Vývoj řízený testy
Test Driven Development

Richard Salač, Ondřej Lanč

Fakulta jaderná a fyzikálně inženýrská
České vysoké učení technické v Praze

23. - 30. 10. 2012

Obsah

Definice testování

Testování

Testováním rozumíme cílené spouštění vyvíjeného programu s předem definovanými provozními podmínkami a očekáváními jeho chování za účelem zlepšení jeho kvality.

Kvalita softwaru

FURPS

- **F - funkčnost (bezpečnost)**
- **U - použitelnost (z pohledu uživatele)**
- **R - spolehlivost (chybovost, přesnost)**
- **P - výkon (rychlost, odezva, vytížení HW)**
- **S - Rozšiřitelnost (údržba, testovatelnost, monitoring)**

Kvalita softwaru

FURPS+

- Implementace - návrh aplikace, závislost na architektuře
- Rozhraní - připojení na zařízení a externí systémy
- Operační systémy - kompatibilita a nutnost úprav
- Obchodní a právní aspekty - licence, konkurenceschopnost

Vývojářské testování

Vývojář

- Píše program tak, aby fungoval
- Předem známý konec

Tester

- Píše test tak, aby program nefungoval
- Nekonečný příběh

Vývojářské testování

Definice chyby

- Program nedělá něco, co by dělat měl
- Program dělá něco, co by dělat neměl
- Program dělá něco, o čem se specifikace nezmiňuje
- Program nedělá něco, o co v dokumentaci není, ale mělo by být

Vývojářské testování

Vývojářské testování

- testy funkčnosti
- testy spolehlivosti

Vývojářské testování

Funkční testování

- black-box testing
- Akceptační testy

Strukturní testování

- white-box testing
- Testování jednotek (Unit testing)
- Testování komponent
- Integrační testy

Rozsah testování

Adresář

- Jméno - max. 20 znaků, abeceda 42 prvků
- Adresa - max. 20 znaků
- Telefonní číslo - 9 znaků, 10 možných číslic

Celkem možností

10^{74} možností

Rozsah testování

Adresář

- Jméno - max. 20 znaků, abeceda 42 prvků
- Adresa - max. 20 znaků
- Telefonní číslo - 9 znaků, 10 možných číslic

Celkem možností

10^{74} možností

Odhad chyb

Analýza mezí

- n vs. $n - 1$
- $>=$ vs. $>$

Složené meze

- Vznikající na základě více faktorů
- Tvořené za běhu programu

Odhad chyb

Třídy dobrých dat

- Normální případy - běžně očekávané hodnoty
- Minimální normální data
- Maximální normální data
- Kompatibilita se starými daty

Třídy špatných dat

- Příliš málo dat (žádná data)
- Příliš mnoho dat
- Nesprávný typ
- Neplatná data
- Nesprávná velikost dat

Vývojářské testování

Tipy por usnadnění

- Rozdělení na třídy ekvivalence
- Ruční kontrola

Minimální rozsah testování

- Není možné testovat vše
- Pokryjeme alespoň všechny cesty algoritmem

Úplné pokrytí kódu

- Za přímý průchod přičteme 1
- Za každé větvení přičteme 1
- Za každou alternativu přepínače přičteme 1

Minimální rozsah testování

Příklad

```
void znamenko(double cislo)
{
        if( cislo > 0 )
            printf("kladne");
        else
            printf("zaporne");
}
```

Testování

Zdrojový kód testovaného programu je neměnný. Žádný test nesmí měnit původní kód ani pro svou práci tyto změny vyžadovat.

