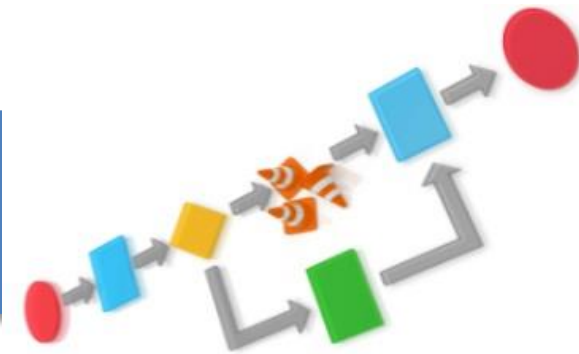


Návrh softwarových systémů - úvod, motivace

Jiří Šebek

Návrh softwarových systémů
(B6B36NSS)



```
public final void onSensorChanged(SensorEvent event)
{
    m_flightIntensity = event.values[0];
    m_etAmblight.setText("" + m_flightIntensity + " lx");
}
...
...resume()
...light, ...NORMAL);
```

Obsah

- Motivace
- Integrace s ostatními obory SI
- Modely, metodiky SI
- Verzování SW



Úvod

- Motivace SI
 - Velké projekty
 - Údržba, škálovatelnost, modifikovatelnost
 - Lidé přicházejí a odcházejí, ale firma zůstává
 - Udržování know-how
 - Předcházení chybám
 - Komunikace v týmu je základ úspěchu
 - A další



Cíl předmětu

- Podobné jako u jiných SI předmětů
 - Spolupracovat v týmu
 - Prezentovat výsledky práce
 - Komunikovat
 - Samostatně studovat
 - Analyzovat projekt a následně řešit problémy
 - ...



Cíl předmětu

- Specifické
 - Seznámení s SI architekturami a jejich návrhem
 - Různé druhy implementací
 - Umět rozeznat správnou analýzu a návrh architektur
 - ..



Organizace

- Stránky :
 - <https://cw.fel.cvut.cz/wiki/courses/b6b36nss/start>
- Požadavky pro absolvování:
 - Předmět je ukončen zápočtem a zkouškou (viz stránky předmětu)



Organizace

- **Zápočet:**
- Odevzdat semestrální úlohu
- Progress Presentation(~ 7.týdnu) → 5b
- Domácí úkoly → 10b (odevzdání je povinné i po deadlinu)
- Odevzdání na gitlab(~ 14.týdnu) → 20b (min 10b)
- midterm testu v 6. týdnu → 5b
- Účast na cvičeních je povinná. Tolerovány budou nejvýše 3 neomluvené absence.

Organizace

- **Zkouška:**

- Zkouška se bude skládat ze 2 částí:
- návrhová → 30b (min 15b)
- analytická → 30b (min 15b)
- K účasti na zkoušce je nutné mít zápočet.
- Zkouška studentů kombinovaného studia bude probíhat za stejných podmínek, jako zkouška studentů prezenčního studia.



Softwarové inženýrství

- Softwarové inženýrství je systematický, disciplinovaný a kvalifikovaný přístup k vývoji, tvorbě a údržbě softwaru.

definice IEEE, 1993

- Důležitý proces pro střední a velké projekty (malé projekty se dají odevzdat bez nějakého procesního plánování, ale mají pak problémy s udržitelností, škálovatelností apod.)

