Téma přednášky

- **Modely životního cyklu**
- **Metodiky vývoje softwaru**
  - heavyweight metodiky
  - agilní metodiky

Úvod do softwarového inženýrství

IUS 2019/2020

9. přednáška

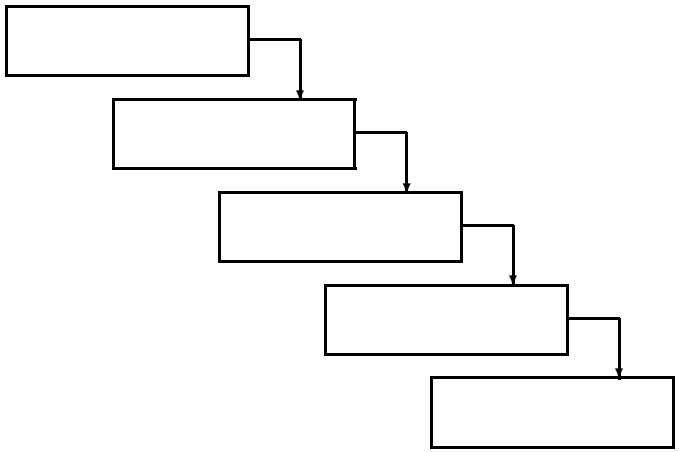
Ing. Radek Kočí, Ph.D.
Ing. Bohuslav Křena, Ph.D.

25. listopadu a 29. listopadu 2019

Lineární modely životního cyklu

Lineární (sekvenční) modely

- životní cyklus jde postupně od první etapy až do poslední



- typický představitel je vodopádový model

Organizace následujících přednášek

- **10. přednáška – Řízení softwarových projektů**
  - Ing. Dana Brhelová, Artysis, s.r.o.
  - pondělí 2. 12. 2019
  - pátek 6. 12. 2019
- **11. přednáška – Management SW projektů, řízení kvality softwaru, SW metriky, tým a motivace lidí**
  - pondělí 9. 12. 2019
  - pátek 13. 12. 2019
- **12. přednáška – Ochrana intelektuálního vlastnictví, etický kodex softwarového inženýra**
  - pondělí 16. 12. 2019
  - pátek 20. 12. 2019

W-model

Vlastnosti

- vychází z V-modelu
- aktivity spojené s ověřováním a testováním jsou na stejné úrovni jako návrhové aktivity $\Rightarrow$ druhé souběžné V
- levá strana: ověřování výstupů etap a plánování a návrh testů
- pravá strana: provádění testů, změny kódu, regresní testování, ...

V-model

Vlastnosti

- vychází z vodopádového modelu
- písmeno V symbolizuje grafické uspořádání etap, zdůrazňuje vazby mezi návrhovou a testovací částí
- písmeno V je také synonymem pro validaci a verifikaci
- zachovává si jednoduchost a srozumitelnost vodopádového modelu

