Metodiky vývoje softwaru

vodopád, iterativní postupy, agilní vývoj...

© 2008 Radek Ošlejšek
FI MU Brno

oslejsek@fi.muni.cz
http://www.fi.muni.cz/~oslejsek/PA103

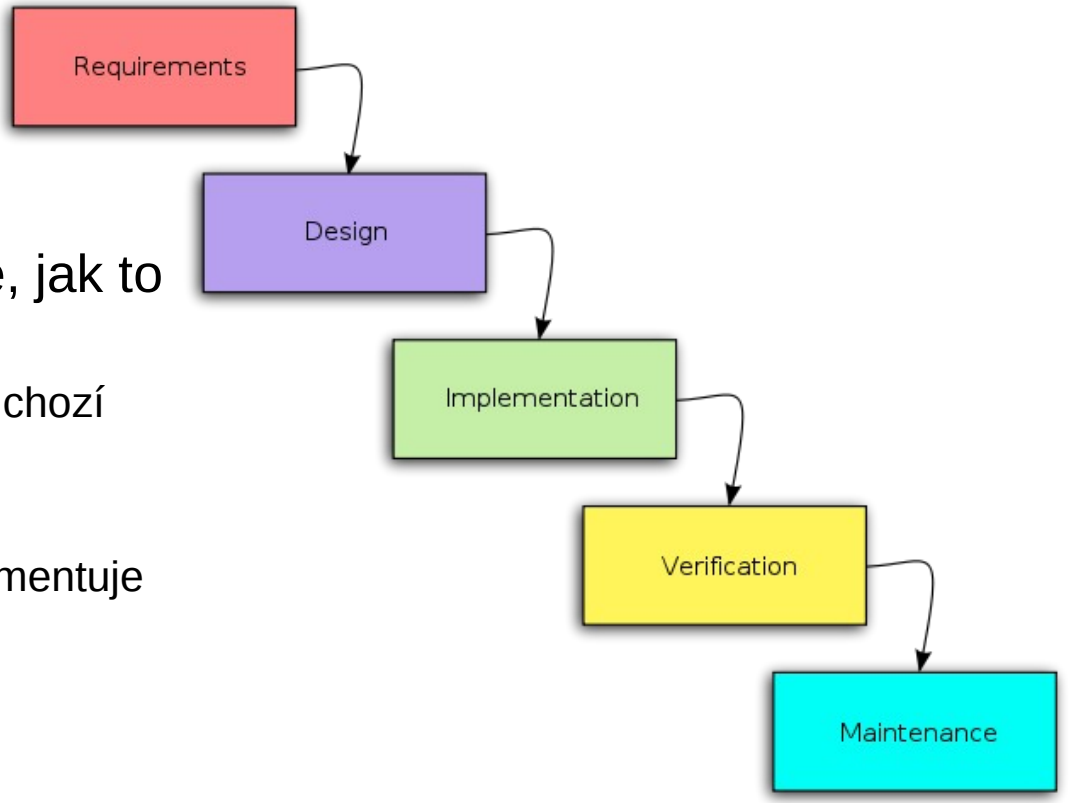
Životní cyklus softwaru

Životní cyklus softwaru

- Životní cyklus SW popisuje život SW od jeho návrhu, přes implementaci, až po jeho předání a údržbu
- Existuje řada modelů:
  - Vodopád, letadlo, výzkumník, spirála, ...
- V objektovém světě se masivně v praxi používají dva modely:
  - Vodopád
  - Iterativní a inkrementální vývoj

Vývoj typu vodopád

- Rozděluje projekt na základě prováděných aktivit:
  - Analýza, návrh, kódování (implementace), testování
- Příklad rozdělení ročního projektu:
  - 2 měsíce analýzy
  - 4 měsíce návrhu
  - 3 měsíce kódování
  - 3 měsíce testování
- Nikdy to není tak jednoduché, jak to na první pohled vypadá:
  - Chyby způsobují návrat do předchozí etapy/etap
  - Nejasná hranice mezi etapami
  - Překrývaní etap (např. se implementuje ještě během návrhu)
  - Pozdní odhalení chyb
  - Odhad ceny



