TEST-DRIVEN DEVELOPMENT

Калашніков Богдан ПЗС-2

TEST-DRIVEN DEVELOPMENT

- Методологія розробки програмного забезпечення, не тестів.
- TDD is a design activity.
- З’явилася в кінці 90-х років.
- Походить від концепції екстремального програмування “test-first”.

ПЕРШИЙ ЗАКОН TDD

- You may not write production code until you have written a failing unit test. (Robert Cecil Martin)

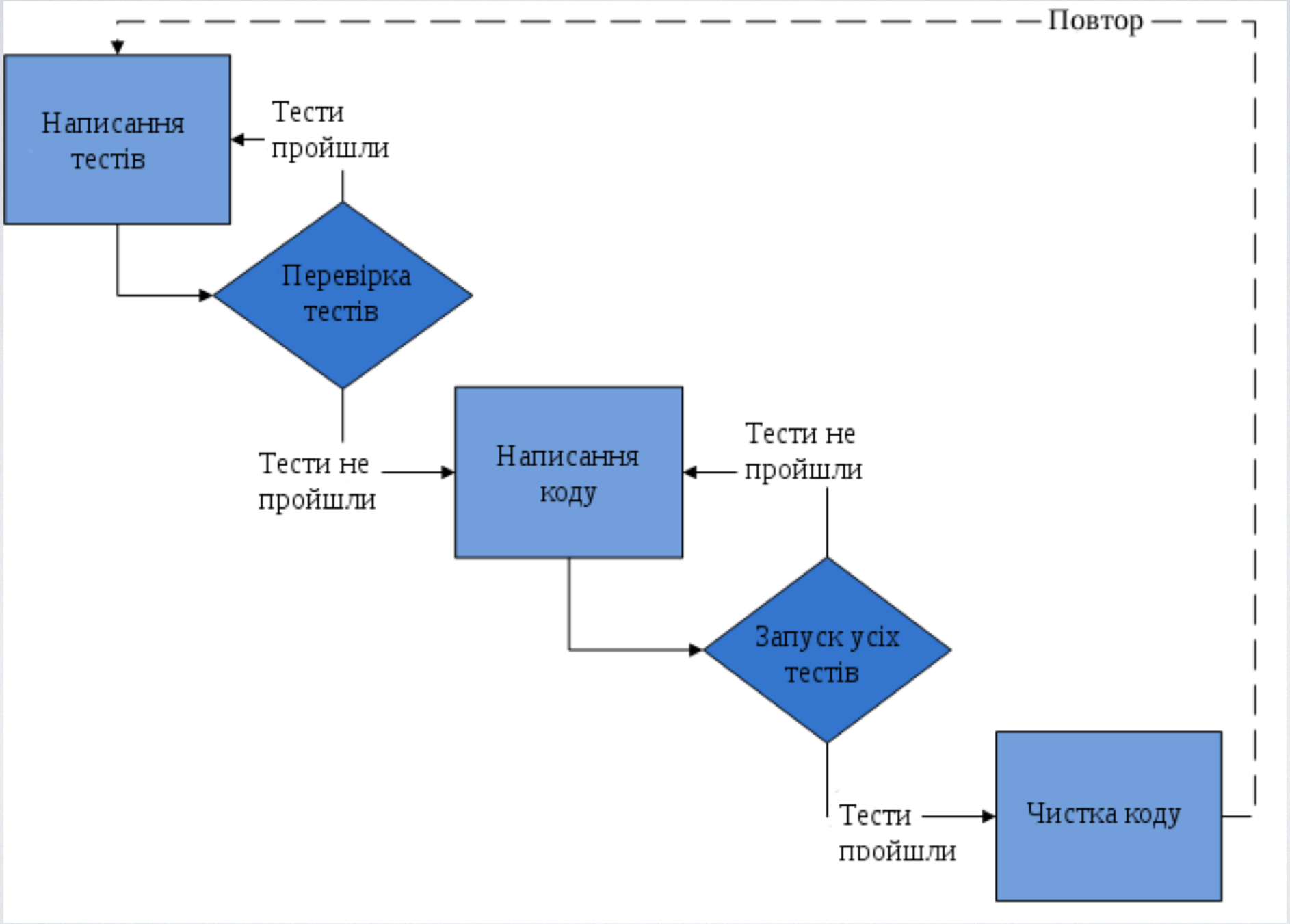
- Не можна писати код, поки немає невдалого тесту.

