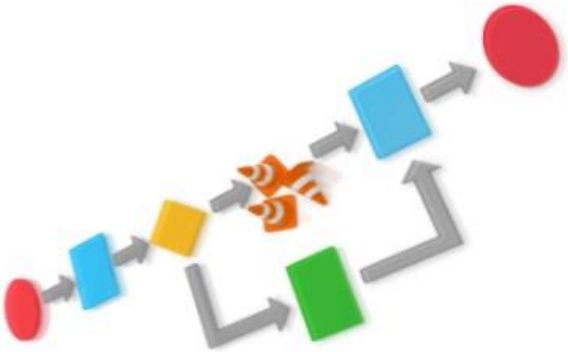
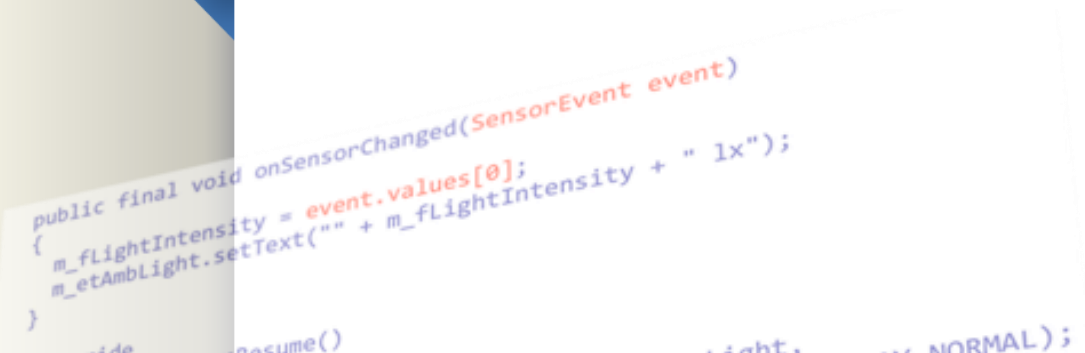
Návrh softwarových systémů - úvod, motivace

Jiří Šebek, Martin Tomášek

Návrh softwarových systémů (B6B36NSS)





Obsah

- Motivace
- Integrace s ostatními obory SI
- Kdo / co ovlivňuje cílový SW
- Modely, metodiky SI
- Verzování SW

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
- Oponent review (~ 8.týdnu) → 5b
- Final Presentation(~ 14.týdnu) → 5b
- Odevzdání na gitlab → 20b (min 10b)
- Zůčastnit se 1 testu:
- 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 udržovatelností, škálovatelností apod.)

Proč SI?

- 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

Softwarové inženýrství

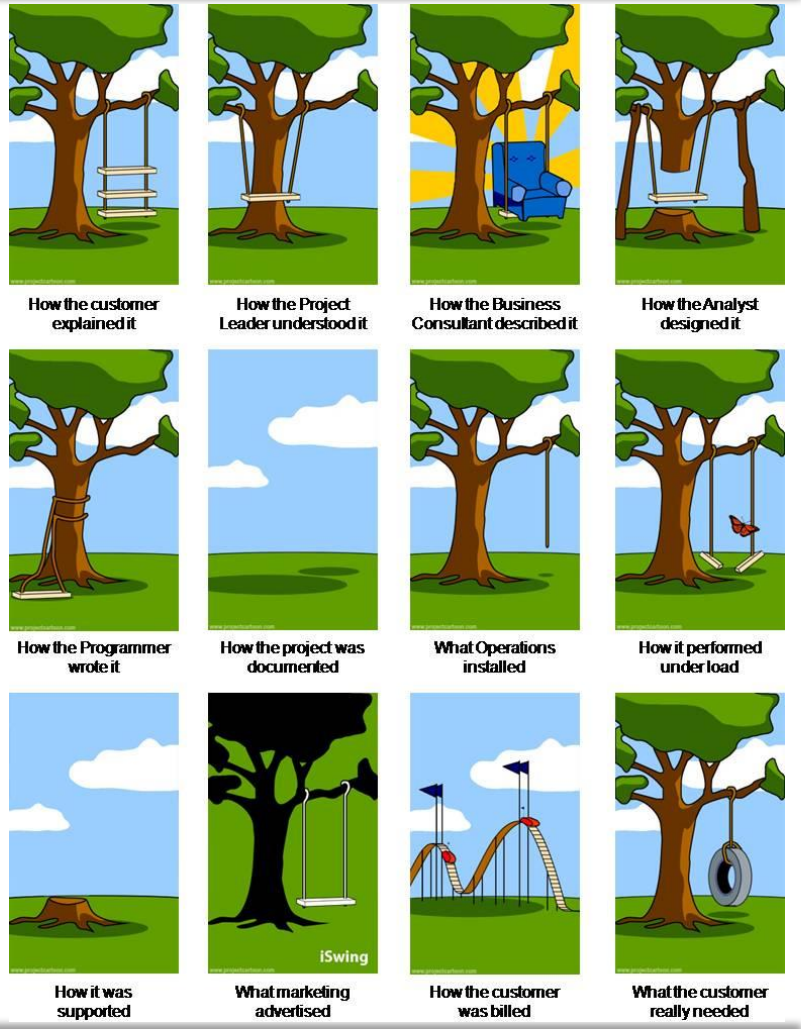
- Využívá modely, metodiky...
- Model
- popisuje fáze životního cyklu softwaru a vztahy mezi nimi
- Metodika
- určuje postupy každé fáze (co se má dělat)
- určuje také vstupy a výstupy
- Dělení :
  - Tradiční
  - agilní

