Technický dluh (Technical Debt)

Technický dluh funguje podobně jako finanční půjčka. Rychlejší vývoj na začátku vám „půjčí“ čas, ale v budoucnu jej budete muset splácet i s úroky. Úrokem je v tomto případě zvýšené úsilí a čas, který musíte vynaložit na úpravu nebo opravu onoho nekvalitního kódu při každé další změně v systému.

Jak technický dluh vzniká?

Dluh nemusí vznikat jen leností, často jde o strategické rozhodnutí nebo důsledek evoluce projektu:

- **Tlak na termín:** „Musíme to vydat zítra, opravíme to později.“ (Záměrný dluh)
- **Nedostatek znalostí:** Tým v době psaní kódu nevěděl, jak daný problém vyřešit lépe. (Neúmyslný dluh)
- **Zastarání (Bit rot):** Technologie se vyvíjejí a kód, který byl před pěti lety špičkový, je dnes považován za zastaralý a neefektivní.
- **Nedostatek dokumentace a testů:** Chybějící testy ztěžují budoucí změny a zvyšují riziko chyb.

Důsledky nespláceného dluhu

Pokud se dluh pravidelně nesplácí pomocí refaktorování, nastávají tyto problémy:

```
1. **Zpomalení vývoje:** Každá nová funkce trvá déle, protože se
programátoři musí "prosekávat" nekvalitním kódem.
2. **Větší chybovost:** Jedna oprava na jednom místě rozbije tři jiné věci
(tzv. fragilní kód).
3. **Frustrace týmu:** Kvalitní vývojáři neradi pracují v "nepořádku", což
může vést k jejich odchodu.
4. **Konečný bod (Bankruptcy):** Dluh je tak vysoký, že už není možné přidat
žádnou funkci a projekt se musí přepsat od nuly.
```

Jak dluh spravovat?

Technický dluh není vždy špatný – někdy je nutné ho přijmout, abyste získali konkurenční výhodu na trhu. Klíčem je jeho aktivní správa:

- **Identifikace:** Označujte v kódu místa s dluhem (např. komentářem *TODO: refactor''). *

Pravidlo skautů: *„Zanechte kód vždy o kousek čistší, než jste ho našli.“ * **Alokace času:** Vyhraďte např. 20 % času v každém vývojovém cyklu na splácení dluhu (refaktorování). * **Automatizace:** Používejte nástroje pro statickou analýzu kódu, které na dluh samy upozorní. —- ====== Rozdíl: Technický dluh vs. Špatný kód ====== Pamatujte, že **nepřehledný a nefunkční kód** napsaný z nedbalosti není technický dluh, ale prostě špatná práce. Skutečný technický dluh je vědomý kompromis mezi rychlostí dodání a kvalitou návrhu. ^ Metrika ^ Technický dluh ^ Špatný kód ^ | **Záměr** | Strategický kompromis. | Nedbalost nebo neschopnost. | | **Dopad** | Nutnost budoucího úklidu. | Okamžitá nefunkčnost a nestabilita. | | **Řešení** | Refaktorování. | Kompletní přepsání. | —- Související pojmy: Refaktorování, DRY princip, SOLID principy, Unit Testing, Agilní vývoj, Code Smells.