Iterativní vs. inkrementální

- Iterativní vývoj
  - Iterativně = „předělávat“
  - Kroky směřující k vylepšení, zpřesnění nebo opravení části systému
  - Každá iterace obsahuje analýzu, návrh, testování apod. (tj. miniaturní vodopád), ale s různou intenzitou
    - Např. počáteční iterace více analyzují, pozdější více implementují
  - Sice nevyžaduje inkrementální vývoj, ale velmi dobře s ním funguje a mnoho autorů pod pojmem iterativní vývoj zároveň předpokládá i inkrementální
- Inkrementální vývoj
  - Jednotlivé části systému (přírůstky, inkrementy) vytváříme „nezávisle“ na zbytku a pak integrujeme
  - Vývoj jednotlivých přírůstků může probíhat iterativně, vodopádem, XP ...
  - Alternativa k inkrementálnímu vývoji je vývoj celého systému naráz
  - Inkrementálně = „přidávat k“
- Vývoj typu vodopád je tedy:
  - Inkrementální vývoj s jediným přírůstkem představujícím celý systém
  - Iterativní vývoj s jedinou iterací
    - Analýza požadavků, analýza, návrh, kování i testovaní se provádí pouze jednou

Iterativní / inkrementální vývoj

- Rozděluje projekt na základě výběru funkcionalit budoucího systému
  - Vybere se funkcionalita, která se kompletně navrhne, implementuje a otestuje, pak se vybere další funkcionalita...
- Příklad rozdělení ročního projektu:
  - V první iteraci vybereme čtvrtinu funkcí, pro které uděláme analýzu, návrh, implementaci a testování
  - Ve druhé iteraci vybereme další čtvrtinu funkcí...
  - Celkem jsou čtyři iterace, každá iterace dá částečně funkční systém.
- Nikdy to není tak jednoduché, jak to na první pohled vypadá:
  - Integrace přírůstků zabere čas, často se projeví chyby v předchozích přírůstcích
  - První iterace spotřebují mnoho času na analýzu
  - Závěrečné iterace v sobě zahrnují i zaškolování uživatelů
- Každá iterace v podstatě obsahuje miniaturní vodopád

Hybridní postupy

- Objevili se i postupy kombinující oba přístupy:
  - 1996 McConnell: dodávání po částech (staged delivery)
    - Nejdříve analýza a obecný návrh pomocí vodopádu, pak kódování a testování iterativně
    - Př: 4 měsíce analýzy a návrhu, pak 2 měsíce iterativního vývoje

Vodopád vs. Iterativní/inkr. vývoj (I)

- Vodopád:
  - Mnohými analytiky z OO komunity je považován za překonaný
    - Je těžké určit, jestli je projekt na správné cestě k úplné implementaci
    - Je příliš snadné prohlásit úvodní fáze za dokončené a přitom přehlédnout mnoho chyb
    - Jedinou jistotu, že je navržený software správný, nám dá až testování jeho implementace; proto je lepší implementovat a testovat průběžně
  - V praxi je velice často používán, přestože firma deklaruje iterativní vývoj :-(
    - *„Děláme jednu analytickou iteraci následovanou dvěma návrhovými iteracemi...“*
    - *„Kód této iterace má spoustu chyb, ale opravíme je až potom.“*
    - Skutečná iterace produkuje otestovaný kód jehož kvalita se co nejvíce blíží kvalitě finálního produktu
    - Testování a integrace jsou velmi náročné aktivity, které by rozhodně neměly zůstat nedokončené

Vodopád vs. Iterativní/inkr. vývoj (II)

