2023년 2월 19일
TDD, BDD, DDD
보고서

TDD, BDD, DDD

임원호

-목차-

[참고 문헌]

Ⅰ. TDD

1. TDD 정의

“TDD란 Test Driven Development의 약자로 ‘테스트 주도 개발’이라고 한다. 반복적인 테스트를 이용한 소프트웨어 방법론으로써, 작은 단위(Unit)의 테스트 케이스를 작성하고, 이를 통과하는 코드를 추가하는 단계를 반복하여 구현한다. 짧은 개발 주기의 반복에 의존하는 개발 프로세스이며, 애자일 방법론 중 하나인 eXteam Programming(XP)의 ‘Test-First’개념에 기반을 둔 단순한 설계를 중요하게 한다.”[1]

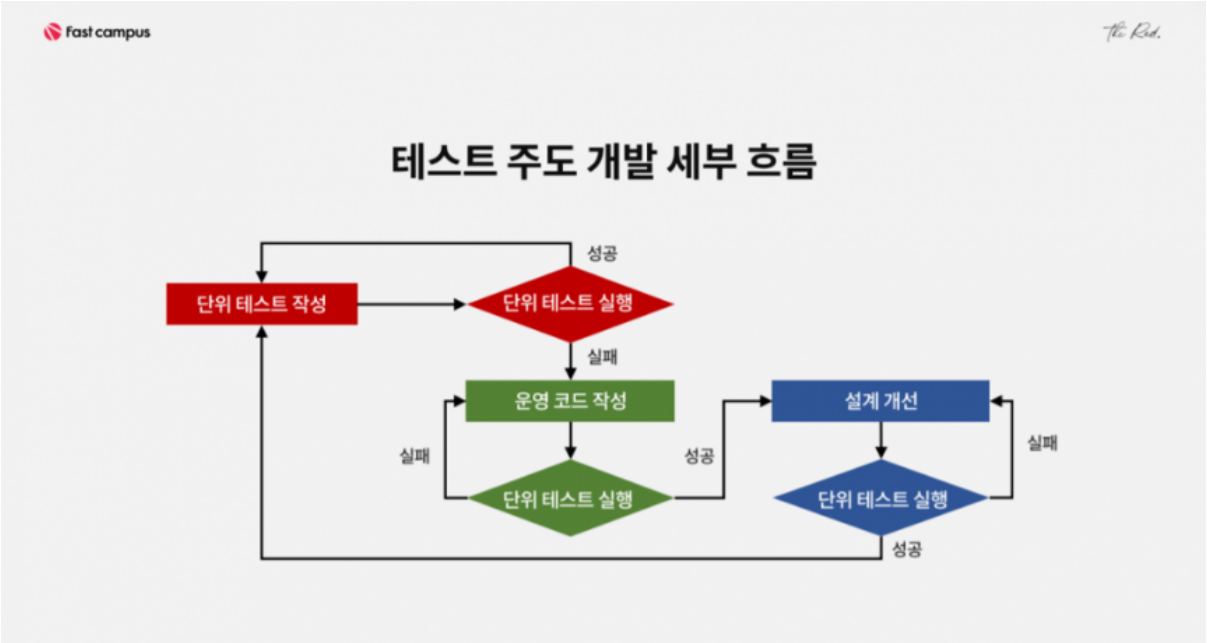
2. TDD 왜 사용 할까?

TDD의 정의에서, 작은 단위의 테스트 케이스를 작성하고, 추가하는 단계를 반복하여 구현한다고 정의 되어있다. 우리는 일반적인 프로그램을 만든다고 할 때, 설계 -> 구현 ->

[1] Inpa Dev, “TDD”, [TEST] 📚 TDD 방법론 (테스트 주도 개발) - 알기 쉽게 정리,(2023.02.20.최종검색).

디버깅의 과정을 거치게 되어있는데, 실제 설계 및 구현은 오래 걸리지 않지만, 디버깅에 상당한 시간이 걸리는 것을 알고 있다. 또한 디버깅 과정에서 Main문을 이용한 디버깅과, System.out.print를 이용한 디버깅 방법을 사용하고 있는데, 한번 스파게티처럼 꼬이게 되면, 버그를 찾기란 상당히 힘들게 된다. 모든 것은 설계 및 구현의 문제가 발생했다고 볼 수 있지만, 설계와 구현에서 발견하지 못한 부분과 잘못 설계된 과정이 디버깅 끝에서 발견될 수 있기 때문에, 작은 단위부터 하나씩 훑는 개발 방법인 TDD가 사용되었다. 하지만, TDD에 대한 개발자들의 의견은 엇갈리게 되는데, 회사마다 일하는 방식이나, 처한 업무 환경에 편차가 있다 보니, 현실과 괴리감이 크다는 게 가장 많은 의견이었다.