Testování

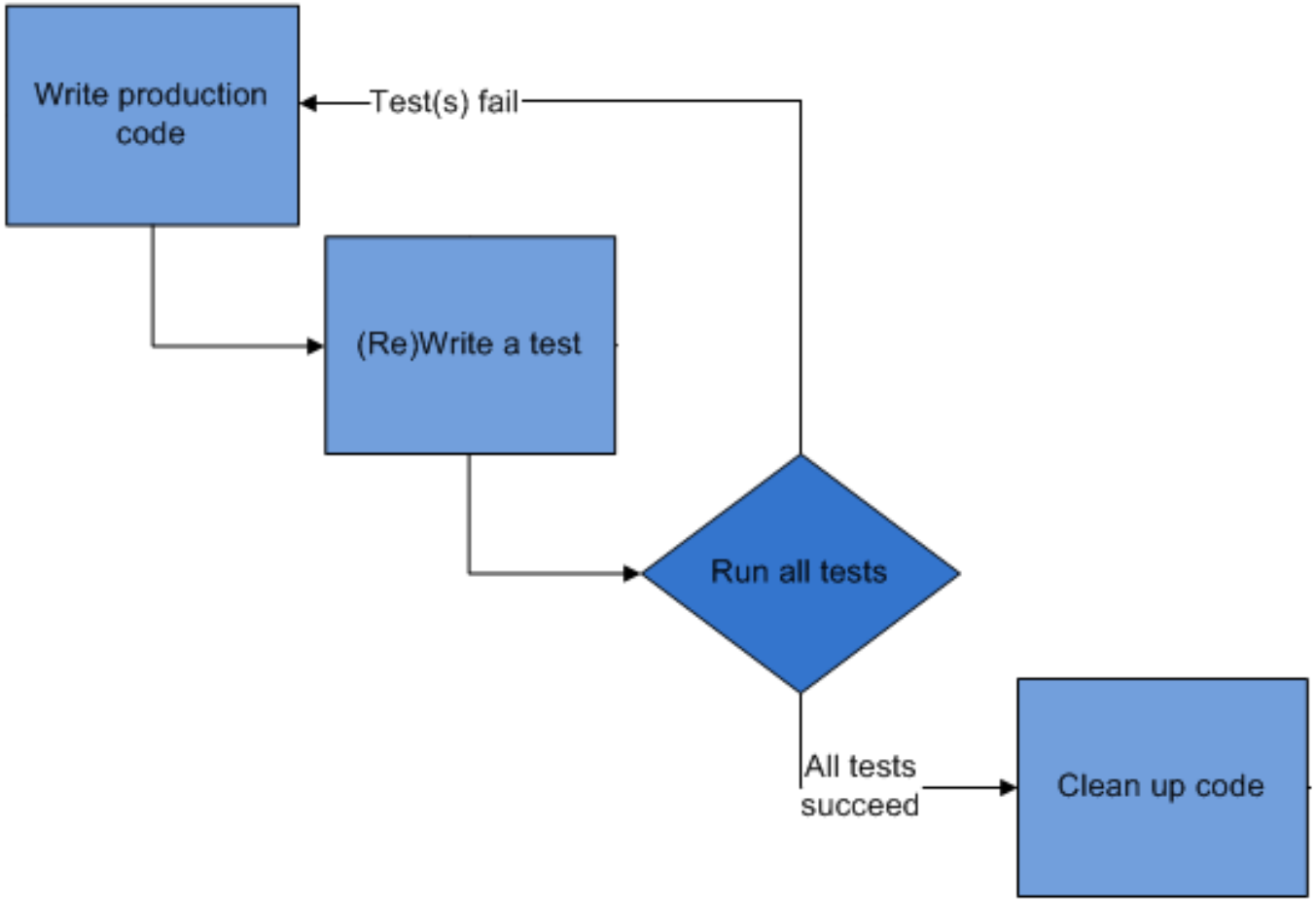
Jak napsat test

- Cíl testu
- Vstupní testovací data
- Očekávaný výsledek testu

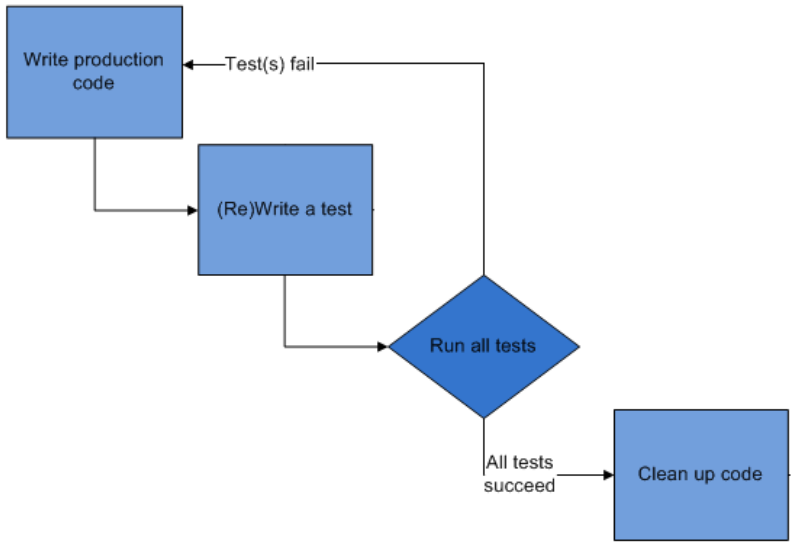
Průběh testu

- Definice proměnných testu a inicializace pomocných objektů
- Inicializace *SUT*
- Spuštění testu
