# Extrémne programovanie a iné agilné metodológie

Zimný semester 2007/08

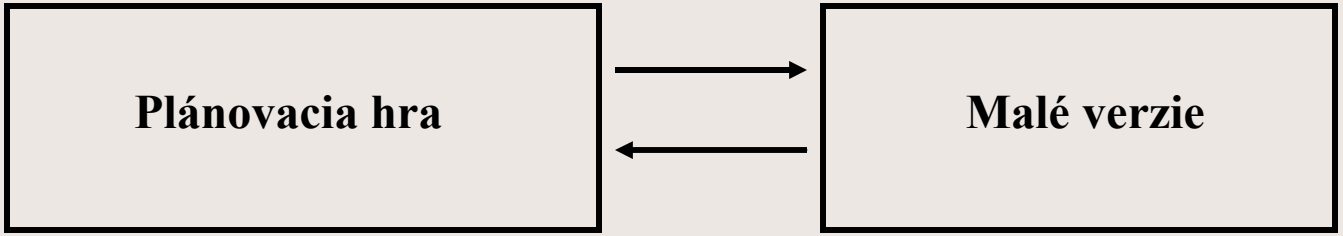
**Ing. František Gyárfáš, PhD.**

Katedra aplikovanej informatiky

**gyarfas@ii.fmph.uniba.sk**

**http://www.ii.fmph.uniba.sk/~gyarfas/**

# Riadenie projektu v XP

# Metafora

- Metafora je systémový pohľad z výšky 10 000 metrov.
- Metafora popisuje zmysel projektu, nie jeho realizáciu. Napr. výpočet dôchodku je niečo ako tabuľka, počítač môžeme chápať ako plochu stola.
- Metafora pomáha pochopiť základné prvky a ich vzťahy.
- Metafora nahrádza to, čo ostatní nazývajú architektúrou. Reprezentuje cieľ architektúry, nie jej detaily.
- Metafora poskytuje príbeh, v rámci ktorého sa vývoj systému odohráva.
- Tento príbeh môžu ľahšie zdieľať obchodníci aj technici.

**Pravidlo vyššej pravdy**
Ak požadujeme metaforu, zvyčajne získame aj základnú architektúru, ktorá sa ľahko vysvetľuje aj zdokonaľuje.

# Plánovacia hra

- stanoviť šírku zadania a priority jednotlivých cieľov
- zostaviť tým
- odhadnúť náklady a harmonogram
- dať všetkým, ktorí sa projektu zúčastnia, pocit istoty, že sa systém skutočne dokončí

# Mantinely plánovania

**Alibistické pravidlo evolúcie**
Vývoj softvéru je vždy dialóg medzi žiaducim a možným.

**Dialektické pravidlo evolúcie**
Žiaduce aj možné sa neustále mení.

# Plánovanie

Plánovanie je to, čím začíname prácu na XP projekte. Neplánujeme všetky detaily, ale iba odhadujeme, akým smerom sa bude práca uberať.

- **Plánujeme iba to, čo potrebujeme pre ďalšiu iteráciu.** Ak robíme dlhodobejšie plánovanie, tak bez detailov.
- **Prijatie zodpovednosti –** v XP nie je možné zodpovednosť **zadať, ale iba prijať**. Manažér nemôže zadať úlohy tímu. Musí požiadať tím, aby zodpovednosť prijal.
- **Ten, kto príjme zodpovednosť za implementáciu, musí vedieť odhadnúť, koľko to bude trvať.**
- Plánujeme tak, aby sa jednotlivé časti mohli implementovať v ľubovoľnom poradí. To nám umožní implementovať jednotlivé časti podľa ich priority.

V plánovacej hre časť plánovania vykonávajú **zákazníci** a časť **programátori**.

# Plánujúci zákazníci

O čom by mali rozhodovať zákazníci?

- **Šírka zadania –** aká veľká časť problému sa musí vyriešiť, aby malo prevádzkovanie systému zmysel
- **Priorita –** Ak môžeme mať najprv iba A alebo B, čo si vybrať?
- **Skladanie verzií** – koľko je potrebné urobiť, aby sa mal zákazník lepšie so systémom ako bez neho?
- **Dátumy uvoľňovania softvéru** – ktoré dátumy sú tak dôležité, že by existencia softvéru (alebo jeho funkčnej časti) mohli mať veľký význam?