ДРУГИЙ ЗАКОН TDD

- You may not write more of a unit test than is sufficient to fail, and not compiling is failing. (Uncle Bob)
- Не потрібно писати більше тестів, ніж необхідно для невдалого проходження.

ТРЕТІЙ ЗАКОН TDD

- You may not write more production code than is sufficient to pass the currently failing test. (Bob Martin)
- Не можна писати більше коду, ніж необхідно для вдалого проходження тесту.


Повтор
Написання тестів
Тести пройшли
Перевірка тестів
Тести не пройшли
Написання коду
Тести не пройшли
Запуск усіх тестів
Тести пройшли
Чистка коду

I НАПИСАТИ ТЕСТ

- Тест потрібно писати в найпершу чергу.
- Необхідно чітко знати та розуміти вимоги та специфікації.

2 ПЕРЕВІРКА ТЕСТУ

- Перевірити коректність тесту.
- Оскільки тестованого коду ще немає – тест має бути невдалим.

3 НАПИСАННЯ КОДУ

- Потрібно написати стільки коду, скільки необхідно для вдалого проходження тесту, не більше.
- На данному етапі головне пройти тест, якість коду виправимо в наступних кроках.

4 ЗАПУСК ТЕСТІВ

- Треба впевнитись, що програма проходить всі тести, отже, відповідає вимогам.
- При невдачі можна повернутись до написання коду.

5 РЕФАКТОРИНГ

- Тепер можна вдосконалити написаний код.
- Оскільки всі тести вже добре працюють, можемо сміливо проводити зміни.

СТИЛЬ РОЗРОБКИ

- KISS (Keep It Simple, Stupid)
- YAGNI (You Ain't Gonna Need It)
- Fake It Till You Make It

ПРІОРІТЕТ ТРАНСФОРМАЦІЙ

- Під час розробки код програми постійно змінюється та трансформується.
- Для уникнення складнощів, необхідно надавати перевагу більш простим трансформаціям над складнішими.

ПОМИЛКИ ПРИ ЗАСТОСУВАННІ TDD

- Дуже багато тестів без коду.
- Тест для класів вищих рівнів (фасадів).
- Відсутність фази рефакторингу.
- Unit тести – це вже TDD.
- Запуск TDD на великих проектах із legacy кодом.
- Використання TDD без достатніх знань OOP.
- Testing oriented development anti-pattern.

ПРОЕКТИ НЕ ДЛЯ TDD

- Розробка інтерфейсу користувача
- Робота з базами даних та зовнішніми ресурсами (СКБД)
- Великі проекти з legacy-кодом, які розроблялись без використання тестів

FAKE- І MOCK- ОБ'ЄКТИ

- Тести не повинні виходити за межі тестованого модуля, використовувати мережу, робити запити до бази даних.
- Для цього необхідні Fake- та Mock- об'єкти.
- Fake vs mock: mock-об'єкти містять твердження, що перевіряють поведінку тестованого коду.

ВИКОРИСТАННЯ FAKE, MOCK ОБ'ЄКТІВ

- 1. Оголошення інтерфейсу для доступу до зовнішнього ресурсу.
- 2. Створення інтерфейсу, який повинен мати дві реалізації.
- - Перша надає доступ до зовнішнього ресурсу.
- - Друга є fake- або mock- об'єктом.

ІНСТРУМЕНТИ

- **Moq**
- **jMock**
- **NMock**
- **EasyMock**
- **Typemock**
- **jMocki**
- **Unitils**
- **Mockito**
- **Mockachino**
- **PowerMock**

ПЕРЕВАГИ TDD

- Тести, тести, ще більше тестів.
- Осмислення завдання та функціональності.
- Більш простий код програми.
- Зменшення часу розробки – економія на “debugging”.
- Зменшення кількості помилок в коді.

ПЕРЕВАГИ PART II

- Модульність та простий рефакторинг коду.
- Впевненість та продуктивність розробника.
- Повне покриття коду тестами.
- Тести можуть бути гарною документацією.

