Skillnaden mellan TDD och BDD

TDD (Test-Driven Development) och **BDD (Behavior-Driven Development)** är båda metodiker som hjälper till att förbättra kvaliteten på programvaran genom att använda tester tidigt i utvecklingsprocessen. Trots att de har likheter, är deras fokus och tillämpningar olika.

Syfte och Fokus

TDD fokuserar på att skriva tester för att verifiera att kodens funktionalitet fungerar korrekt. TDD handlar om att säkerställa att varje liten del av koden uppfyller sitt specifika tekniska syfte.

Målet med TDD är att skriva kod som är tekniskt korrekt och robust genom att börja med enhetstester för varje liten funktion eller metod.

Syfte och Fokus

BDD fokuserar på att beskriva och testa beteendet hos systemet baserat på affärskrav och användarens behov. Det är en vidareutveckling av TDD som inkluderar icke-tekniska intressenter i utvecklingsprocessen.

Målet med BDD är att skapa en gemensam förståelse för hur programvaran ska fungera och leverera värde till användaren.

Hur de Definierar Tester

TDD använder enhetstester för att testa specifika kodbitar, som metoder och funktioner, på en låg nivå. Testerna fokuserar på detaljerad funktionalitet, t.ex. att en metod returnerar rätt resultat eller hanterar fel korrekt.

Exempel på TDD-test: Ett test för en metod som adderar två tal och returnerar summan.

BDD använder beteendebaserade tester, ofta skrivna i naturligt språk, för att beskriva hur systemet ska agera i olika scenarier. Dessa tester är på en högre nivå och beskriver hur olika funktioner ska fungera utifrån användarens perspektiv.

Exempel på BDD-scenario: "Som användare vill jag kunna logga in på min konto-sida så att jag kan se mina personliga uppgifter."

Struktur och Syntax

TDD använder ofta strukturen **Arrange-Act-Assert (AAA)**, vilket innebär att utvecklaren arrangerar en testmiljö, utför en åtgärd, och sedan verifierar resultatet.

```
[Fact]
public void Addition_ShouldReturnCorrectSum()
{
    // Arrange
    int a = 2;
    int b = 3;

    // Act
    int result = Add(a, b);

    // Assert
    Assert.Equal(5, result);
}
```

Struktur och Syntax

BDD använder **Given-When-Then**-syntax, som gör testet mer läsbart för icke-tekniska intressenter. Detta format beskriver kontexten (Given), en åtgärd (When), och förväntat resultat (Then).

Scenario: Lyckad inloggning
Given att användaren är på inloggningssidan
When användaren anger korrekt användarnamn och lösenord
Then ska användaren komma till sin dashboard

Vem är Involverad?

TDD är mestadels ett verktyg för utvecklare och testare. Eftersom testerna är tekniska och fokuserar på kodens funktionalitet på en låg nivå, är de främst avsedda för det tekniska teamet.

BDD inkluderar både tekniska och icke-tekniska intressenter (som produktägare och kunder). Testerna är skrivna på ett språk som alla kan förstå, vilket underlättar samarbete och säkerställer att hela teamet har en gemensam bild av hur systemet ska fungera.

Output och Dokumentation

TDD resulterar i en samling enhetstester som bevisar att specifika kodbitar fungerar korrekt. Testerna fungerar som dokumentation över den tekniska implementeringen och kan hjälpa framtida utvecklare att förstå hur metoder fungerar.

BDD ger en samling beteendebaserade tester som fungerar som "levande dokumentation" över systemets förväntade beteende. Dessa tester kan lättare förstås av hela teamet och fungerar som en källa till både dokumentation och kravspecifikation.

Sammanfattning

TDD

Fokuserar på **teknisk funktionalitet**

Tester skrivs i tekniskt språk

Används främst av utvecklare och testare

Använder **Arrange-Act-Assert** för teststruktur

Resulterar i tekniska tester och dokumentation

BDD

Fokuserar på **användarens upplevelse**

Tester skrivs i naturligt språk

Inkluderar både tekniska och icke-tekniska intressenter

Använder **Given-When-Then** för teststruktur

Resulterar i "levande dokumentation" av beteendet

När Används TDD respektive BDD?

TDD passar bra för att utveckla enskilda komponenter, metoder och logik. Det är en bra metod för att säkerställa hög kodkvalitet och stabilitet i specifika delar av programmet.

BDD är idealiskt för att definiera och verifiera programvarans övergripande beteende från användarens perspektiv. Det används när man vill säkerställa att systemet fungerar enligt affärslogik och kundens krav.

Slutsats

TDD och **BDD** kompletterar varandra väl. TDD hjälper till att skapa robust och vältestad kod på detaljnivå, medan BDD säkerställer att systemet uppfyller de affärs- och användarkrav som ställs. Tillsammans bidrar dessa metoder till att skapa en mer komplett och pålitlig mjukvara, med fokus på både teknisk korrekthet och användarvärde.