# Plánujúci programátori

O čom by mali rozhodovať programátori?

- **Odhady –** ako dlho bude trvať implementácia určitej funkcie?
- **Dôsledky –** strategické rozhodnutia by sa mali robiť iba na základe informácií o technických dôsledkoch rozhodnutia.
- **Proces** – ako sa bude organizovať práca v tíme.
- **Podrobný harmonogram** – postupnosť zadaní. Najriskantnejšie časti programu by sa mali presunúť na začiatok projektu, aby sa znížilo jeho riziko.

# Plánovacia hra

Plánovacia hra sa hrá podľa pravidiel:

- **Cieľ –** cieľom hry je maximalizovať hodnotu vytvoreného programu. Z hodnoty samotného softvéru je treba odpočítať náklady na jeho vývoj a riziko, ktorému sa pri vývoji vystavujeme.
- **Stratégia –** stratégia tímu je investovať čo najmenej tak, aby sa čo najskôr uviedli najcennejšie funkcie do prevádzky.
- **Hracie kamene** – sú to karty zadania (sú to kartičky, na ktorých sú popísané jednotlivé plánované činnosti). S nimi sa hrá, kto a kedy ich vykoná.
- **Hráči** – sú dvaja hráči plánovacej hry – **Vývoj** a **Vedenie**. **Vývoj** sú všetci implementátori a **Vedenie** sú tí, ktorí rozhodujú o tom, čo má systém robiť (obchodníci, vedúci pracovníci a skutoční budúci užívatelia)

# Ťahy

**Ťahy –** hra sa skladá z troch fáz:

- **Prieskum** – zisťovanie, ktoré nové veci by systém mohol robiť
- **Záväzok** – vyberie sa podmnožina z možných požiadaviek
- **Riadenie** – organizujeme vývoj podľa potrieb nášho plánu

# Prieskum

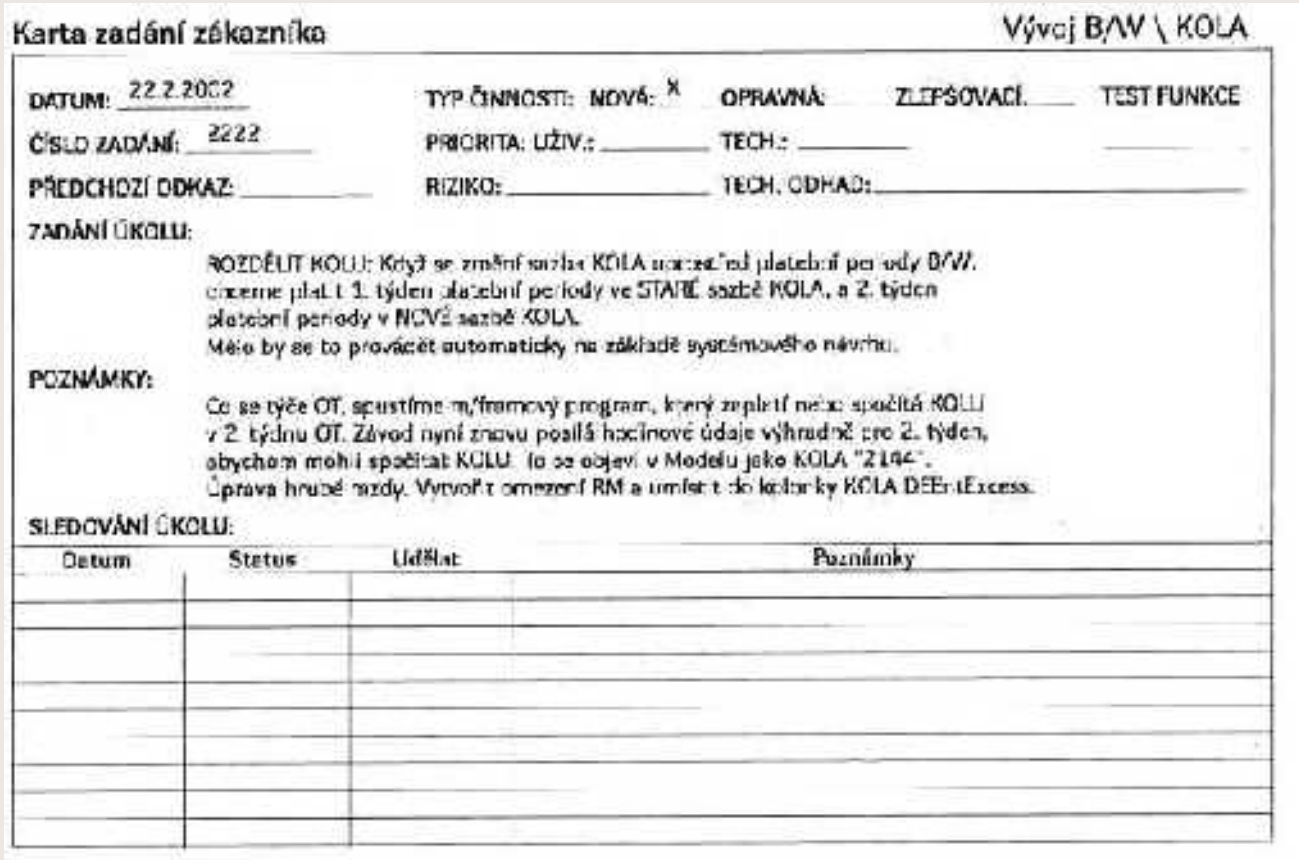
**Prieskum:** účelom prieskumu je poskytnúť hráčom objasnenie toho, čo všetko by mal systém nakoniec robiť.

