BDD (Behavior-Driven Development)

BDD (z anglického **Behavior-Driven Development**, česky **vývoj řízený chováním**) je agilní softwarová metodika, která rozšiřuje principy **TDD** (Test-Driven Development) o **spolupráci mezi vývojáři, testery a obchodními zástupci (product owners)**. Cílem BDD je zajistit, že tým vyvíjí **správnou funkcionalitu**, která skutečně splňuje potřeby uživatelů a podnikání.

Na rozdíl od klasického testování, kde se testy píší až dodatečně, BDD začíná **definicí požadovaného chování systému před** samotným kódováním – a to v jazyce srozumitelném všem zúčastněným stranám.

Původ a principy

BDD zformuloval **Dan North** v roce 2003 jako reakci na běžné problémy TDD:

- Testy psané pouze vývojáři často neodpovídají skutečným obchodním požadavkům.
- Nedostatek komunikace mezi technickým a netechnickým týmem.

Klíčové principy BDD:

- **Společný jazyk** – tým používá „doménový jazyk“ (Ubiquitous Language) pro popis požadavků.
- **Příklady místo abstrakcí** – chování se specifikuje pomocí **konkrétních případů použití** (scénářů).
- **Automatizace specifikací** – tyto příklady lze spustit jako **živé testy**.
- **Zpětná vazba v reálném čase** – okamžitě vidíme, zda systém chová podle očekávání.

Rozdíl mezi TDD, ATDD a BDD

Metodika	Zaměření	Hlavní aktéři	Forma specifikace
TDD (Test-Driven Development)	Jednotkové testy, nízkoúrovňová logika	Vývojáři	Kód (např. JUnit, pytest)
ATDD (Acceptance Test-Driven Development)	Akcepční kritéria, shoda na požadavcích	Vývojáři, testeři, product owner	Tabulky, příklady
BDD	Chování systému z pohledu uživatele	Celý tým (včetně netechnických členů)	Přirozený jazyk (Gherkin)

BDD je tedy **nadstavbou TDD**, která přidává **komunikační a specifikační vrstvu**.

Gherkin – jazyk pro popis chování

BDD využívá formální, ale člověkem čitelný jazyk **Gherkin** k popisu scénářů. Každý scénář má strukturu:

- **Given** – počáteční stav („předpoklady“),
- **When** – akce uživatele nebo událost,

- **Then** – očekávaný výsledek.

Příklad:

```
Scenario: User adds item to cart
  Given I am on the product page for "Notebook XYZ"
  When I click the "Add to Cart" button
  Then the cart should contain 1 item
  And I should see a confirmation message
```

Tyto scénáře se ukládají do souborů s příponou `.feature` a slouží jako **živá dokumentace** i jako **spustitelné testy**.

Nástroje pro BDD

Nejznámějším nástrojem pro BDD je **Cucumber**, ale existují i alternativy:

Jazyk	Nástroj	Poznámka
——————-	——————————-	———-
Java	Cucumber-JVM	Nejrozšířenější implementace
JavaScript	@cucumber/cucumber, Playwright + Gherkin	Podpora pro moderní E2E testování
.NET (C#)	SpecFlow	Plně integrován do Visual Studia
Python	behave, pytest-bdd	behave je nejbližší Cucumberu
Ruby	Cucumber	Původní implementace

Tyto nástroje mapují řádky Gherkinu na tzv. **step definitions** – funkce v programovacím jazyce, které skutečně provádějí test (např. ovládají webový prohlížeč přes Selenium).

Výhody BDD

- **Lepší komunikace** – odstraňuje nedorozumění mezi obchodní a technickou stránkou.
- **Jasnější požadavky** – chování je popsáno konkrétními příklady, ne vágními popisy.
- **Živá dokumentace** – `.feature` soubory jsou vždy aktuální, protože selhání testu signalizuje nesoulad.
- **Rané odhalení chyb** – chyby v pochopení požadavků se projeví hned při psaní scénářů.
- **Automatizace akcepčních testů** – snadná validace, že systém dělá to, co má.

Nevýhody a rizika

- **Náklady na zavádění** – vyžaduje změnu myšlení a školení týmu.
- **Nadměrná automatizace** – BDD není vhodný pro všechny typy testů (např. jednotkové testy).
- **Technické detaily v Gherkinu** – pokud scénáře popisují „jak“ místo „co“, ztrácí se hlavní výhoda.
- **Údržba testů** – změna UI může vyžadovat úpravu mnoha step definitions.

Doporučené postupy (Best Practices)

- **Začněte workshopy „Three Amigos“** – společné schůzky vývojáře, testera a product ownera k definici chování.
- **Používejte přirozený jazyk** – scénář by měl číst jako uživatelský příběh.
- **Vyhněte se podmínkám a cyklům** v `.feature` souborech – raději vytvořte samostatné scénáře.
- **Nepřetěžujte Gherkin technickými detaily** – ty patří do kódu, ne do specifikace.
- **Integrujte BDD do CI/CD** – automaticky spouštějte scénáře při každém commitu.

BDD v praxi – typický workflow

```
1. **Definice požadavku** – product owner popíše uživatelský příběh.
2. **Týmový workshop** – společně se vytvoří konkrétní příklady (scénáře v Gherkinu).
3. **Automatizace** – vývojář naimplementuje step definitions.
4. **Vývoj kódu** – vývojář píše kód tak, aby všechny scénáře prošly.
5. **Ověření** – tým společně ověří, že chování odpovídá očekávání.
6. **Refaktoring** – kód se upravuje, ale scénáře zůstávají stejné (zajišťují regresní bezpečnost).
```

Související pojmy

- TDD
- Cucumber
- Gherkin
- Testování softwaru
- Agilní vývoj
- Uživatelský příběh
- Akcepční testování

Externí odkazy

- Dan North – „Introducing BDD“ (původní článek): https://dannorth.net/introducing-bdd/
- Oficiální Cucumber dokumentace: https://cucumber.io/docs/bdd/
- BDD v češtině – komunitní zdroje: https://www.agilecz.cz (vyhledávej „BDD“)

Viz také

- Behavior-Driven Development – FAQ
- Živá dokumentace
- Testovací pyramidy
- CI/CD