Softwarové inženýrství

- Využívá modely, metodiky...
- Model
 - popisuje fáze životního cyklu softwaru a vztahy mezi nimi
- Metodika
 - určuje postupy každé fáze (co se má dělat)
 - určuje také vstupy a výstupy
 - Dělení :
 - Tradiční
 - agilní

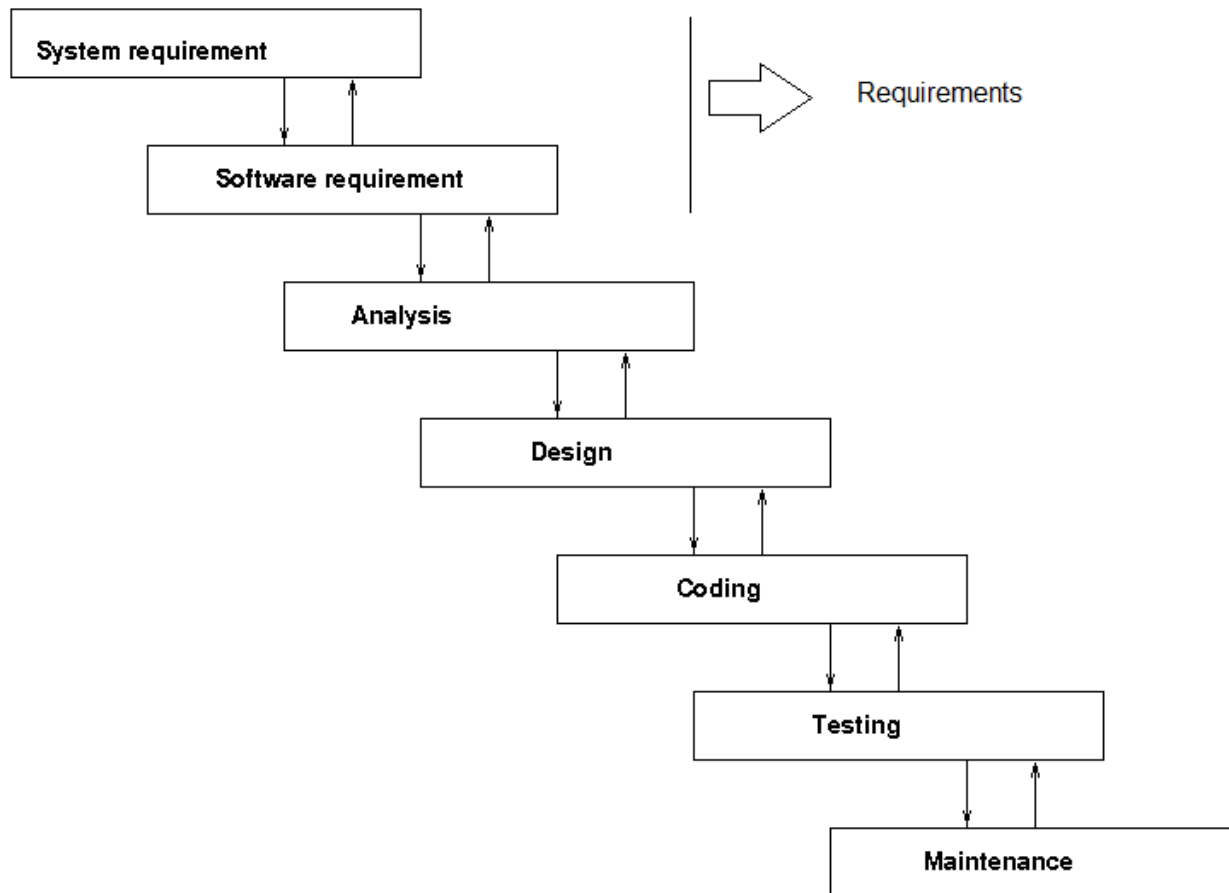
Softwarové inženýrství

- Využívá modely, metodiky...
- Model
 - popisuje fáze životního cyklu softwaru a vztahy mezi nimi
- Metodika
 - určuje postupy každé fáze (co se má dělat)
 - určuje také vstupy a výstupy
 - Dělení:
 - Tradiční: čas + cena = funkcionalita
 - UP
 - Agilní: funkcionalita => cena + čas
 - XP

Modely

- Vodopád (sekvenční)
- Spirála (iterativní)
- Prototype (iterativní)
- Inkrementální (agilní)
- Paralelní (agilní)