- **Vedenie vytvorí karty zadaní** – vznikajú kartičky, obsahujúce názov zadania a krátky popis jeho účelu. Karty sa ukladajú do kartotéky zadaní.

- **Vývoj odhadne ideálny čas implementácie zadaní** – ideálny čas je doba, ktorú by trvala implementácia iba tohto jedného zadania, keby riešiteľa nič nerušilo a nemal by žiadne porady.

- **Rozdelenie zadaní** – ak **Vývoj** nie je schopný odhadnúť celé zadanie, alebo si **Vedenie** uvedomí, že časť zadania je dôležitejšia ako zbytok, môže sa zadanie rozdeliť na dve či viac samostatných zadaní. Podobne môžeme aj zlúčiť niekoľko malých úloh.

# Karty zadania



**Karta zadání zákazníka** — Vývoj B/W \ KOLA

DATUM: 22.2.2002 — TYP ČINNOSTI: NOVÁ: X — OPRAVNÁ: — ZLEPŠOVACÍ: — TEST FUNKCE

ČÍSLO ZADÁNÍ: 2222 — PRIORITA: UŽIV.: — TECH.: —

PŘEDCHOZÍ ODKAZ: — RIZIKO: — TECH. ODHAD: —

ZADÁNÍ ÚKOLU:

ROZDĚLIT KOLU: Když se změní sazba KOLA uprostřed platební periody B/W, chceme platit 1. týden platební periody ve STARÉ sazbě KOLA, a 2. týden platební periody v NOVÉ sazbě KOLA.
Mělo by se to provádět automaticky na základě systémového návrhu.

POZNÁMKY:

Co se týče OT, spustíme m/framový program, který zpracuje nebo spočítá KOLU v 2. týdnu OT. Závod nyní znovu posílá hodinové údaje výhradně pro 2. týden, abychom mohli spočítat KOLU. To se objeví v Modelu jako KOLA "2 týd".
Úprava hrubé mzdy. Vytvořit omezení RM a umístit do kolonky KOLA DEEmExcess.

SLEDOVÁNÍ ÚKOLU:

| Datum | Status | Udělat | Poznámky |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Záväzok

**Záväzok:** účelom záväzku je, aby **Vedenie** zvolilo šírku zadania a dátum uvoľnenia ďalšej verzie a **Vývoj** sa zodpovedne zaviazal k jeho splneniu.