- Iterativní/inkrementální vývoj:
  - Průběžná implementace a testování => včasné varování o chybách
  - Nutnost předělávat kód
    - Iterativní/inkrementální vývoj předpokládá, že pozdější iterace budou upravovat nebo mazat existující kód
    - Ve vývoji SW to nemusí být až tak velká nevýhoda – je lepší špatný kód přepsat, než ho nějak obcházet
    - Existují techniky, které zefektivňují úpravy kódu
      - Automatické testování (*regression tests, unit tests*)
        - » testovací třídy; např. jUnit pro Javu
      - Refaktorizace (*refactoring*)
        - » pravidla pro transformaci zdrojových kódů, může být zautomatizováno
      - Průběžná integrace (*continuous integration*)
        - » např. automatická překlad kódu při uložení změn programátorem + automatické testování

- => iterativní vývoj používejte pouze pro projekty, se kterými chcete uspět :-)

-- Martin Fowler: UML Distilled

Iterativní modely vývoje

agilní vs. model-driven vývoj

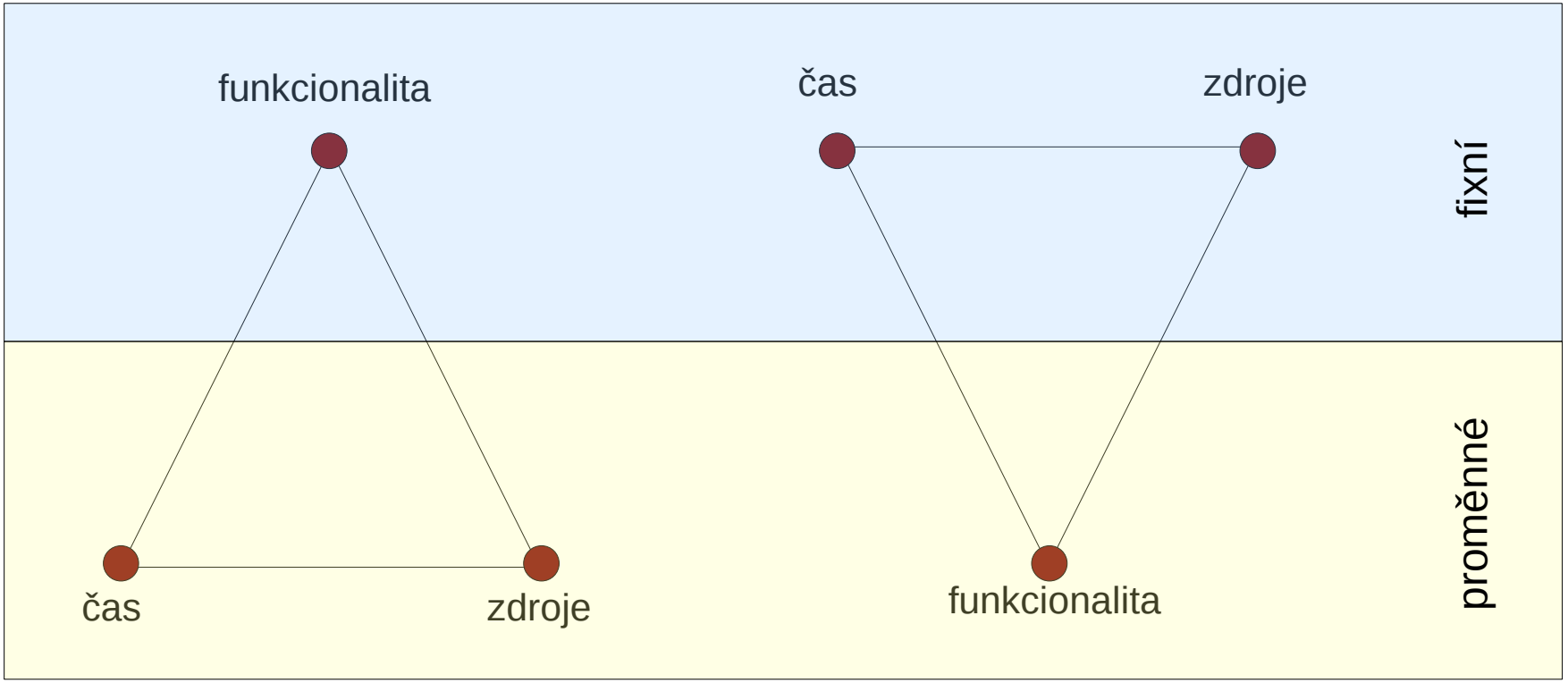
Agilní metody vývoje

- Metody zaměřené více na lidi – určujícím faktorem v úspěchu projektu je kvalita lidí pracujících na projektu a jejich spolupráce
- Velmi krátké iterace (jeden měsíc a méně)
- ***UML se používá pouze jako doplněk***
- Malé ale výkonné týmy (dvojice)
- Zapojení zákazníka do vývoje (zákazník se zúčastní sestavování návrhu a testů, ideálně je součástí vývojového týmu)
- Agilní metodiky:
  - XP – Extreme Programming
  - Scrum
  - FDD – Feature Driven Development
  - Crystal
  - DSDM – Dynamic Systems Development Method
  - ...
- Manifesto of Agile Software Development http://agileManifesto.org