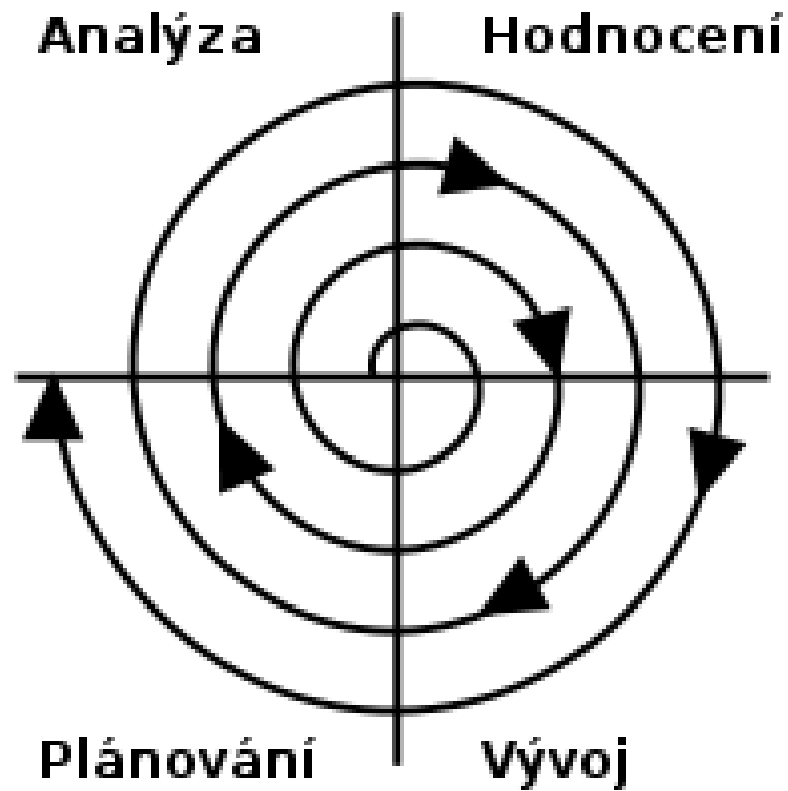
Model vodopád



Model vodopád vlastnosti

- Jeden z nejznámějších modelů je vodopád (existují různé adaptace)
- Jedná se o posloupnost fází, které na sebe navazují
- V základu tohoto modelu se postupuje vždy o jeden krok dopředu nebo dozadu
- Nevýhody:
 - Velký časový úsek mezi zadáním a prvním releasem
 - definice všech požadavků už zezáčátku vývoje
 - Při změně zadání je potřeba schválit výstupy všech fází