**Triedenie zadaní podľa hodnoty** – **Vedenie** vytvorí tri hromádky:

1) zadania, bez ktorých by systém nefungoval
2) zadania, ktoré sú menej nevyhnutné, ale poskytujú podstatnú hodnotu zákazníkovi
3) zadania, ktoré by mali zákazníci radi

**Triedenie zadaní podľa rizika** – **Vývoj** vytvorí tri hromádky:

1) zadania, ktoré dokáže odhadnúť pomerne dobre
2) zadania, ktoré nedokáže presne odhadnúť
3) zadania, ktoré nedokáže odhadnúť vôbec

**Nastavenie rýchlosti** – **Vývoj** oznámi **Vedeniu**, ako rýchlo môže tím programovať v ideálnom čase za kalendárny mesiac

**Zvolenie šírky zadania** – **Vedenie** určí príslušnú sadu kariet. Buď určením dátumu a vypočítaním, ktoré karty sa do vtedy dajú splniť, alebo vypočítaním dátumu cez výber kariet.

# Riadenie

**Riadenie:** účelom riadenia je aktualizovať plán podľa momentálnych poznatkov **Vývoja** a **Vedenia**. Obsahuje štyri ťahy:

- **Iterácia** – na začiatku každej iterácie (1-3 týždne) si **Vedenie** vyberie najcennejšie zadania v rámci jednej iterácie, ktoré sa majú implementovať.
- **Prehodnotenie –** ak **Vývoj** zistí, že prehnal odhad rýchlosti, môže sa opýtať **Vedenia**, aká je najcennejšia sada zadaní, ktorá sa má v aktuálnej verzii zachovať na základe odhadu novej rýchlosti.
- **Nové zadanie** – ak **Vedenie** zistí, že potrebuje nové zadanie uprostred vývoja danej verzie, môže ho napísať. **Vývoj** zadanie časovo odhadne, **Vedenie** odstráni zadanie s ekvivalentným časovým odhadom z plánu a vloží do neho nové zadanie
- **Nový odhad** – ak má **Vývoj** pocit, že plán už neposkytuje presnú mapu vývoja, môže znovu odhadnúť zostávajúce zadania a znovu nastaviť rýchlosť.

# Malé verzie

**Small is beautiful**
Všetky nové verzie by mali byť čo najmenšie a mali by obsahovať to najcennejšie pre zákazníka

**Pravidlo najmenšej úplnosti**
Každá odovzdaná verzia musí dávať zmysel ako celok. Polovične vytvorené funkcie do odovzdanej verzie nepatria.

**Pravidlo vrabca v hrsti**
Je vždy lepšie plánovať verzie po mesiaci, ako po polroku.

# Programovanie XP projektu

- **Jednoduchý návrh –** systém má byť navrhnutý tak jednoducho, ako v danom okamihu možné.
- **Testovanie –** programátori píšu neustále testy, ktoré musia prebiehať bez chýb. Zákazníci píšu testy funkcionality, ktoré predvádzajú jednotlivé dokončené funkcie.
- **Refaktorizácia** – odstraňuje opakovanie kódu, zlepšuje komunikáciu, zjednodušuje systém, bez zmeny funkčnosti.
- **Párové programovanie** – všetok kód píšu dvojice programátorov.

# Projekt v XP - 2

• **Spoločné vlastníctvo –** všetci smú kedykoľvek meniť ktorýkoľvek zdrojový text.

• **Nepretržitá integrácia** – systém integrujeme niekoľkokrát denne vždy, keď je nejaká úloha dokončená.

• **40 hodinový týždeň**

• **Zákazník na pracovisku –** do vývojového týmu patrí aj zákazník, budúci užívateľ, ktorý je k dispozícii na plný úväzok, aby odpovedal na otázky.

• **Štandardy pre písanie zdrojového textu–** programátori píšu všetok zdrojový text v súlade s pravidlami, ktoré zdôrazňujú komunikáciu prostredníctvom zdrojového textu.

