**Testování, Unit testování a dokumentace zdrojového kódu**

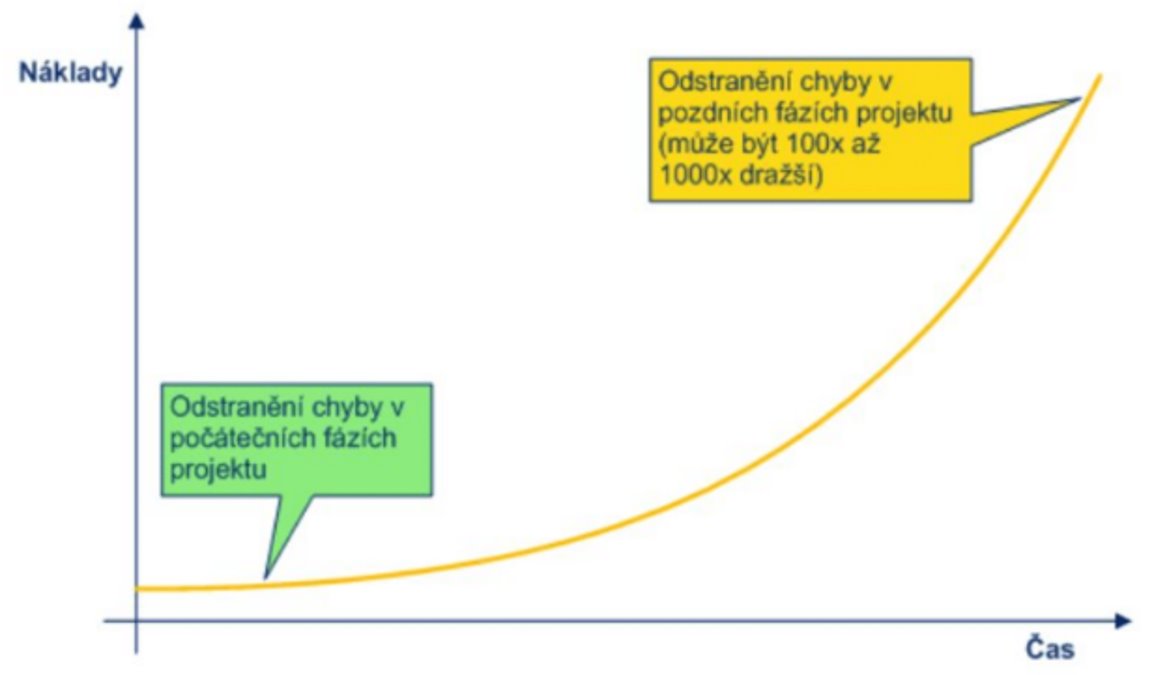
**Co je testování?**

* **Technický výzkum kvality produktu nebo služby účelem poskytnout informace všem zainteresovaným**
* Ověření, že program nebo systém dělá to, co se od něj očekává
* Zkoušení/simulace provozu SW za účelem hledání chyb.​

1. Testování ukazuje přítomnost defektu​
   1. Testování může ukázat, že jsou defekty přítomny, ale nemůže dokázat, že v sw nejsou žádné defekty. Testováním snižujeme pravděpodobnost, že v sw zůstanou neobjevené defekty.​
2. Vyčerpávající testování je nemožné​
   1. Testovat všechno není realizovatelné (všechny kombinace vstupů a výstupní podmínky), s výjimkou triviálních případů.​
3. Včasné testování​
   1. Testování by mělo začít co nejdříve je to možné v životním cyklu vývoje sw nebo systému a mělo by být zaměřeno na definované cíle​ ​
4. Shlukování defektů​
   1. Jedna chyba na sebe „nabaluje“ další.​
5. Pesticidní paradox​
   1. Jakmile stejné testy opakujeme neustále dokola, časem stejnou sadou testů nenajdeme žádné defekty. ​ Je potřeba testovací případy pravidelně revidovat, upravovat a psát nové odlišné testy pro jiné části sw nebo systému, abychom mohli odhalit nové defekty​
6. Testování je závislé na kontextu​
   1. Testování je vykonáváno odlišně pro rozdílné souvislosti/významy.​ Např. sw kritický z pohledu bezpečnosti se testuje jinak než stránka eshopu​
7. Falešná představa o neexistenci omylů​
   1. Nalezení a zaevidování defektu nepomůže, pokud systém/sw není použitelný a nesplňuje potřeby/očekávání uživatelů