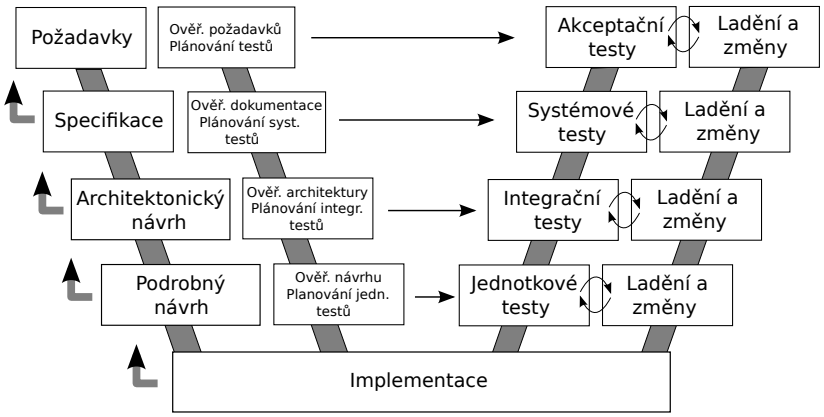
Levá část

- vývojové aktivity
- plánování testů

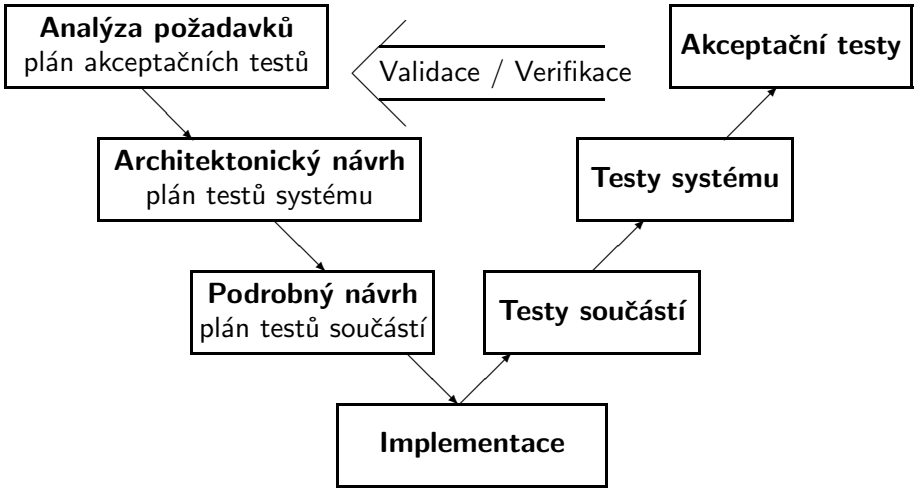
Pravá část

- testovací aktivity
- provádění testů podle plánů

W-model



V-model



Inkrementální model

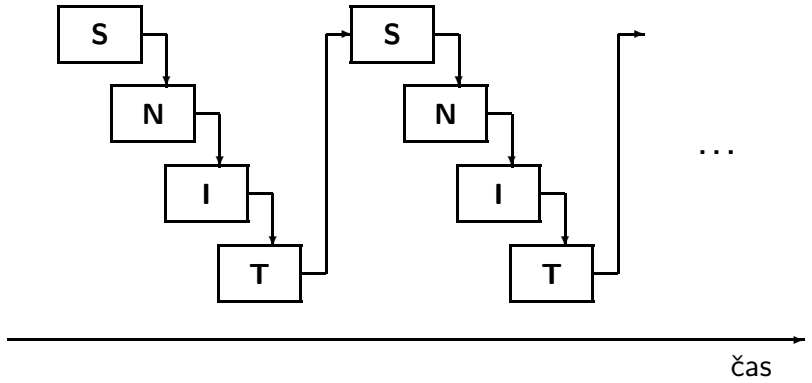
Vlastnosti

- na základě specifikace celého systému se stanoví ucelené části systému
- možnosti
  - série vodopádů, každý pro jednu část systému; vodopád je dokončen před dalším přírůstkem; je možné předávat uživateli po částech
  - počáteční analýza, specifikace a návrh jsou provedeny vodopádem; následuje iterativní přístup kombinovaný s prototypováním; systém se vyvíjí postupně, v každé další verzi je systém rozšířen
  - ...
- zjednodušení zavedení změn během vývoje, omezení projektových rizik
- vývoj po částech může vést ke ztrátě vnímání logiky a technických požadavků celého systému
- zvýšená pozornost musí být věnována rozhraním modulů (částí)

Iterativní modely životního cyklu

Iterativní modely

- sekvence etap se v životním cyklu opakuje



Spirálový model

Vlastnosti

- Barry Boehm, *A Spiral Model of Software Development and Enhancement*, 1986
- kombinace prototypování a analýzy rizik
- vyžaduje stálou spolupráci se zákazníky
- přístupy řízené riziky (*risk-driven approach*)

Proces vývoje

- vývoj je rozdělen na cykly, v každém cyklu se řeší ucelená část vývoje
- každý cyklus je rozdělen na kvadranty vymezující zásadní činnosti, které se mohou opakovat v každém následujícím cyklu (stanovení cílů, vyhodnocení, plánování)
- postupně se každým cyklem rozšiřuje množina zvládnutých problémů ⇒ vývoj postupuje po spirále

Iterativní modely životního cyklu

Vlastnosti

- systém se vyvíjí v iteracích
- v každé iteraci se vytvoří reálný výsledek
- zákazník se účastní vývoje (předpoklad)

Výhody

- v každé iteraci se vytvoří reálný výsledek ⇒ zákazník má možnost validovat výsledek se svými požadavky, rychlejší odhalení chyb ve specifikaci

Nevýhody

- náročnější na řízení
- potenciálně horší výsledná struktura
  ⇒ existují techniky, jak tento nedostatek zmírnit (např. refaktorizace)