# Jednoduchý návrh

V XP správny návrh programového vybavenia sa v ľubovoľnom okamihu vyznačuje týmito pravidlami:

- všetky testy fungujú
- oznamuje všetky zámery, ktoré sú dôležité pre programátorov
- má čo najmenší počet tried a metód a neobsahuje duplicitnú logiku

**Pravidlo krásnej súčasnosti**
**Navrhujeme pre zajtrajšok , implementujeme pre dnešok.**

Ak je budúcnosť neistá alebo môžeme zmeniť svoj názor, potom programovať na základe budúcich potrieb je ľahkovážne.

# Testovanie

- testy sa píšu o skôr, ako vlastný zdrojový text
- všetky testy sa uchovávajú navždy
- testy sa spúšťajú vždy všetky a spolu

| Abs | **Absolútne pravidlo**<br>Bez testu funkcia neexistuje. |
| --- | --- |

# V akom stave prerušiť prácu

- Ak pracujete sám, nechajte posledný test v zlyhávajúcom stave.
- Ak pracujete v tíme, vždy nechajte program so všetkými testmi funkčnými.
- Ak musíte skončiť a neviete opraviť nejaký test, zahoďte celú prácu.

**Pravidlo individuálnej zodpovednosti**
*Vykomentovať zlyhávajúci test je zločin.*

# Párové programovanie

Všetok zdrojový text píšu dvaja programátori pri jednom počítači.

Zoskupovanie do párov je dynamické a premenlivé. Ráno môžeme utvoriť pár s niekým a popoludnie už s niekým iným.

Ak máme vytvoriť niečo, čo príliš nepoznáme, môžeme požiadať o partnerstvo niekoho, kto s tým nedávno pracoval.

# Vlastníctvo

Individuálne vlastníctvo zdrojového textu vedie k tomu, že ľudia sa zdráhajú zasahovať do cudzieho textu. Keďže nechcú vyrušovať vlastníka, zdrojový text zostáva pomerne stabilný a nerozvíja sa tak rýchlo, ako je treba. A čo keď potom vlastník kódu ukončí pracovný pomer?

V XP preberajú všetci zodpovednosť za celý systém. Nie všetci poznajú celý systém rovnako dobre, ale niečo vedia o všetkých častiach. Ak pár pracuje a vidí príležitosť k zlepšeniu zdrojového textu, začne ho zdokonaľovať, ak mu to uľahčí život.

**Pravidlo spoločnej zodpovednosti**

*Každý, kto vidí príležitosť k pridaniu hodnoty do ľubovoľnej časti programu, má to urobiť.*

# Spoločné vlastníctvo kódu

Spoločné vlastníctvo môže obstáť iba s pomocou testov.

Spoločné vlastníctvo môže obstáť iba s nepretržitou integráciou.

Tendencia spoločného vlastníctva je, že bráni zložitému zdrojovému kódu, aby sa do systému vôbec dostal. Ak vieme, že sa veľmi skoro (za pár hodín) na kód pozrie niekto iný, dobre si rozmyslíme, či tam vložíme niečo, čo by sme ťažko vedeli obhájiť.

Spoločné vlastníctvo šíri znalosť celého systému medzi členmi tímu. Je nepravdepodobné, že by existovala časť systému, o ktorej by vedeli iba dvaja ľudia.

# Nepretržitá integrácia

Zdrojový text sa integruje a testuje každých niekoľko hodín, najmenej však raz denne.

Najjednoduchší spôsob, ako to dosiahnuť, je mať k dispozícii jeden počítač vyhradený pre integráciu. Keď je tento počítač voľný, pár s novým zdrojovým kódom sa k nemu posadí, pridá svoj nový kód, odstráni problémy a pustí všetky testy.

**Pravidlo posledného**
Integráciu nie je dovolené opustiť, kým všetky testy neprebehnú bezchybne.

**Pravidlo minimalizácie škôd**
Ak nie sme schopní dosiahnuť, aby všetky testy prebehli bezchybne, musíme zahodiť všetko, čo sme naprogramovali.

# Vlastnosti integrácie

Žiaden zdrojový text neostáva nezintegrovaný dlhšie, ako niekoľko hodín.

Ak vyvíjame svoju časť programu, chceme mať ilúziu, že sme jediní programátori, ktorí na systéme pracujú. Robíme zmeny, kde to považujeme za potrebné.

