок.

Тестувальник передбачає, де розробники могли помилитись, базуючись на минулих проектах (напр. ділення на нуль, некоректні дати).



Exploratory Testing

Дослідницьке тестування.

Одночасне вивчення системи, дизайн та виконання тестів (через сесії та чартери). Чудово працює при нестачі документації.



Checklist-based

Тестування на основі списків перевірок (Checklists). Списки створюються на основі досвіду і дають загальний гайдлайн (напр., евристики юзабіліті).

Спільні підходи (Collaboration-based)

User Stories (3 C's)

Спільне написання вимог командою розробки, бізнесом та QA.

- **Card:** Опис формату "Як [роль], я хочу [дія], щоб [цінність]".
- **Conversation:** Обговорення деталей.
- **Confirmation:** Критерії приймання.
- Хороша User Story відповідає критеріям **INVEST**.

Acceptance Criteria (Критерії приймання)

Визначають межі задачі та формують базу для ATDD (Acceptance Test-Driven Dev).

- **Scenario-oriented:** BDD формат Given/When/Then.
- **Rule-oriented:** Списки правил або таблиці.
- Тести створюються ДО написання коду і стають виконуваними вимогами (Executable Requirements).

Розділ 5: Управління тестуванням

Планування, оцінка зусиль, управління ризиками та
дефектами, контроль процесів.

Питання до Розділу 5 (9 питань)

Планування та Оцінка

- ? Яка основна мета створення тест-плану та які розділи він зазвичай містить?
- ? У чому полягає роль тестувальника під час планування релізу (Release Planning) та ітерації?
- ? Яка різниця між критеріями входу (Entry Criteria) та виходу (Exit Criteria)?
- ? Як розрахувати оцінку зусиль за допомогою техніки Three-point estimation та Planning Poker?
- ? Що таке "Тестова піраміда" та "Тестові квадранти" (Testing Quadrants)?

Ризики та Контроль

- ? Які фактори найчастіше використовуються для пріоритезації тест-кейсів?
- ? Поясніть різницю між проєктними (Project) та продуктовими (Product) ризиками.
- ? Як аналіз продуктових ризиків впливає на обсяг (scope) та ретельність тестування?
- ? Які обов'язкові поля повинен містити якісний звіт про дефект (Defect Report)?

Тест-план: Мета та зміст



Мета: Описує обсяг, ресурси та графік тестових активностей. Слугує засобом комунікації в команді.



Зміст: Контекст, припущення, зацікавлені особи (ролі), комунікація, реєстр ризиків.



Підхід до тестування: Рівні та типи тестування, критерії входу/виходу, метрики, середовища, дані.



Графік: Розклад ітерацій, релізів, бюджет та необхідні ресурси (ISO/IEC/IEEE 29119-3).

Оцінювання зусиль (Estimation)

Техніка	Опис
Ratios (На основі пропорцій)	Метрична оцінка на основі історичних даних (напр. співвідношення девів до QA 3:2).
Extrapolation (Екстраполяція)	Передбачення зусиль на основі середнього показника минулих ітерацій.
Wideband Delphi (Planning Poker)	Експертна оцінка. Учасники роблять оцінки ізольовано, потім обговорюють розбіжності до досягнення консенсусу.
Three-point estimation	Експертна оцінка з трьох значень: оптимістичне (a), найбільш ймовірне (m), песимістичне (b). Формула: $E = (a + 4m + b) / 6$.

Управління ризиками

Проектні ризики

Впливають на здатність проекту досягти цілей (графік, бюджет).

- ▷ **Організаційні:** затримки, урізання бюджету.
- ▷ **Людські:** нестача навичок, конфлікти.
- ▷ **Технічні:** погані інструменти, scope creep.
- ▷ **Постачальники:** банкрутство, затримки 3-rd party.

Продуктові ризики

Стосуються характеристик якості самого продукту (ПЗ).

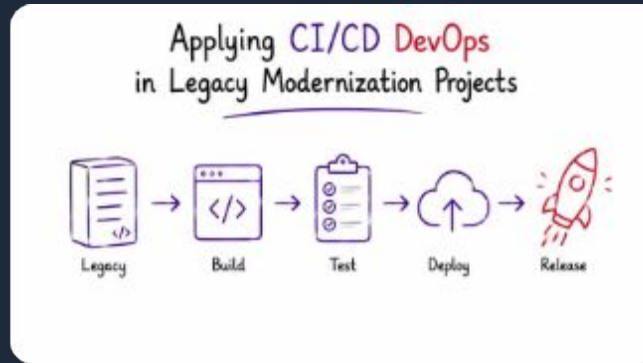
- ▷ Відсутній функціонал, некоректні розрахунки.
- ▷ Вразливості безпеки, погана продуктивність.
- ▷ **Рівень ризику:** Розраховується як Ймовірність (Likelihood) x Вплив (Impact).