  - Individuality a interakce mají přednost před nástroji a procesy
  - Fungující software má přednost před obsáhlou dokumentací
  - Spolupráce se zákazníkem má přednost před sjednáváním smluv
  - Reakce na změnu má přednost před plněním plánu

Model Driven Engineering - MDE

- „Seriózní“ přístup, kdy implementujeme striktně na základě UML modelů
  - Průběžně vniká dokumentace
  - Podpora pro fázi nasazení a údržby
- ***UML je základním nástrojem*** MDE
- Rational Unified Process - RUP
  - RUP je komerční a licencovaná verze UP od IBM
  - UP a RUP se dají pokládat za synonyma
  - Nejedná se přímo o metodiku (postup), ale spíše o „*process framework*“
    - Nejdříve je třeba stanovit vhodný postup pro konkrétní projekt (tzv. d*evelopment case*)
    - Často se přizpůsobuje i firmě a nástrojům
- Metodiky:
  - Iconix, Select Perspective, ...

MDE vs. agilní přístup



model-driven engineering

agilní přístup

Model Driven **Architecture** - MDA

- Podobně jako MDE/RUP je založena na UML s cílem zformalizovat a zautomatizovat přechod mezi modely
- OMG standardy pro definování rozsahu, obsahu, způsobu vytvoření a použití modelů
- Architektonická metoda začlenění modelů do vývojového procesu
- Čtyři úrovně modelů:
  - CIM – Computation Independent Model
    - Model nezávislý na počítačovém zpracování (business analýza)
  - PIM – Platform Independent Model
    - Platformově nezávislý model řešení
  - PSM – Platform Specific Model
    - Platformově specifický model řešení
  - Code
    - Kód aplikace, tj. výsledná realizace řešení
- ***MDA definuje způsob transformace modelů***