**Proč testovat?**

Podcenění testování vede k zvýšení nákladů na vývoj.​



**Testovací cyklus**

**Plánování testů:​**

Definice projektu, co kontrolovat a jak vyhodnocovat​ Identifikace testovacích požadavků, strategie testování, zdrojů pro testy a příprava testovacího plánu.​ ​

**Analýza a příprava testů​**

Návrh jednotlivých testů (test cases) vůči požadavkům​ Identifikace a příprava potřebných testovacích dat.​ ​

**Provedení a vyhodnocení testů​**

Provedení testů a zaznamenání jejich výsledků.​ Analýza výsledků a vyhodnocení, zda nedošlo k chybě.​ ​

**Sledování defektů**​

Průběžná kontrola defektů

Graphical user interface, diagram

Description automatically generated

**FURPS – Model kvality testování​**

**F**unctionality - Funkčnost​

* Zda jsou poskytovány uvedené funkčnosti
* Přístupová práva k funkcionalitě

**U**sability – Použitelnost​

* Hodnocení z pohledu koncového uživatele
* Dojem uživatele, dokumentace, školící materiály

**R**eliability – Spolehlivost​

* Hodnocení frekvence selhání
* Odolnost vůči selhání (datová a funkční strana)

**P**erformance – Výkon​

* Hodnocení rychlosti odezvy systému
* Zpracování byznys aktivit
* Vytížení zdrojů (síť, OS, HW), rozložení zátěže na jednotlivé komponenty

**S**uportability – schopnost být udržována

* Hodnocení
  + Nasaditelnosti
  + Udržovatelnosti
  + Konfigurovatelnosti

**Rozděnelní testů podle metody**

**White box**

* Známe principy fungování testovaného softwaru
* Známe strukturu programu, datový model, použitý kód

**Black box**

* Není třeba znát princip nebo strukturu fungování
* Známe vstupy a očekávané výstupy ​
* Vycházíme z požadavků na systém a ověřujeme, zda je software splňuje.​
* Testovací scénáře
  + Na základě analýzy specifikace nebo dokumentace
  + Neznáme vnitřní strukturu systému​

**Grey box**

* Kombinace

**Programování a testování?**

Automatizované testování je typicky tvořeno software testerami nebo Quality assurance inženýrama

* Automatizované testy musí být vhodné, nemohou být použity na všechno „testování“
* Pomocí frameworků na různé programovací jazyky jdou použít pro testování
* Testování FRONTENDU
  + Pomocí dalších frameworků
    - Simulace kliknutí s různýma parametrama(FORCE CLICK)
    - Screenshoty
    - Simulace různých BROWSERŮ
  + Vyhledání elementů pomoci XPATH (F12 and ctrl+F)

**Unit testy**

* Automatizované testy jednotlivých komponentů aplikace na úrovni objektů a tříd
* První fáze testování, která je prováděna programátorem a ve fázi „VÝVOJ“
  + Ověření, zda změněný nebo nový kód nepadá do chyby
  + Detekuje a zamezuje výskyt základních chyb v pokročilejších fázích
* Izolované od kódu
* Mohou sloužit jako dokumentace = ukazovat co má kód správně dělat

Příklad Unit testu na metodu pro sčítaní dvou čísel v PYTHONU

Text

Description automatically generated

**Dokumentace zdrojového kódu**

Psaní a vysvětlení jak funguje jednotlivé části kódu, co mají dělat a jak použít

**K čemu slouží?**

* Ulehčení pochopení kódu pro jiné programátory
  + Důležité na velkých projektech
* Vylepšení kvality kódu
* Zjednodušuje budoucí úpravy nebo navázání na kód