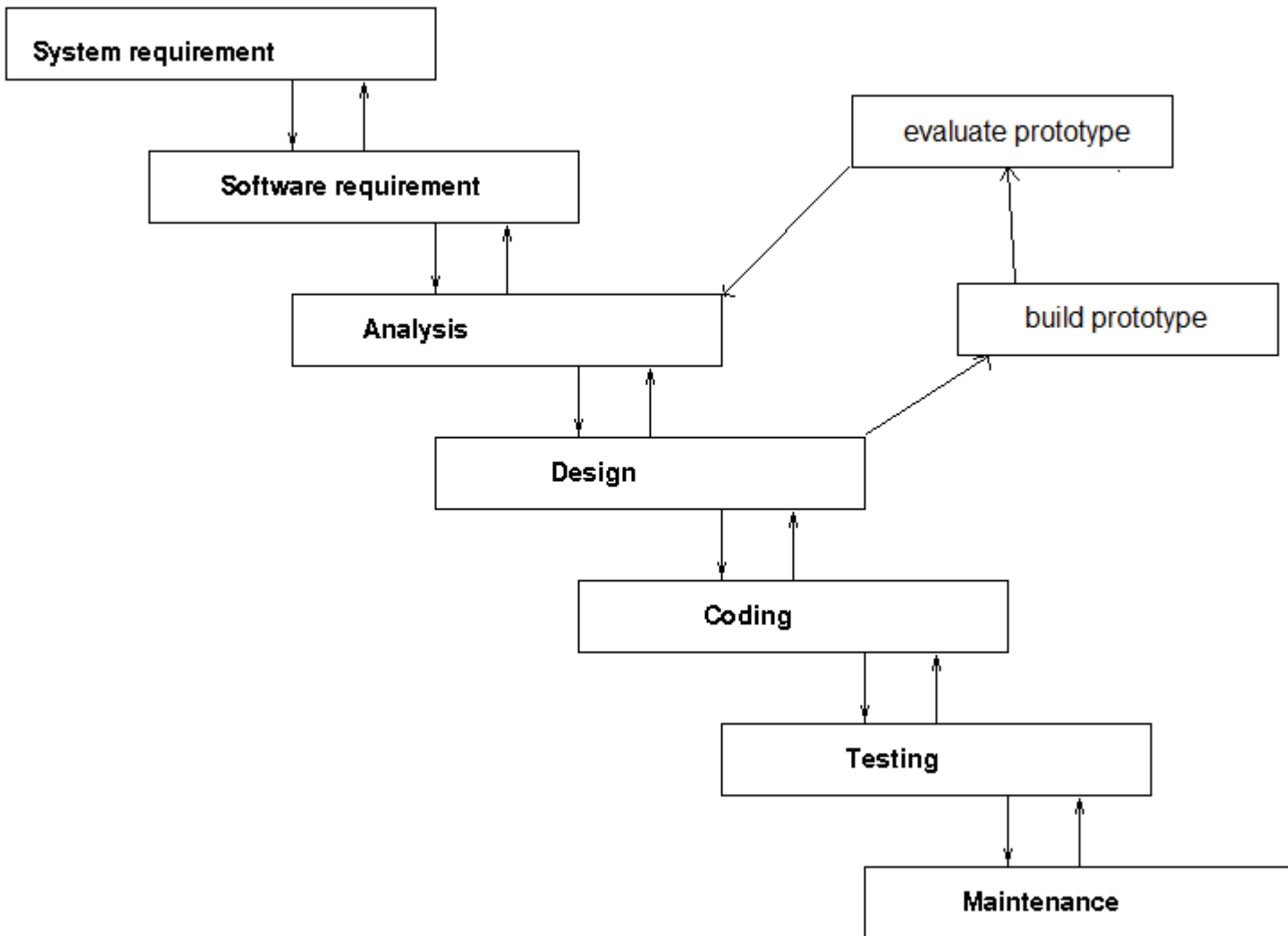
Model spirála



Model spirála vlastnosti

- cyklické opakování jednotlivých kroků vývoje
- iterativní typ modelu
- Je odpovědí na nevýhody vodopádu
- Důležitá vlastnost modelu je analýzu rizik v každém cyklu