НЕДОЛІКИ TDD

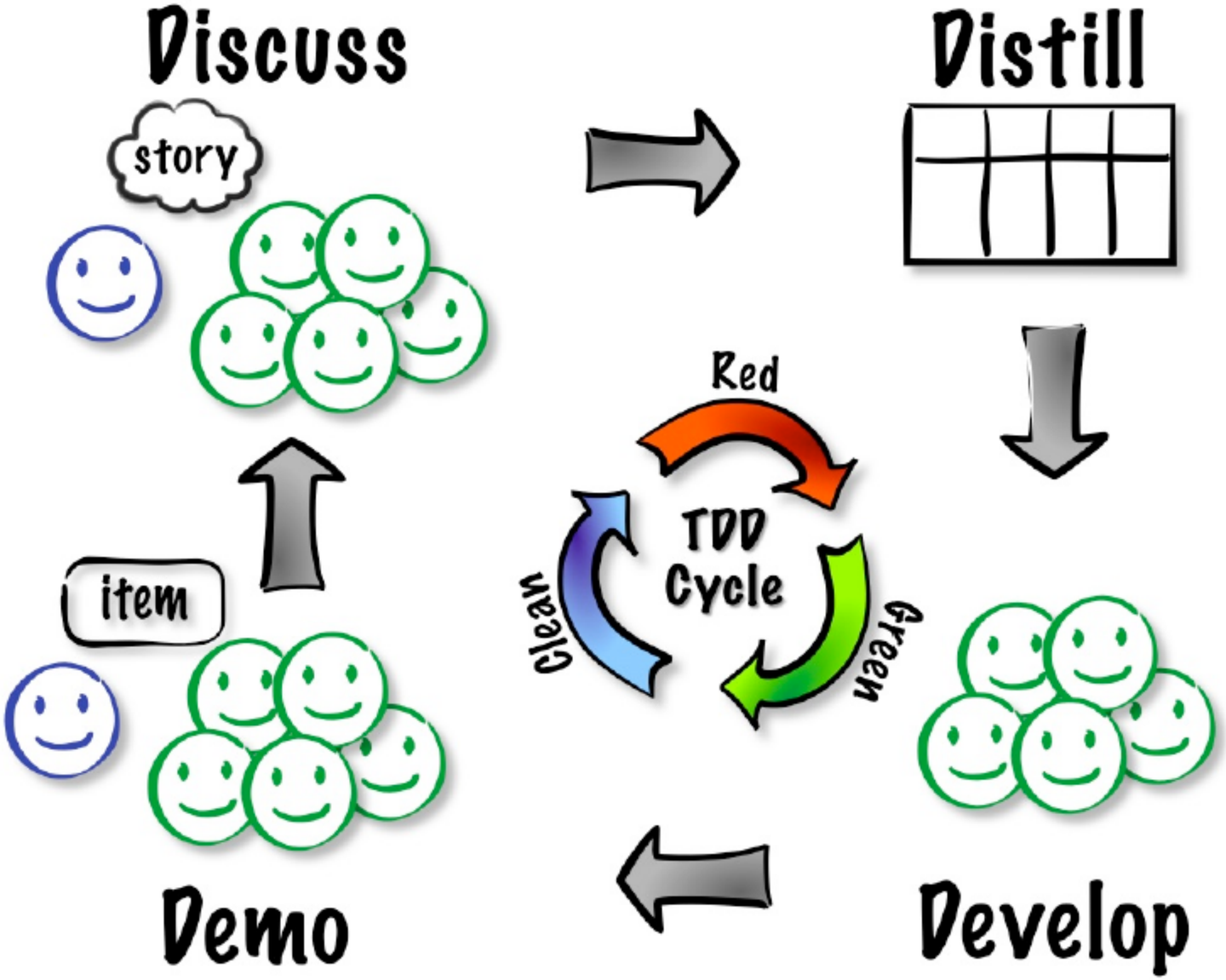
- Тяжко звикнути.
- Не всі частини коду можна протестувати.

- Помилки при неправильній інтерпретації вимог розробником.
- Хибне почуття надійності.
- Накладні витрати та великий обсяг тестів.

ACCEPTANCE TDD

- ATTD – методологія розробки, подібна до TDD, але передбачає використання acceptance тестів.
- Acceptance tests – тести, що перевіряють функціональність програми на відповідність вимогам замовника.

Development (ATDD) Cycle



TDD VS ATDD IN AGILE

- TDD і ATDD передбачають циклічність розробки.
- ATTD передбачає більшу взаємодію із замовником (чи project-owner).
- ATDD збільшує ефективність розробки.
- ATDD залучає всіх до участі у розробці проекту – всі знають вимоги.
- “TDD – робити речі правильно, ATDD – робити правильні речі”.

ІНСТРУМЕНТИ ДЛЯ ATDD

- Concordion
- FitNesse
- Robot Framework

ЗАМІСТЬ ВИСНОВКУ

ЗАПИТАННЯ?

ДЯКУЮ ЗА УВАГУ!

Калашніков Богдан ПЗС-2