- Ověření testovacích podmínek

Test-last


Write production code
Test(s) fail
(Re)Write a test
Run all tests
All tests succeed
Clean up code

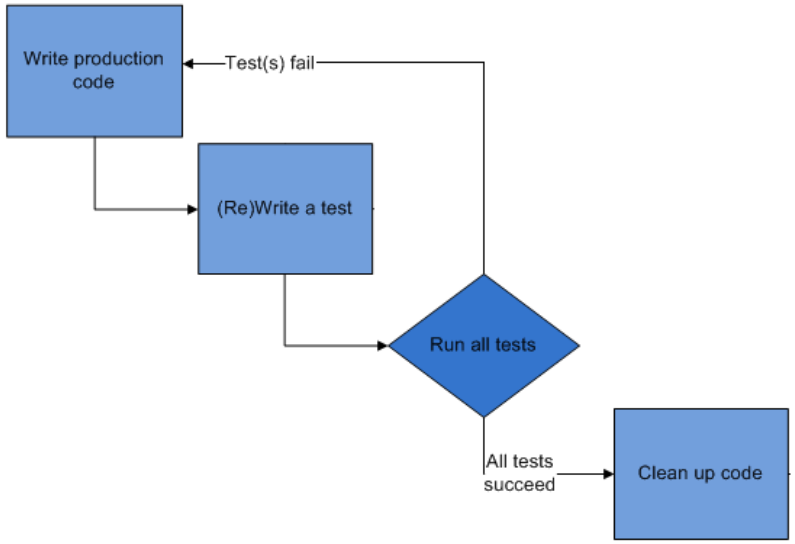
Test-last



Kódování

- Implementace nové funkcionality
- Vychází z analýzy případů užití (use case)