Model prototype



Model prototype popis

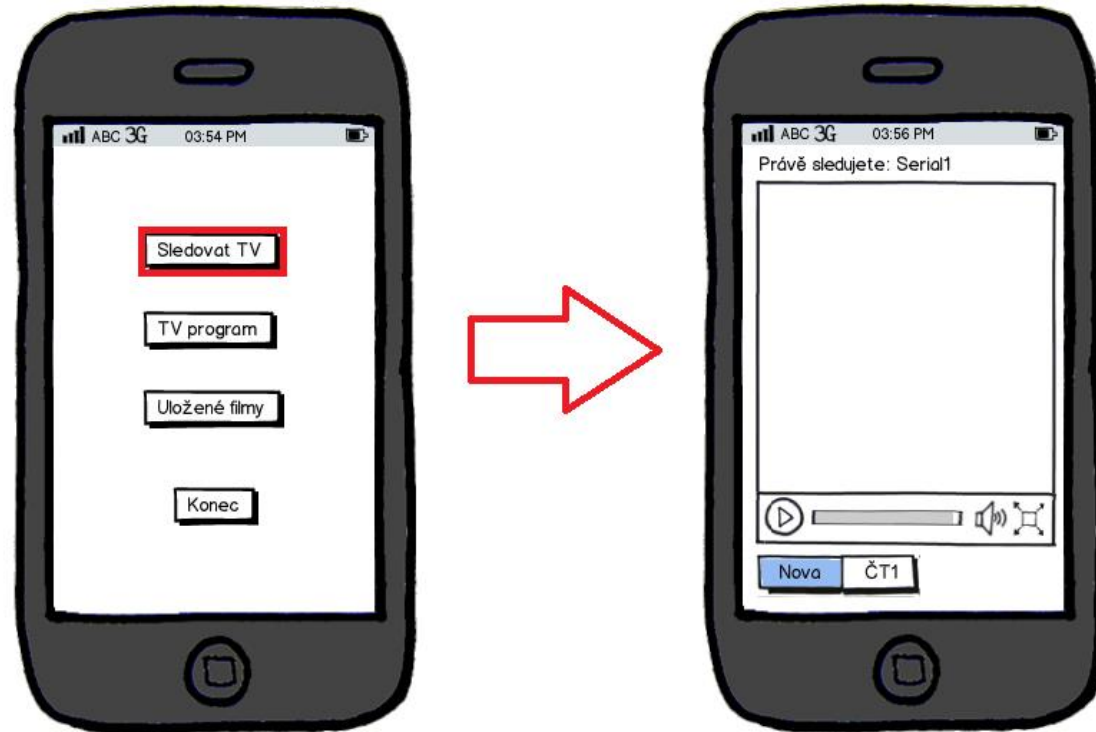
- Prvotní sběr a analýza požadavků.
 - Rychlý návrh.
 - Tvorba prototypu.
 - Vyhodnocení prototypu zákazníkem.
 - Pokud je zákazník spokojen, posun
 - Zpracování připomínek zákazníka a posun
 - Specifikace systému na základě prototypu.
 - Návrh a implementace systému.
-
- ```
graph TD; A[Rychlý návrh.] --> B[Tvorba prototypu.]; B --> C[Vyhodnocení prototypu zákazníkem.]; C --> D[Pokud je zákazník spokojen, posun]; D --> E[Zpracování připomínek zákazníka a posun]; E --> F[Specifikace systému na základě prototypu.]; F --> G[Návrh a implementace systému.]; F --> A;
```

# Model prototype

- U UI softwaru dělíme prototypy na :
  - Low fidelity
  - High fidelity

# Model prototype

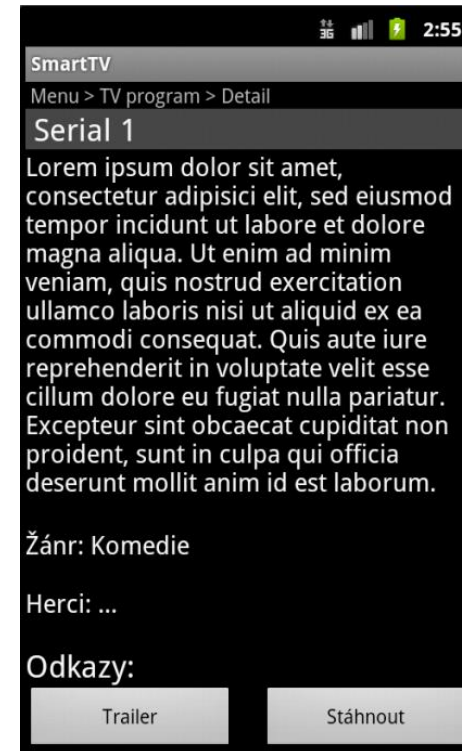
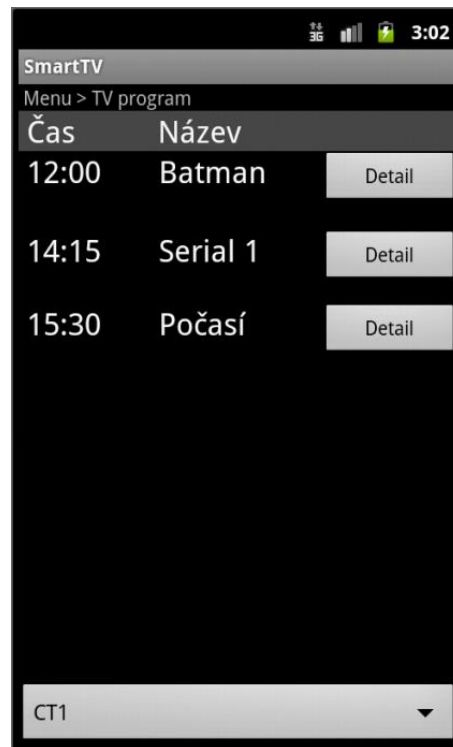
- U UI softwaru dělíme prototypy na :
  - Low fidelity