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

Kdo / co ovlivňuje finální produkt

- Zadavatel
- Právní oddělení
- Oddělení architektury
- Vývojové oddělení
- Testovací oddělení
- Oddělení bezpečnosti
- Provozní oddělení
- Cílová infrastruktura
- Doba životnosti SW



Kdo / co ovlivňuje finální produkt

- Technické omezení
- Časové omezení (doba dodávky)
- Výkonové požadavky
- Změny ve vedení firmy
- Globální politika, směřování a skandály
- Cena
- Konkurence
- A mnoho dalších

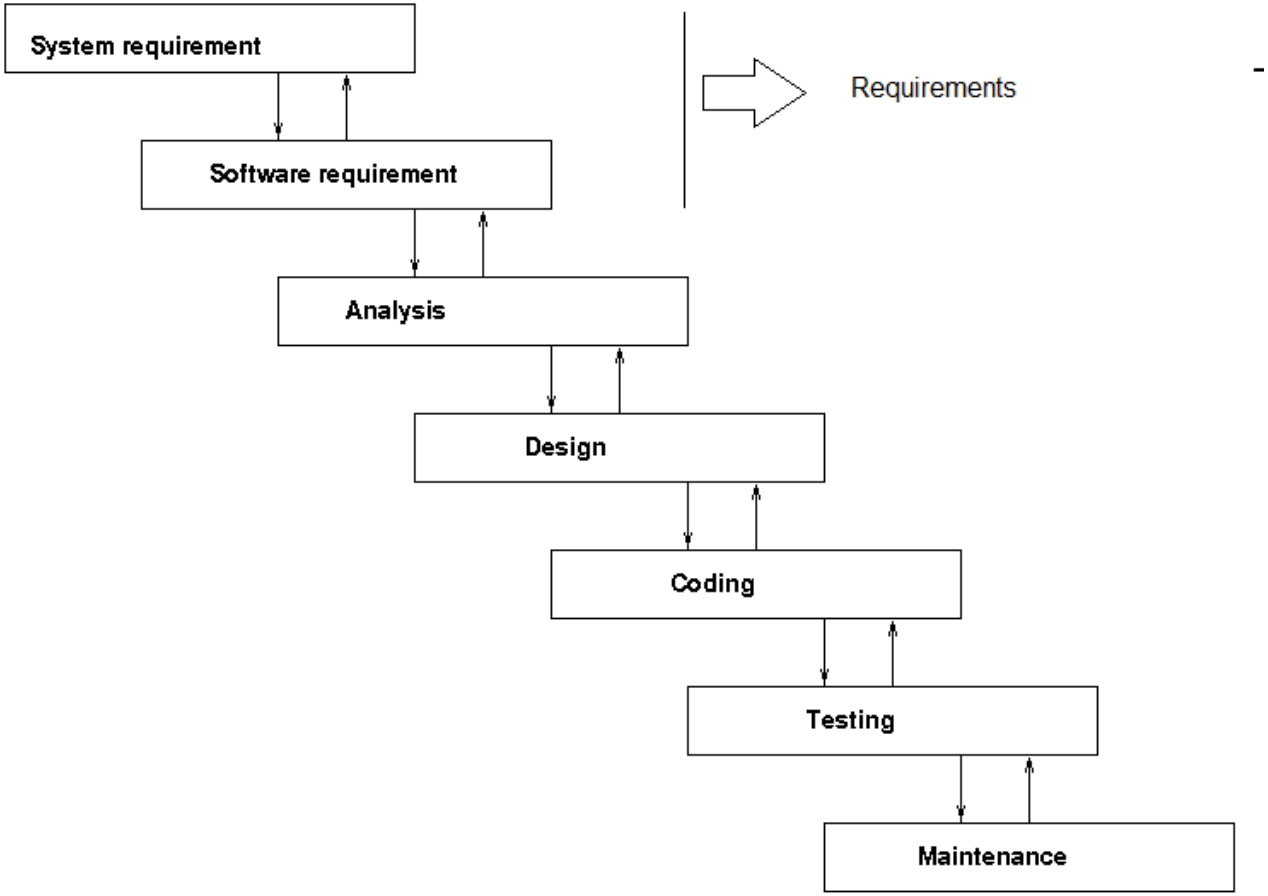
Pouhé SI nestačí

- SI není Silver Bullet – ale nabízí nástroje a metodiky, které ve vývoji SW pomáhají