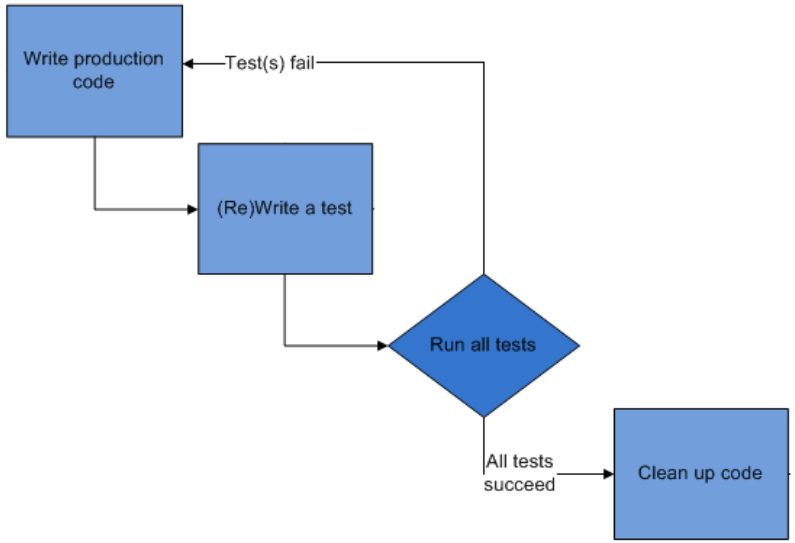
Test-last



Napsat test

- Ověření funkcionality
- Pokrytí kódu

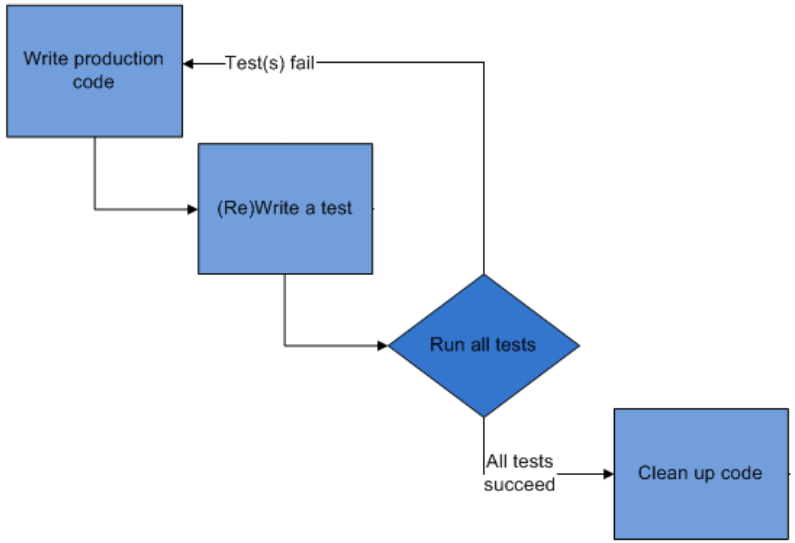
Test-last



Spuštění testů

- Úspěšný test odhalí chybu
- Nalezli jsme chybu - zpět ke kódování

Test-last



Čištění kódu

- Refactoring

Test-last