3. TDD 의 사용



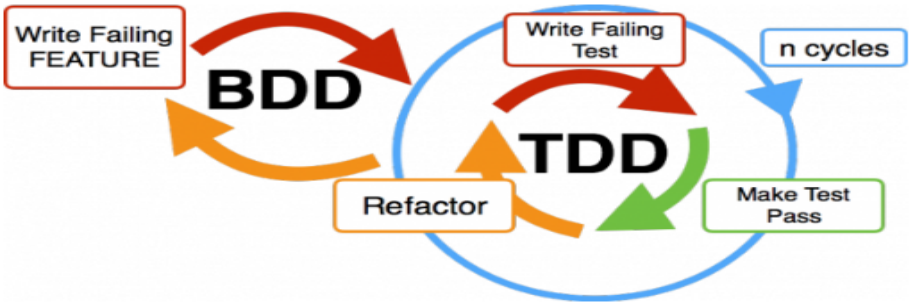
TDD는 RED -> GREEN -> REFACTOR의 과정을 계속해서 반복하게 된다. Red 단계에서는 실패하는 테스트 코드를 작성하여, 실패를 성공하는지 확인하는 코드를 작성한다. Green단계에선 테스트 코드를 성공시키기 위한 실제 코드를 작성한다. 성공시킬 수 있는 실제코드를 작성했으므로, 우리는 실제 코드를 마음 놓고, 리펙토링할 수 있다. 리펙토링을 최대한 수행한 뒤, 다음 테스트 코드를 작성하면 된다. 여기서 가장 핵심적인 부분은 실패하는 테스트 코드를 작성할 때까지, 실제 코드를 작성하지 않는 것과, 실패하는 테스트를 통과할 정도의 최소 실제 코드를 작성해야 하는 것이다.

TDD를 통해 얻는 결과는 상당하다.

1. 기존 개발과 달리 테스트 코드를 이용해 실제 코드를 작성하기 때문에, 디버깅 시간을 단축할 수 있다.
2. 코드가 내 손에서 벗어나기 전에 가장 빠르게 피드백을 받을 수 있다.
3. 작성한 코드가 가지는 불안정성을 개선하여 생산성을 높일 수 있다.
4. 재설계 시간을 단축할 수 있다.
5. 기능을 추가하거나, 삭제하는데 용이하다.

Ⅱ. BDD

1. BDD 정의



(그림 : https://saucelabs.com/blog/a-two-minute-bdd-overview)

"BDD는 Behaviour-Driven Development의 약자로, 처음에 TDD가 가지는 테스트 이외의 측면, 특히 설계적인 측면을 강조하게 된다. TDD를 처음 접하는 사람의 오해를 덜어주고, 초심자가 TDD를 더 빨리 익힐 수 있도록 하자는 취지에서 ThoughtUJorks의 Dan North에 의해 탄생했다."[2]

2. BDD 왜 사용 할까?

BDD 자체가 TDD에 기반을 두고 있다는 점에서 TDD와는 둘의 엄청난 차이를 보이지는 않는다. BDD와 TDD에서 가장 중요한 것은 말의 변화인데, 테스트라는 단어를 비롯해, TDD에서 쓰이는 말들이, TDD에 대한 각종 오해의 근원이라고 생각해, 바로잡은 것이 BDD의 시초이다. TDD에서 말하는 코드를 작성하기 전에 테스트를 먼저 작성해야 한다. => 이 말은 BDD에서는 코드를 작성하기 전에 코드가 수행할 행위에 대한 명세를 먼저 작성해야 한다.로 바꿔주는 것이다. TDD의 assert에서 사용하는 Junit의 스타일 보다는, actual_value.should_be와 같이 명확하게 작성하는 것이 특징이다.