  - Důležitá je komunikace a plánování
  - Dodržování směrování projektu
  - Umění říci NE, umění prioritizovat požadavky a přizpůsobit se změně
  - Určení správného vývojového modelu a řízení dodávky
  - Určení zodpovědné osoby, která má produkt ve správě a je zodpovědná za rozvoj a směrování produktu

Modely

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

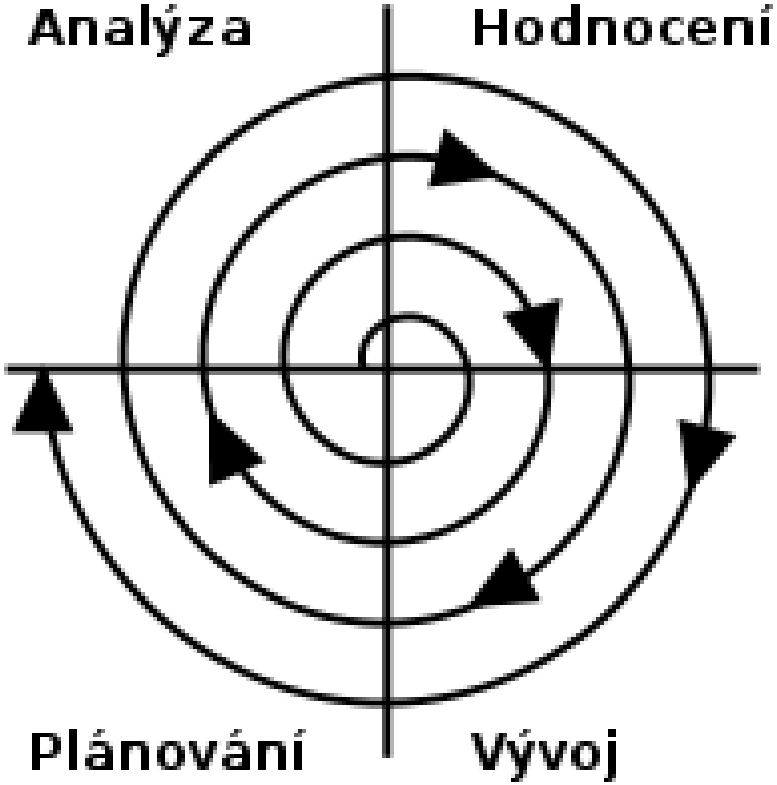
Model vodopád



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 zezačátku vývoje
- Při změně zadání je potřeba schválit výstupy všech fází

