Testovací dluh

Testovací dluh (z anglického **test debt**) je forma **technického dluhu**, která vzniká, když tým:

- **nedodělá** potřebné testy,
- **píše špatné, nedostatečné nebo nespolehlivé testy**,
- nebo **odkládá automatizaci testů** kvůli časovému tlaku.

Na rozdíl od kódu, který „funguje“, testovací dluh **sníží důvěru v kvalitu systému**, zpomaluje vývoj a zvyšuje riziko regresí (návratu již opravených chyb).

Původ pojmu

Pojem vychází z analogie s **technickým dluhem** (Ward Cunningham, 1992), ale konkrétně se zaměřuje na **testovací strategii a pokrytí**. Testovací dluh se často akumuluje v raných fázích projektu, kdy tým „zrychluje vývoj“ tím, že testy ignoruje – což se později vyplácí náklady na ladění, manuální testování a nestabilitu systému.

Typy testovacího dluhu

Typ dluhu	Popis	Příklad
Chybějící testy	Neexistují testy pro kritické části kódu.	Kód pro platbu v e-shopu nemá žádné jednotkové testy.
Povrchní testy	Testy existují, ale pokrývají jen „happy path“, ne okrajové případy.	Test ověřuje jen úspěšné přihlášení, ne špatné heslo.
Nespolehlivé testy (flaky tests)	Testy občas selžou bez změny kódu.	E2E test selže, pokud je server pomalý.
Nepřehledné testy	Testy jsou špatně čitelné, bez jasných asercí.	Test používá magická čísla a žádné komentáře.
Špatně umístěné testy	E2E testy pro logiku, kterou by měly pokrýt jednotkové testy.	Ověření výpočtu DPH probíhá přes UI místo jednotkového testu.
Manuální testovací scénáře bez automatizace	Důležité scénáře se opakují ručně.	Každé nasazení vyžaduje 2 hodiny manuálního testování košíku.

Důsledky testovacího dluhu

- **Zpomalení vývoje**: Vývojáři se bojí měnit kód, protože nevědí, co porouchají.
- **Časté regrese**: Chyby se vrací, protože nejsou zachyceny testy.
- **Nízká důvěra v automatizaci**: Tým začne testy ignorovat nebo je označovat jako „označené k přeskočení“.
- **Zvýšené náklady na QA**: Roste potřeba manuálního testování.
- **Horší kvalita produktu**: Do produkce se dostávají chyby, které by mohly být zachyceny dříve.
- **Morální úpadek týmu**: Vývojáři ztrácejí radost z kódování v „nepřehledném“ systému.

Jak testovací dluh měřit?

Přestože není možné ho přesně kvantifikovat jako finanční dluh, lze použít následující metriky:

- **Test coverage** – procento kódu pokrytého testy (např. pomocí **JaCoCo**, **coverage.py**).

⚠ **Varování**: Vysoké pokrytí ≠ kvalitní testy! Může být 100 %, ale testovat jen „že kód běží“.

- **Počet flaky testů** – testů, které selhávají náhodně.
- **Poměr typů testů** – odchylka od testovací pyramidy (např. 80 % E2E, 5 % jednotkových).
- **Manuální testovací scénáře** – počet kritických scénářů bez automatizace.
- **Stáří neotestovaného kódu** – kolik dní/kommitů od posledního testovacího commitu.

Jak se testovacímu dluhu vyhnout?

- **Testuj od prvního dne** – i malý projekt by měl mít základní jednotkové testy.
- **Dodržuj testovací pyramidu** – většina testů by měla být jednotkových.
- **Nepřijímej PR bez testů** – zaved politiku „code coverage gate“ v CI.
- **Refaktoruj testy** – testy jsou kód; aplikuj na ně stejná pravidla (DRY, čitelnost).
- **Pravidelně odstraňuj flaky testy** – buď je oprav, nebo odstraň.
- **Plať dluh postupně** – při každé úpravě kódu přidej testy pro danou komponentu („boy scout rule“).

Jak testovací dluh splácet?

- **Prioritizuj kritické části** – začni testováním modulů s nejvyšším rizikem (např. platby, autentizace).
- **Použij „stratégii krabičky“** – pokud nemůžeš testovat jednotlivé funkce, otestuj celý modul jako „černou krabičku“.
- **Zautomatizuj manuální scénáře** – začni s nejčastějšími nebo nejrizikovějšími.
- **Vytvoř „dluhový backlog“** – eviduj testovací dluh jako uživatelské příběhy v nástroji pro správu úloh (Jira, Trello).
- **Naplánuj „sprint pro kvalitu“** – občas věnuj celý sprint odstranění dluhu.

Testovací dluh vs. technický dluh

Aspekt	Technický dluh	Testovací dluh
——–	————	————
Zaměření	Kvalita kódu, architektura	Kvalita a přítomnost testů
Důsledek	Těžší údržba kódu	Nízká důvěra v chování systému
Měřitelnost	Statická analýza (SonarQube)	Coverage, počet testů, flakiness
Splácení	Refaktoring, přepis	Psaní testů, zlepšování testovací strategie

 Testovací dluh je **často součástí technického dluhu**, ale vyžaduje specifický přístup.

Související pojmy

- Technický dluh
- Testovací pyramida
- Jednotkový test
- Flaky test
- CI/CD
- Testovací coverage

Externí odkazy

- Martin Fowler – Test Debt: https://martinfowler.com/articles/is-quality-worth-cost.html
- Google Testing Blog – Flaky Tests: https://testing.googleblog.com/2017/04/where-do-our-flaky-tests-come-from.html
- IEEE – „The Real Cost of Test Debt“ (akademický článek)

Viz také

- Testovací strategie
- QA v agilním týmu
- Refaktoring
- Behavior-Driven Development