3. BDD 의 사용

BDD는 TDD에서 사용하는 Junit 시리즈의 프레임워크를 사용하기도 하지만, JBehave라는 프레임워크를 사용하기도 한다. 또한 자바를 제외하고, 루비나, 자바스크립트에도 BDD 프레임워크가 존재한다. 대게, 하나의 특징을 여러 개의 스토리로 만들고, Given - When - Then의 형식으로 코드를 만들어 이를 기반으로 테스트하며 개발한다. 먼저 하나의 특징을 가지고 스토리를 구현할 때, When부터 Then그리고 Given순으로 정리하게 된다. 숫자를 통해 예를 들어보면, When : 사용자가 +버튼을 클릭하면, Then 숫자가 1증가하고, Given 증가한 숫자를 사용자에게 나타내 준다. 이와 반대로 Given순으로 하는 경우도 있다. Given 10보다 작은 상황에서, When + 버튼을 클릭하면, Then 숫자가 1 증가한다. 이렇게 프로젝트를 진행하면, 자연스럽게 진척도를 체크할 수 있는 게 된다.

---

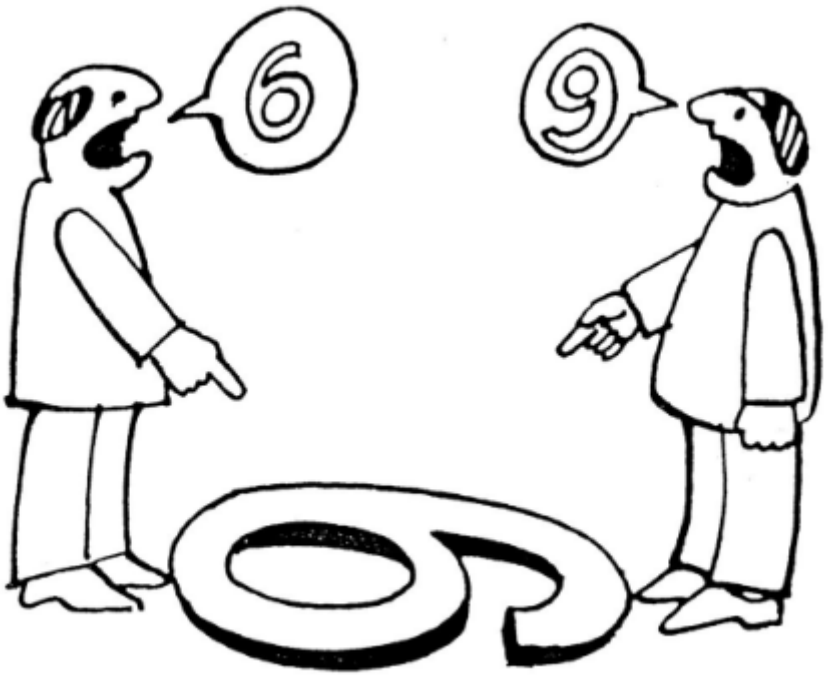
[2] DATAON-AIR, "BDD", BDD(Behavior Driven Development) – JSSpec을 이용한 행동주도 개발, (2023.02.20.최종검색).

Ⅲ. DDD

1. DDD 정의

“DDD란 “Domain”[3]-Driven Design의 약자로, 비즈니스 Domain별로 나누어 설계하는 방식이다. 기존의 애플리케이션 설계가 비즈니스 Domain에 대한 이해가 부족한 상태에서 설계 및 개발되었다는 반성에서 출발하였다.”[4]

2. DDD 왜 사용 할까?



DDD에서는 기존의 현업에서 IT로의 단방향 소통구조를 탈피하여, 현업과 IT의 쌍방향 커뮤니케이션을 매우 중요하게 생각하는 것이 핵심이다. DDD의 핵심 목표는 기능 간의 의존성은 최소화하며, 응집성은 최대화(객체 지향 패러다임과 같다)하는 것이다. DDD는 전략적 디자인과, 기술적 디자인으로 나눌 수 있다. 전략적 디자인은 개념 설계이고, 기술적 디자인은 프로그래밍하기 위한 구체적 설계라고 할 수 있다.

---

[3] Domain : 소프트웨어로 해결해야할 문제의 영역
[4] Happy@Cloud, “DDD”, https://happycloud-lee.tistory.com/94, (2023.02.20.최종검색).

[참고 문헌]

Inpa Dev, “TDD”, [TEST] 📚 TDD 방법론 (테스트 주도 개발) - 알기 쉽게 정리,(2023.02.20.최종검색).

(그림 : https://saucelabs.com/blog/a-two-minute-bdd-overview)

DATAON-AIR, “BDD”, BDD(Behavior Driven Development) – JSSpec을 이용한 행동주도 개발, (2023.02.20.최종검색).