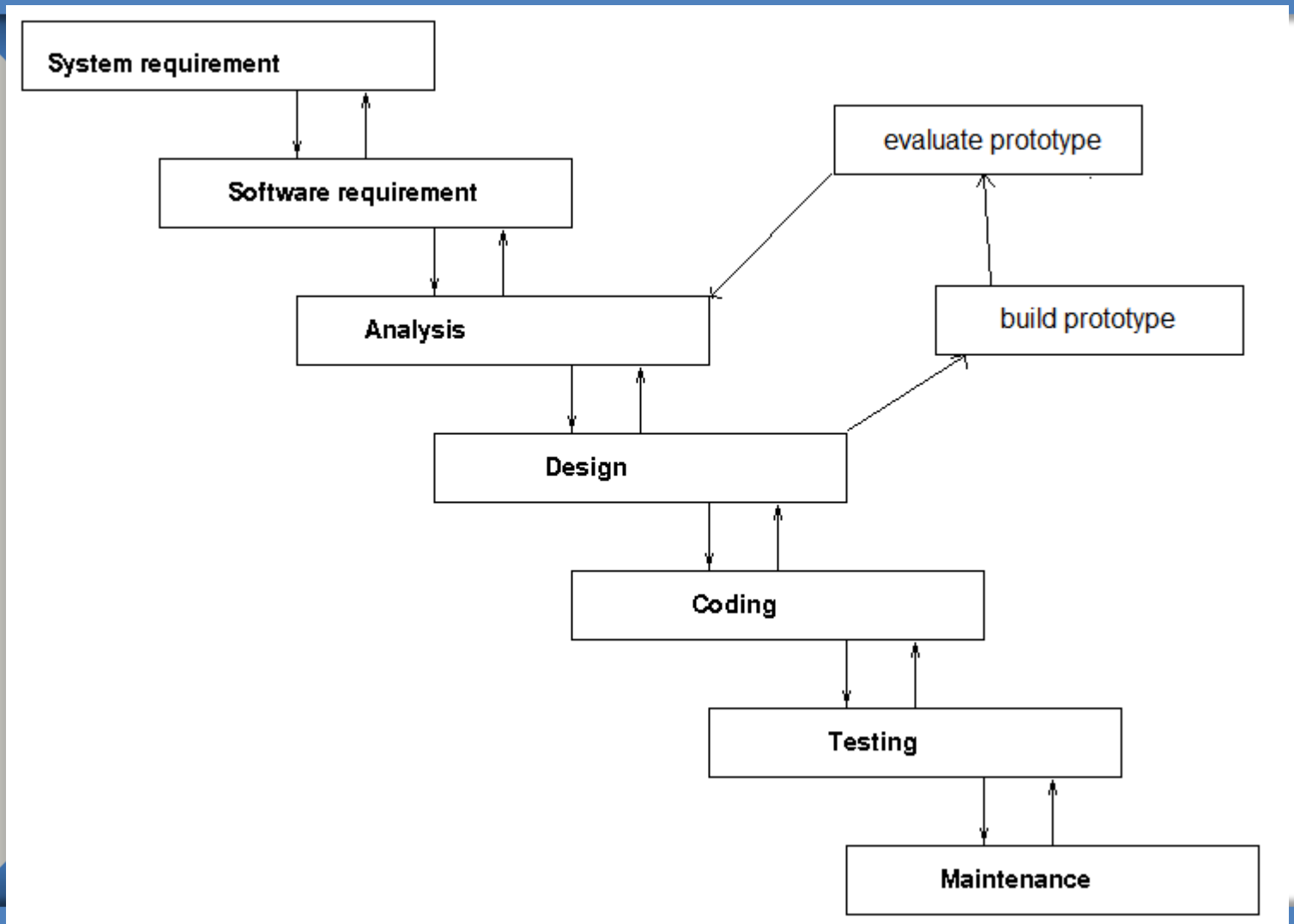
Model spirála



Model spirála vlastnosti

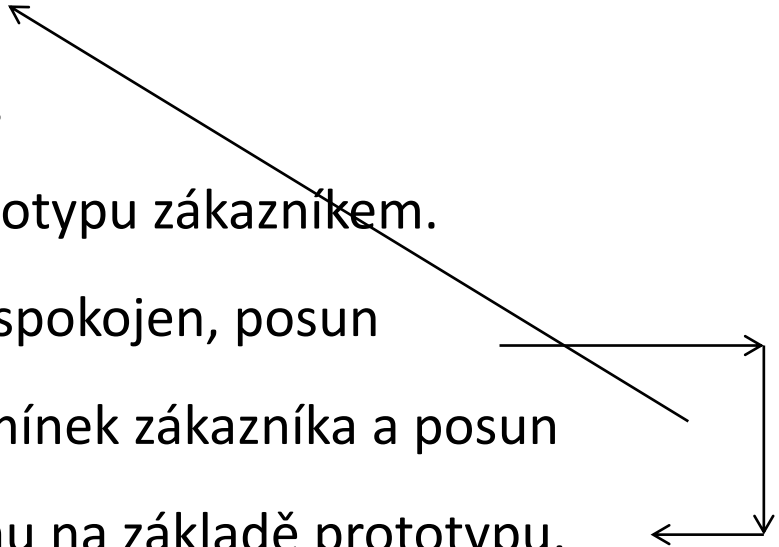
- cyklické opakování jednotlivých kroků vývoje
- iterativní typ modelu
- Je odpovědí na nevýhody vodopádu
- Duležitá vlastnost modelu je analýzu rizik v každém cyklu

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.