created with Balsamiq Mockups - [www.balsamiq.com](http://www.balsamiq.com)

# Model prototype

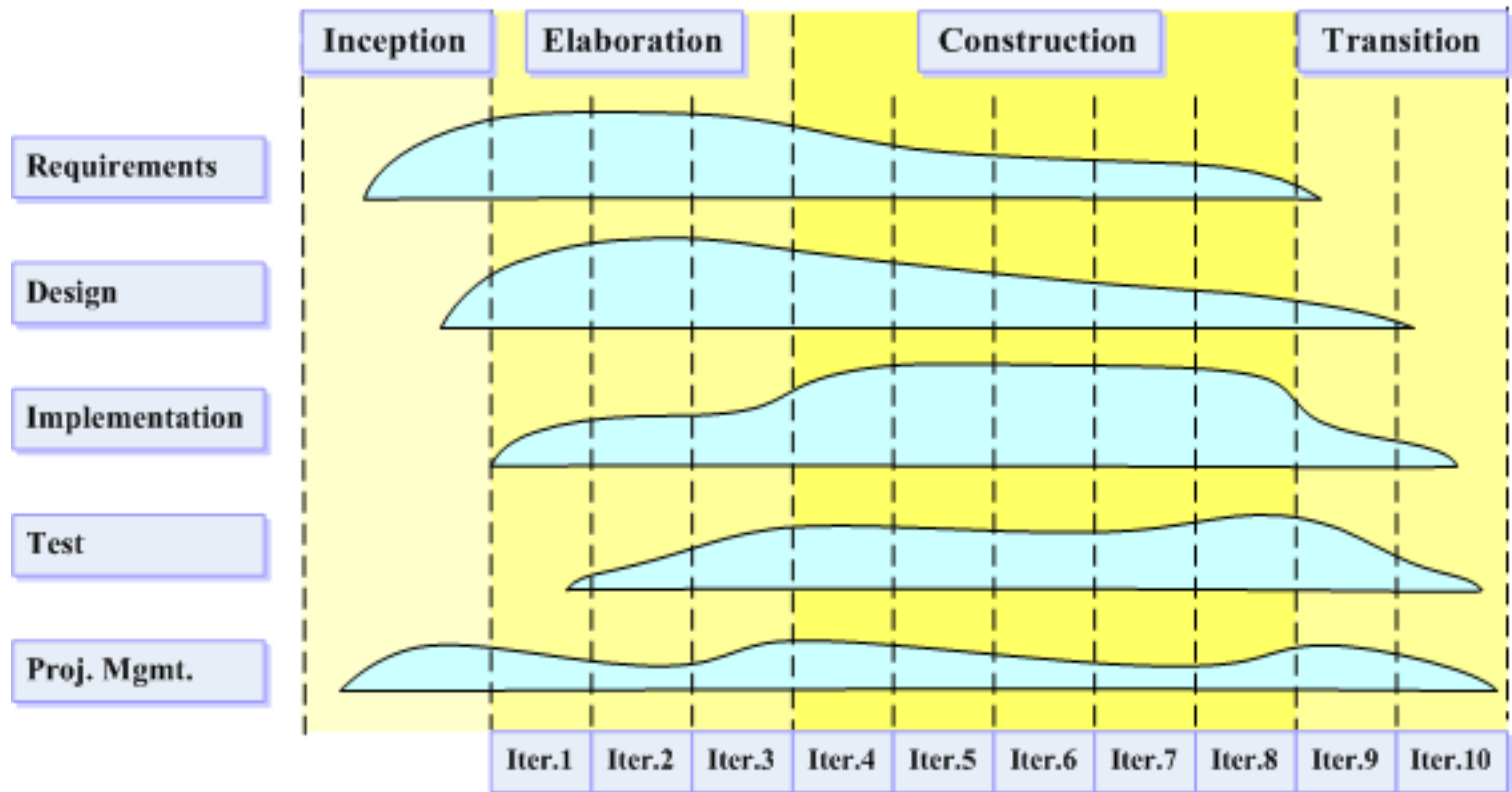
- U UI softwaru dělíme prototypy na :
  - High fidelity