Testy

- Úspěšný test odhalí chybu
- Cíl: způsobit chybu
- Dokumentace API

Na co dát pozor

- Pokrytí kódu
- Pokrytí požadavků

Test-last

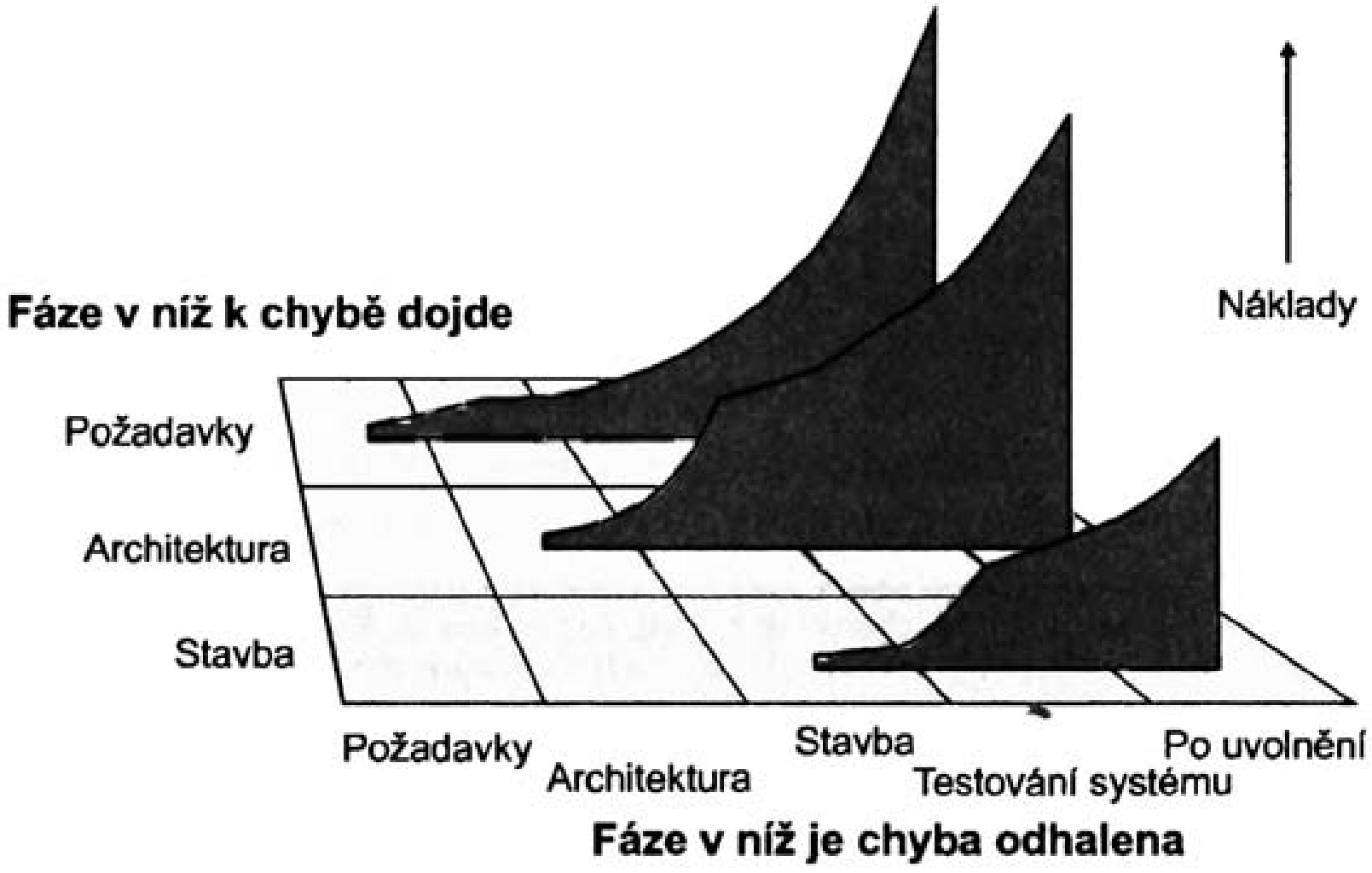
Testy

- Úspěšný test odhalí chybu
- Cíl: způsobit chybu
- Dokumentace API

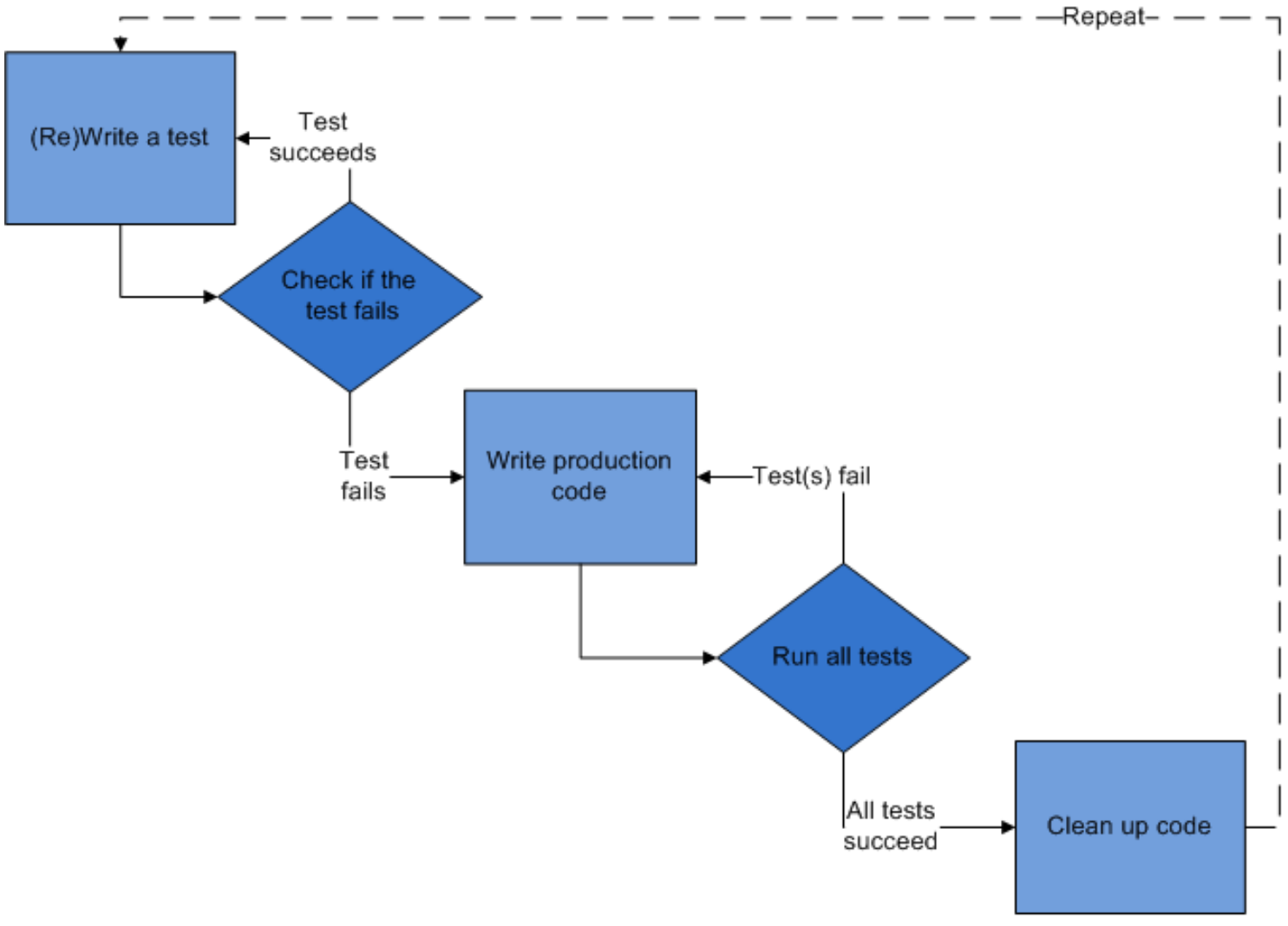