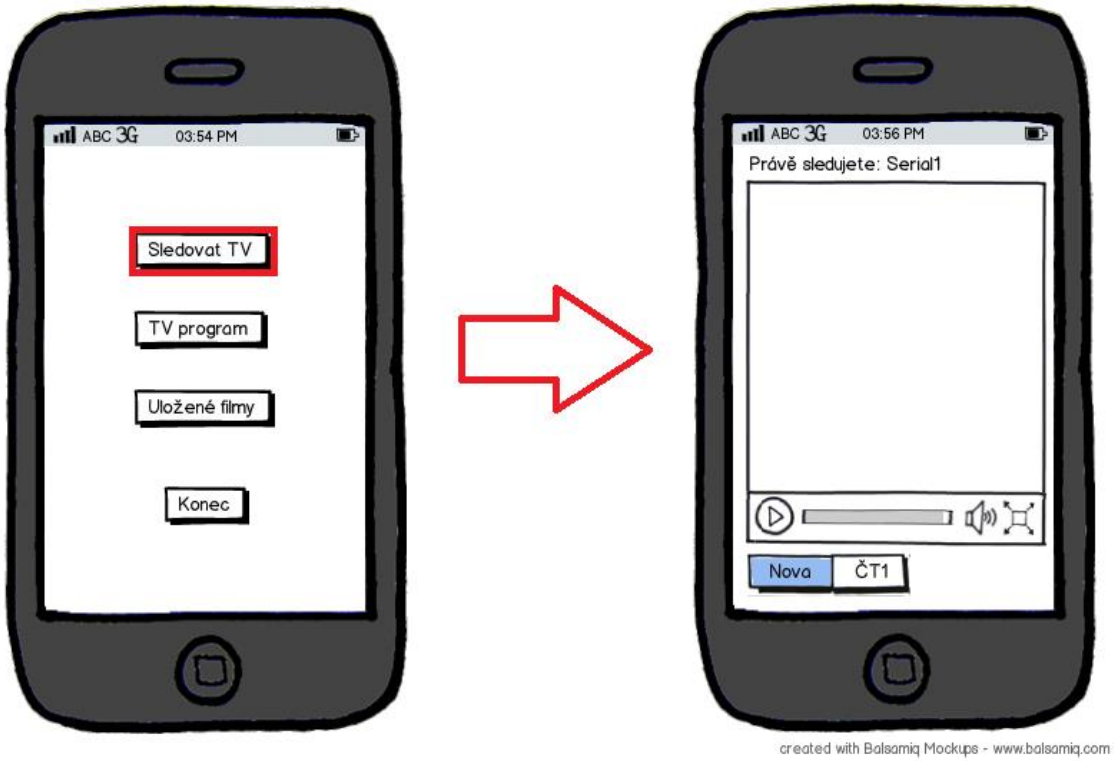


Model prototype

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

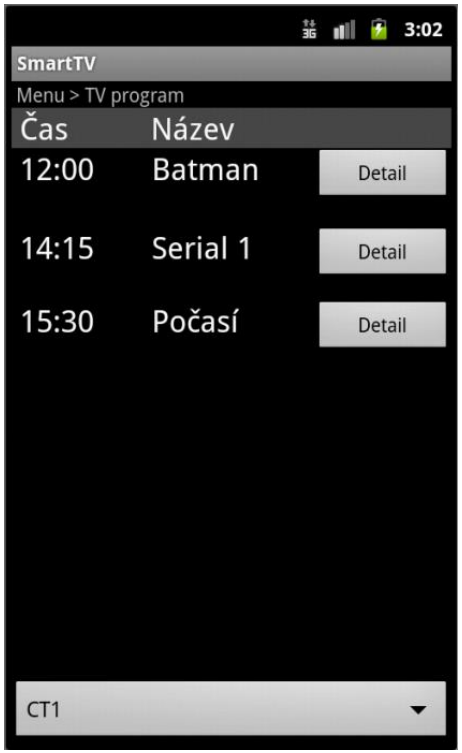
Model prototype

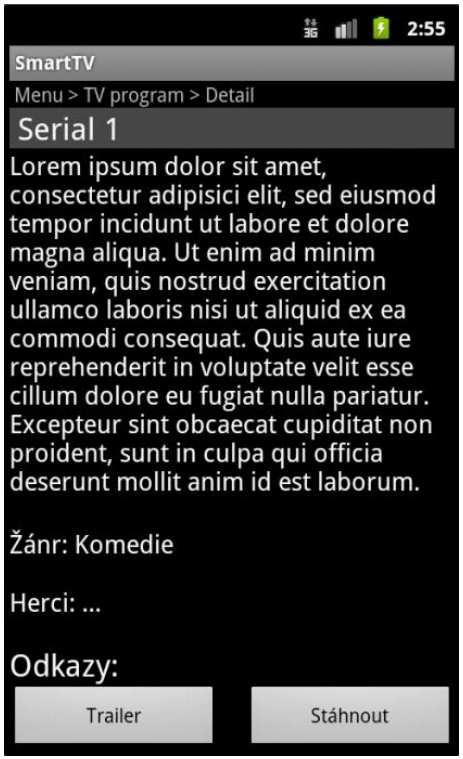
- U UI softwaru dělíme prototypy na :
- Low fidelity