Ak je systém dostatočne refaktorizovaný, je rozdrobený do toľkých malých kúskov, že je relatívne malá pravdepodobnosť, že nami zmenený kód menil v tom istom čase aj niekto iný. Ak k tomu napriek tomu dôjde, integrácia každých pár hodín zabezpečuje, že zmeny sa týkajú iba pár hodín práce.

Neustála integrácia znižuje riziko, že rôzni členovia tímu majú zásadne odlišnú predstavu o funkčnosti a vnútia programu nekompatibilné vlastnosti.

Neustála integrácia dodáva vývoju rytmus: učiť sa, testovať, písať zdrojový text, integrovať. Sformujeme myšlienku, vyjadríme ju, vložíme do systému. A myseľ je znovu čistá, pripravená na ďalšiu myšlienku.

# 40 hodinový týždeň

Ráno sa chceme cítiť sviežo a večer unavene a spokojne.
V piatok chceme byť dosť unavení, aby sme mali dobrý pocit z dvojdňového odpočinku.

40 hodín týždenne na pracovisku môže znamenať u jedného 35 alebo aj 45. Ale nikto nemôže pracovať 60 hodín týždenne mnohokrát za sebou a stále byť kreatívny, svieži, pozorný, opatrný a sebavedomý.

Presčas je príznakom vážneho problému s projektom. Pravidlo XP hovorí: jeden týždeň to ide. Viac ako jeden týždeň to už robiť nesmieme.

# Zákazník na pracovisku

V týme musí sedieť skutočný zákazník, ktorý je k dispozícii a odpovedá na otázky, rieši spory a určuje čiastkové priority.

Pod zákazníkom myslíme niekoho, kto bude vytváraný systém naozaj používať, až už bude v prevádzke.

Veľká námietka proti tomuto pravidlu sa zakladá na tom, že skutoční používatelia sú príliš vzácni, aby ich bolo možné poskytnúť na vývoj.

Manažéri sa preto musia rozhodnúť, čo má väčšiu hodnotu: mať fungujúci a užitočný softvér skôr a lepšie, alebo práca jedného človeka. Ak systém neprinesie spoločnosti väčšiu hodnotu, ako je práca jedného človeka, nemal by sa ani vôbec budovať.

# Štandardy pre písanie zdrojového textu

Ak majú všetci programátori zdieľať všetok zdrojový text, nemôžeme si dovoliť mať viac sád pravidiel na písanie zdrojového kódu.

# Pravidlá workoholizmu

**Hrdinské pravidlo workoholizmu**

*Zničím si zdravie, odcudzím sa rodine a hoci sa aj zabijem, ale urobím všetko, čo bude treba.*

**Smutné pravidlo workoholizmu**

*Workoholizmus nie je hrdinstvo, ale choroba.*

# Ako zvládať workoholizmus

Techniky, ako zvládať workoholizmus (a vynútiť si prestávku):

- **Na úrovni hodín** majte pri počítači fľašu s tekutinou. Biológia pomôže.
- **Na úrovni dní** si dohadujte stretnutie s priateľmi po pracovných hodinách.
- **Na úrovni týždňov** si robte program na víkend, ktorý sa nedá zrušiť.
- **Na úrovni rokov** si plánujte dovolenku podľa francúzskeho hesla: dva týždne nestačia.

# Úloha

***Úloha:*** **Návrh programového projektu ako ročníkový projekt z XP.**

Vytvorte návrh aplikácie, ktorú plánujete vytvoriť v rámci tohto predmetu ako ročníkový projekt. Tento návrh môžete vytvoriť samostatne, alebo ako skupina.(2-6)

Pri výbere aplikácie sa sústreďte na tieto vlastnosti:

**Zaujímavosť témy (čo by Vás bavilo programovať)**

**Programátorská náročnosť**

**Možnosť inkrementálneho vývoja programu, dokončiteľnosť programu**

**Dôraz na programovanie vo dvojiciach, testovanie, refaktorizáciu**

**Termín odovzdania: do stredy 2.12.2007 – 24.00.**