Na co dát pozor

- Pokrytí kódu
- Pokrytí požadavků

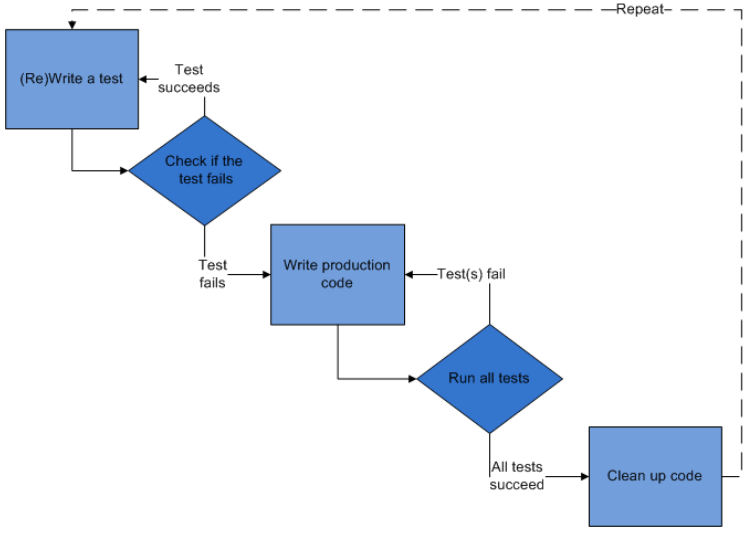
Náklady na odstranění chyby



Test-first


Repeat
(Re)Write a test
Test succeeds
Check if the test fails
Test fails
Write production code
Test(s) fail
Run all tests
All tests succeed
Clean up code