Computation Independent Model

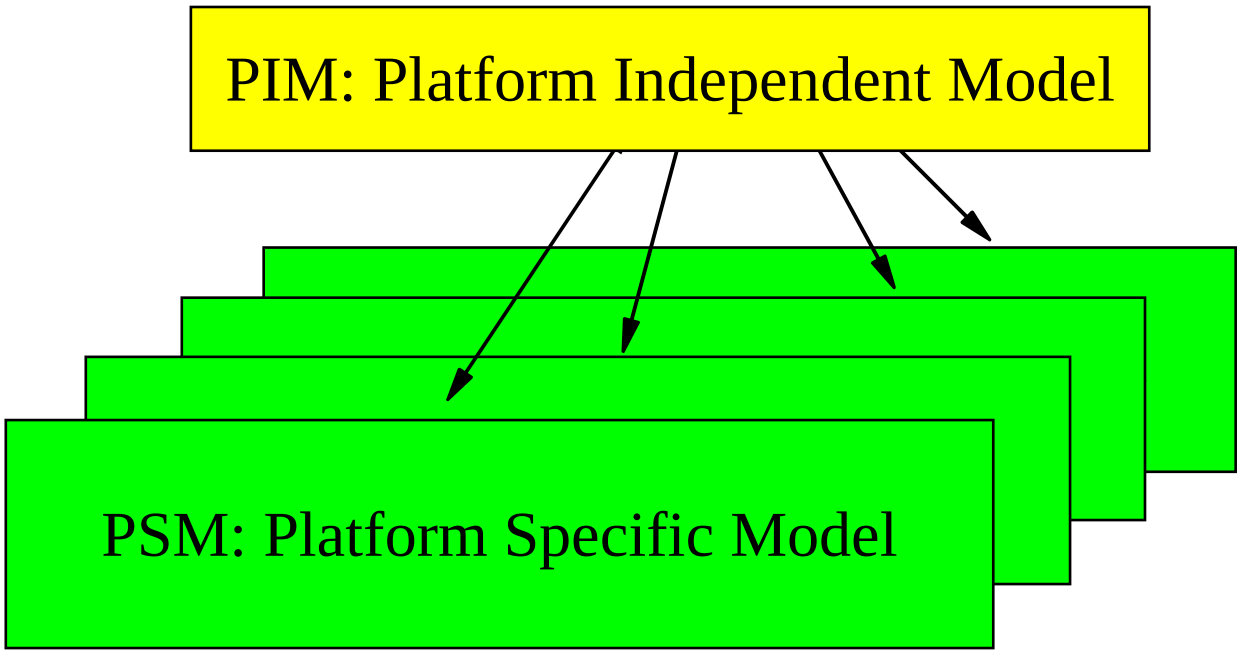
- Business analýza = model procesních toků uvnitř firmy
  - Model podnikových procesů
    - V UML nemá speciální diagram, dá se ale použít diagram aktivit
  - Slovník pojmů problémové oblasti
    - U složitých oblastí se dá vyjádřit pomocí konceptuálního diagramu tříd
- CIM vytvářejí buďto sami uživatelé nebo business analytici
- Zaceluje mezeru mezi experty v modelované problematice a experty na návrh a implementaci systémů

Platform Independent Model

- Analýza, business architektura
- Koncepční řešení problematiky na základě konkrétních požadavků
- Neobsahuje informace spojené s konkrétní technologií realizace
- PIM vytvářejí IT analytici

Platform Specific Model

- Návrh, technologické řešení
- Řešení pro zvolenou platformu (J2EE, CORBA, ...)
- Respektuje technologické standardy a návrhové vzory
- Pro jeden PIM může být několik platformě-specifických modelů
- PSM vytvářejí návrháři



UML

UML and/or
Platform specific notation

Platforms: Web Services, ebXML, J2EE/EJB, CORBA, MS .Net, …

Kód

- Obsah dodávky
- Kód je synchronizován s návrhovým modelem
- Z hlediska MDA je kód chápan jako model konkrétní realizace na dané platformě

Transformace PIM -> PSM

- Transformace Analýza -> Návrh
- Postup:
  1. PIM je doplněn mapovacími značkami, které definují, jaká obecná transformační pravidla budou použita
     - přiřazení obecnějšího návrhového vzoru příslušnému modelovanému elementu
  2. Pro PIM model (resp. jeho části) je zvolena implementační platforma
  3. Na základě mapovacích značek jsou provedeny odpovídající transformace již s ohledem na zvolenou platformu
     - dávky připravených pravidel pro: Java, C#, C++, .Net, ...
     - automaticky se vytvářejí se nové třídy, rozhraní, atributy a operace
     - Na základě mapování tříd do databáze jsou generovány SQL příkazy
- Synchronizace PIM <-> PSM
  - Obousměrná synchronizace důležitá pro inkrementální vývoj
  - Změny pouze s implementační charakteristikou nejsou promítány do PIM