Model prototype

- U UI softwaru dělíme prototypy na :
- High fidelity



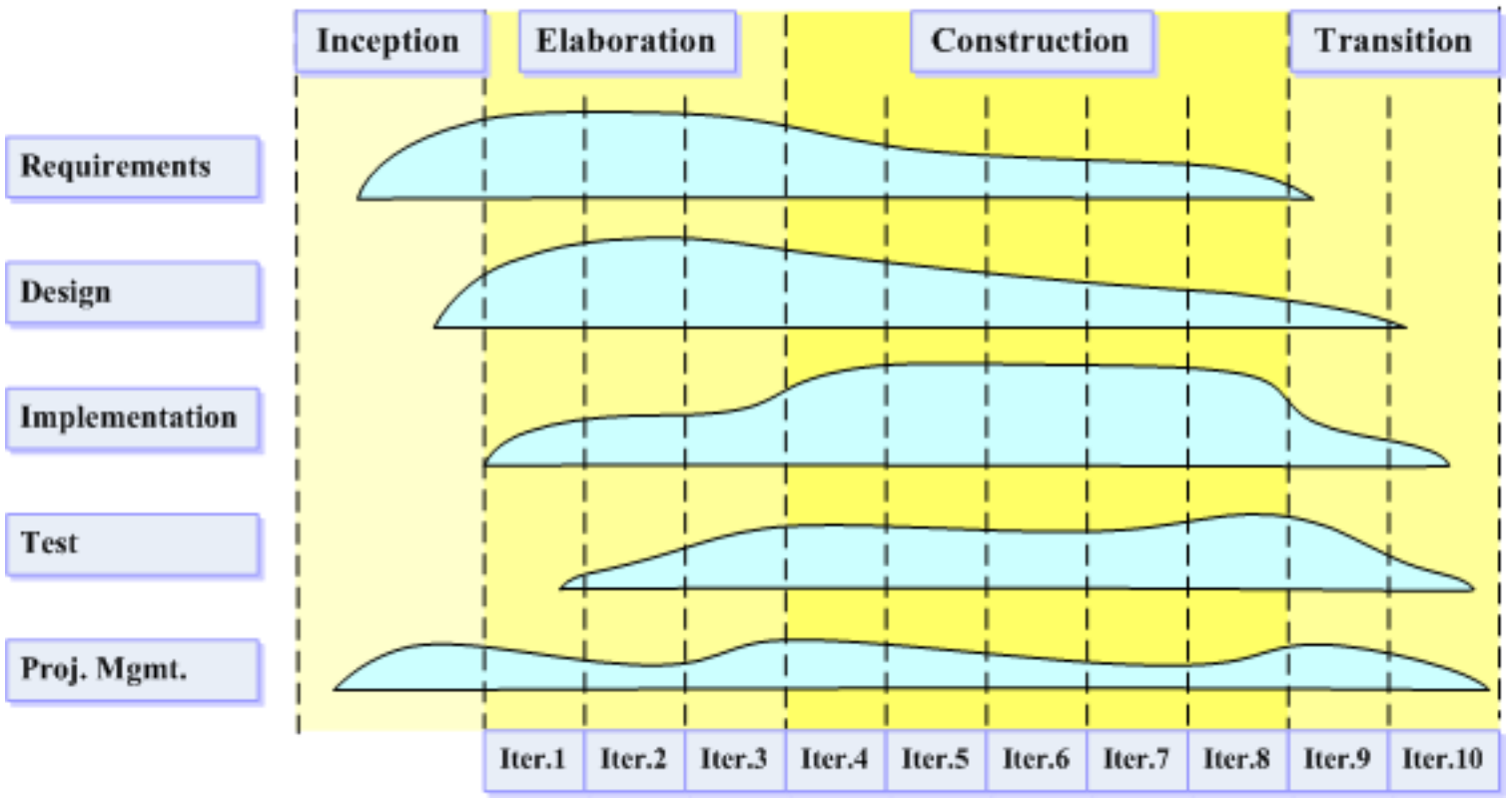


Metodiky

- 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 - Unified Proces



UP - Unified Proces

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

Zdroj:https://en.wikipedia.org/wiki/Unified_Process

XP - Extrémní programování

- **Extrémní programování** (XP) je agilní metodologie vývoje software
- 4 hodnoty:
  - Komunikace
  - Jednoduchost
  - Zpětná vazba
  - Odvaha
  - Párové programování, testování, návrh modulů, jednoduchost, architektura, integrace, iterace,

Zdroj: https://cs.wikipedia.org/wiki/Extr%C3%A9mn%C3%AD_programov%C3%A1n%C3%AD

XP - Extrémní programování

- 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

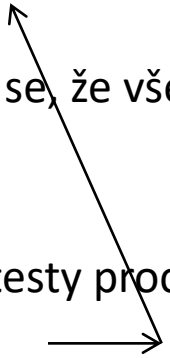
– Programovací praktiky

- Neustálá integrace

Zdroj: https://cs.wikipedia.org/wiki/Extr%C3%A9mn%C3%AD_programov%C3%A1n%C3%AD

TDD - test driven development

- Definice funkcionality → napsat test → implementace
- Cyklus :
- Napsat test
- Spustit testy a ujistit se, že všechny neprojdou
- **Napsat vlastní kód**
- Kód automatickými testy prochází
- Refaktorace



Zdroj: https://cs.wikipedia.org/wiki/Programov%C3%A1n%C3%AD_%C5%99%C3%ADzen%C3%A9_testy

Verzování softwaru

- Čísla 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 ma odlišnou podobu od předchozí verze

Zdroj: http://programujte.com/clanek/2005070702-teorie-programovani-verze-softwaru/

Děkuji za pozornost

Jiří Šebek, Martin Tomášek