Контроль тестування та Метрики



Моніторинг

Збір метрик: покриття вимог, щільність дефектів, статус тест-кейсів, показники залишкового ризику.



Контроль

Дії у відповідь на метрики: зміна пріоритетів, додавання ресурсів, перегляд розкладу та критеріїв виходу.



Звітування

Test Progress Reports (під час роботи) та Test Completion Reports (по завершенню фази або релізу).

Управління дефектами (Bug Report)



Ідентифікатор & Title: Унікальний ID та коротке зрозуміле резюме суті проблеми.



Оточення (Environment): Версія ПЗ, ОС, Браузер, тестові дані, на яких відтворено баг.



Steps to Reproduce: Чіткі послідовні кроки, які дозволять розробнику точно відтворити відмову.



Результати: Очікуваний результат (Expected) проти Фактичного результату (Actual).



Severity & Priority: Severity (Серйозність - вплив на систему) та Priority (Пріоритет - нагальність виправлення).

Розділ 6: Інструменти тестування

Автоматизація, переваги, ризики та підтримка CI/CD процесів.

Питання до Розділу 6 (7 питань)

Підтримка процесу

- ❓ Які типи інструментів використовуються для підтримки управління тестуванням (Test Management)?
- ❓ Як інструменти статичного тестування допомагають виявляти дефекти без запуску коду?
- ❓ Для яких завдань найчастіше застосовуються інструменти підтримки виконання тестів (Test Execution)?
- ❓ Які інструменти є критично важливими для побудови ефективного DevOps пайплайну (CI/CD)?

Автоматизація (Переваги/Ризики)

- ❓ Назвіть щонайменше три ключові переваги впровадження автоматизації тестування на проєкті.
- ❓ Які основні ризики пов'язані з автоматизацією (нереалістичні очікування, високі витрати на підтримку)?
- ❓ Чому купівля дорогого інструменту для автоматизації не гарантує автоматичного покращення якості?

Автоматизація: Переваги та Ризики

Переваги (Benefits)

- ▶ Економія часу на повторюваних ручних задачах (регресія).
- ▶ Уникнення людських помилок при рутинних перевірках.
- ▶ Об'єктивна оцінка (наприклад, вимірювання Code Coverage).
- ▶ Швидкий фідбек (CI/CD пайплайни) та прискорення Time-to-Market.

Ризики (Risks)

- ▶ Нереалістичні очікування керівництва.
- ▶ Недопрогнозування зусиль на підтримку (maintainability) тестових скриптів.
- ▶ Зайва залежність від інструменту, втрата критичного мислення (ручне exploratory тестування залишається важливим).
- ▶ Проблеми з підтримкою вендора або Open-Source ліцензіями.

Запитання?

Дякую за увагу до лекції з Основ тестування.

Image Sources



<https://media.istockphoto.com/id/1048336630/vector/concept-application-testing-debugging-development-process-programming-and-coding-software.jpg?s=1024x1024&w=1024&h=1024&f=webp&fit=cover&position=center>

Source: www.istockphoto.com



https://intertec.io/media/a_clean_whiteboard_sketchnote_style_illustration_f1d52623dc.webp

Source: www.intertec.io



https://img.magnific.com/free-vector/software-developer-concept-idea-programming-coding-system-development-digital-technology-software-developing-company-writing-code-isolated-vector-illustration_613284-3389.jpg?sem=ais_hybrid&w=740&q=80

Source: www.magnific.com



https://media.geeksforgeeks.org/wp-content/uploads/20260211105334877222/system_testing.webp

Source: www.geeksforgeeks.org



https://blog.cloudflare.com/_image?href=https%3A%2F%2Fblog.cloudflare.com%2F_emdash%2Fapi%2Fmedia%2Ffile%2F01KW475MSTM04DWZMY6AQZH8ZZ.png&w=1600&h=900&f=webp&fit=cover&position=center

Source: blog.cloudflare.com



https://vfunction.com/wp-content/uploads/2024/09/blog_static_vs_dynamic_code_analysis_opt_1.webp

Source: vfunction.com

Image Sources



<https://cdn.sanity.io/images/qe2cvmjb/production/5a599052f827804bbd8325d0702bf6867d9f00aa-1560x816.png?auto=format&fit=max&w=1200>

Source: www.mountangoatsoftware.com



<https://slidemodel.com/wp-content/uploads/00-risk-and-mitigation-slide-cover.png>

Source: slidemodel.com