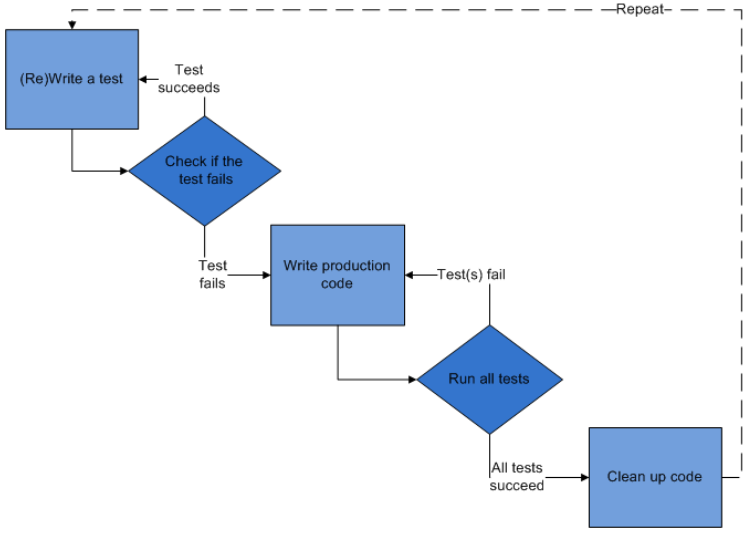
Test-first



Napsat test

- Nová funkcionalita - nové testy
- Use case jako podklad pro napsání testu

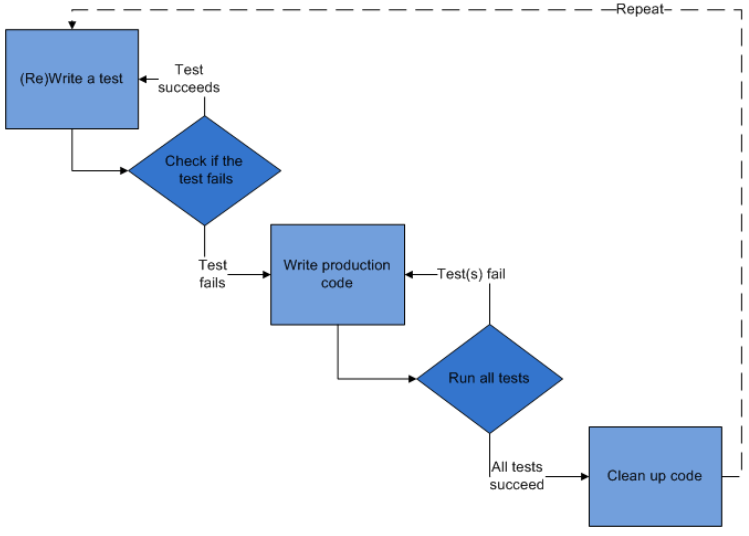
Test-first



Spustit testy

- Ověření funkčnosti testu
- Napoprvé selže
- Pokud ne - chybný test nebo požadavek