# Metodiky

- Tradiční
  - Funkcionalita → čas + peníze
- Agilní
  - čas + peníze → Funkcionalita
- Příklady : Unified Proces (UP), Rational Unified Proces (RUP) – *placená verze podobná UP*, Extrémní programování (XP), SCRUM, test driven development (TDD), Model driven development (MDD)

# UP



# UP

- 4 fáze:
  - zahájení
  - projektování
  - realizace
  - předání
- Každá fáze obsahuje:
  - sběr/revize požadavků
  - analýza
  - návrh
  - implementace



# XP


- 4 hodnoty:
  - Komunikace
  - Jednoduchost
  - Zpětná vazba
  - Odvaha



# XP

- 12 praktik:
  - Business praktiky
    - Plánování hry
    - Zákazník na pracovišti
    - Vydávání malých verzí
    - Metafora
  - Týmové praktiky
    - Párové programování
    - Společné vlastnictví kódu
    - Standardy kódu
    - Udržitelné tempo = 40 hodin týdně
  - Programovací praktiky
    - Neustálá integrace
    - Jednoduchý návrh
    - RefaktORIZACE kódu a testování

# TDD

- Definice funkcionality → napsat test → implementace
  - Cyklus :
    - Napsat test
    - Spustit testy a ujistit se, že všechny neprojdou
    - **Napsat vlastní kód**
    - Kód automatickými testy prochází
    - Refaktorace →
- 
- A diagram illustrating the TDD cycle. It consists of a list of five steps. The first step is 'Napsat test'. The second step is 'Spustit testy a ujistit se, že všechny neprojdou'. The third step is 'Napsat vlastní kód'. The fourth step is 'Kód automatickými testy prochází'. The fifth step is 'Refaktorace'. A horizontal arrow points from 'Refaktorace' to 'Napsat test', and a diagonal arrow points from 'Napsat test' to 'Refaktorace', forming a loop.

# Verzování softwaru

- Číslo verzí se skládá většinou z 3 čísel : **X.Y.Z**
- První číslo je hlavní (X), druhému se říká vedlejší (Y) a třetímu se číslo revize (Z)
- Pokud se zvýší nějaké číslo tak čísla nižší se resetují (př. : 1.0.1 se zvýší na 1.1.0)
- Číslo revize se zvyšuje po pár commitech (opravy menších chyb)
- Pokud už je velké množství revizí, tj. zásahů, při kterých se pouze opravuje kód, přijde doba, kdy je kód potřeba celý přepsat, zpřehlednit a něco málo upravit → zde se zvyšuje vedlejší číslo
- Hlavní číslo se zvyšuje při kompletně přepracované programu, který již má odlišnou podobu od předchozí verze

**Děkuji za pozornost**

Jiří Šebek