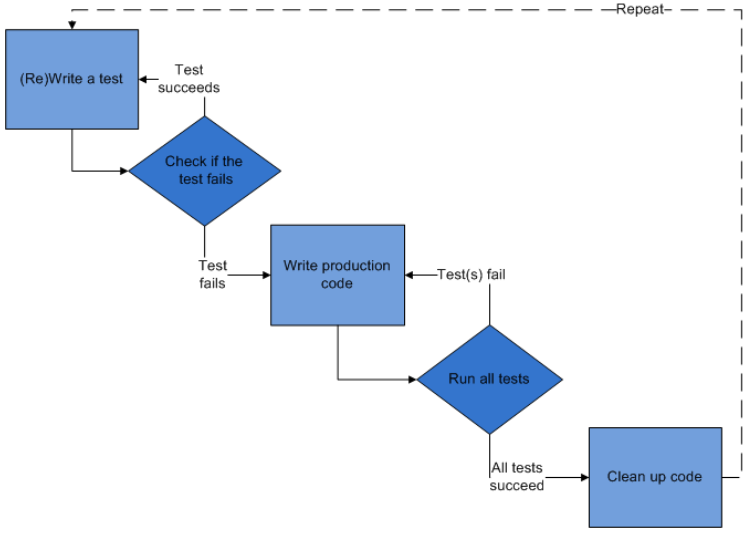
Test-first



Kódování

- Implementace nové funkcionality
- Přidáme jen to, co je nezbytně nutné ke splnění testů

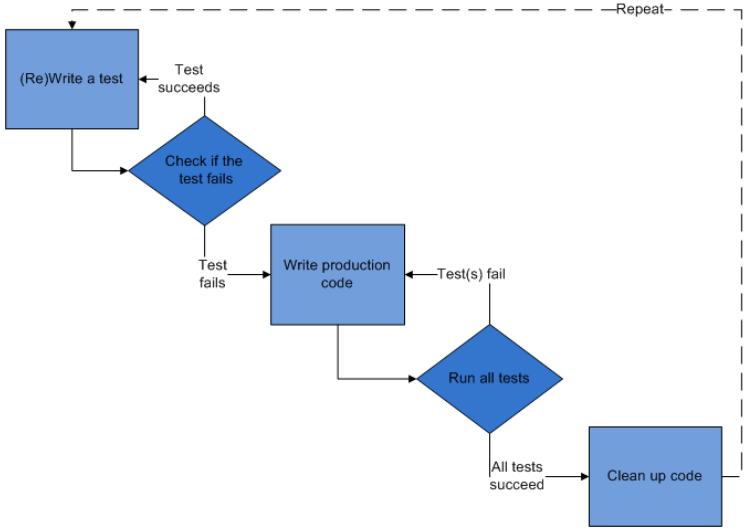
Test-first



Testování

- Spuštění sady testů
- Objevila-li se chyba, vracíme se k implementaci

Test-first



Čištění kódu

- Refactoring

Test Driven Development

Testy

- Specifikace rozhraní
- Prostředek návrhu aplikace
- Pokrytí kódu zajištěno z podstaty TDD

Na co dát pozor

- Pokrytí požadavků

Test Driven Development

Testy

- Specifikace rozhraní
- Prostředek návrhu aplikace
- Pokrytí kódu zajištěno z podstaty TDD

Na co dát pozor

- Pokrytí požadavků

Test Driven Development

Test Driven Development

- Malé iterace
- Vhodné propojení s CVS
- Snadná detekce problémů a návrat k funkční verzi
- Rychlejší vývoj
- Lepší software

Srovnání

Klasický přístup

- Nejdříve vzniká samotná aplikace
- Podkladem pro testy je napsaný kód
- Nutno sledovat pokrytí kódu

Test Driven Developement

- Nejdříve vznikají testy
- Podkladem pro testy je specifikace
- Pokrytí kódu zajištěno z podstaty TDD
- Lepší kód a kratší vývoj

Literární zdroje

- Beck K.: Test Driven Development: By Example, Addison Wesley - Vaseem, 2003
- Knaer, Cem, Falk, Jack, Nguyen, Hung, Quoc: Testing Computer Software, Wiley Computer Publishing, 1999
- *http : //www.agiledata.org/essays/tdd.html*
- *http : //en.wikipedia.org/wiki/Test − driven_development*