Spirálový model

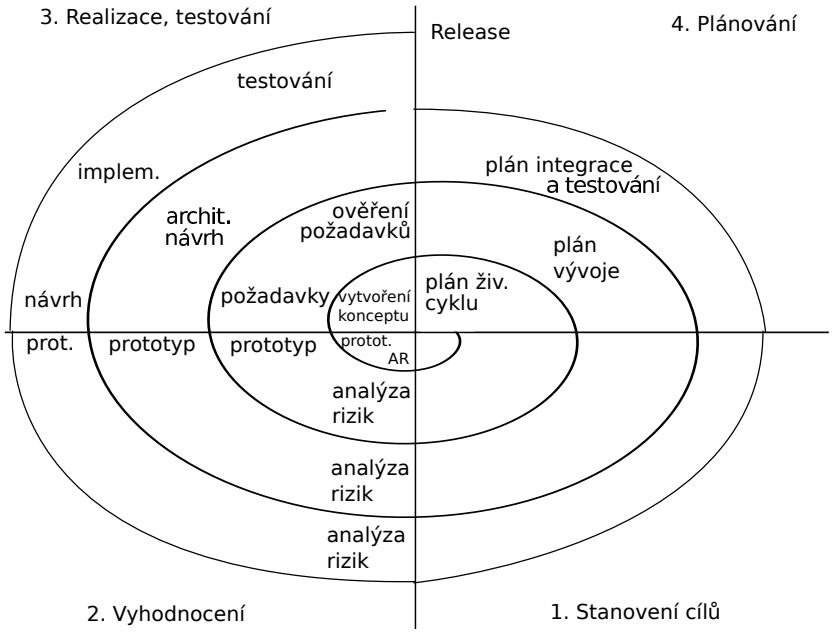
Analýza rizik: Jaké jsou cíle

- zjistit možná ohrožení průběhu projektu
- připravit reakce na tato rizika
- rizika se identifikují a analyzují v každé fázi vývoje
  $\Rightarrow$ včasné vyloučení nevhodných řešení

Analýza rizik: Jaká mohou být rizika

- projektová: odchod lidí, snížení rozpočtu, ...
- technická: neznámé technologie, selhání hardwaru, ...
- obchodní: špatný odhad zájmu, ...

Spirálový model



Spirálový model

Mezníky (Milestones)

- Life Cycle Objectives (LCO): po 2. cyklu
  - vyhodnocení záměrů/cílů projektu, rozhodnutí o pokračování projektu
  - *všechny požadavky podchyceny, stejné chápání požadavků*
  - *cena, plán, priority apod. odpovídají záměrům*
  - *jsou identifikována rizika a procesy pro jejich odstranění/zmírnění*
- Life Cycle Architecture (LCA): po 3. cyklu
  - vyhodnocení výběru architektury, řešení závažných rizik, ...
  - *požadavky a architektura jsou stabilní*
  - *osvědčené postupy testování a vyhodnocování*
- Initial Operation Capability (IOC): po 4. cyklu
  - systém je připraven na distribuci pro uživatelské testování
  - *stabilní verze schopná testového nasazení u zákazníka/uživatele*
  - *srovnání plánovaných a skutečných výdajů a použitých zdrojů*

Spirálový model

Význam cyklů (*počet cyklů není pevně stanoven*)

- první: globální rizika, základní koncept vývoje, volba metod a nástrojů
- druhý: vytváření a ověřování specifikace požadavků
- třetí: vytvoření a ověření návrhu
- čtvrtý: implementace, testování a integrace

Úvodní fáze každého cyklu identifikuje

- cíle cyklu: např. výkonnostní požadavky, funkcionální požadavky apod.
- alternativy: různé způsoby řešení cílů
- omezující podmínky: např. cena, plán projektu
- *následně se vyhodnocuje (analýza rizik, prototypování, simulace, ...)*

Metodika Rational Unified Process – RUP

Šest základních praktik

- využívání existujících komponent,
- vývoj softwarového produktu iteračním způsobem,
  - verze systému, po každé iteraci spustitelný kód
- model softwarového systému je vizualizován,
  - UML, ...
- průběžná kontrola kvality produktu,
  - objektivní měření, metriky, ...
- správa požadavků na softwarový systém,
  - umění získávání požadavků od zákazníka
- řízení změn systému
  - každá změna je přijatelná, všechny změny jsou sledovatelné

Spirálový model

Výhody

- komplexní model vhodný pro složité projekty
- chyby a nevyhovující postupy jsou odhaleny dříve (analýza rizik)
- nezávislost na metodice či strategii

Nevýhody

- závislý na analýze rizik — musí být prováděna na vysoké odborné úrovni
- vyžaduje precizní kontroly výstupů, zkušené členy týmu
- software je k dispozici až po posledním cyklu (*lze vyřešit použitím většího počtu implementačních cyklů*)
- problematické je přesné plánování termínů a cen

Metodika RUP – základní elementy

Pracovníci a role (*kdo*)

- chování je popsáno pomocí činností
- důležitá je *role*: analytik, návrhář, ...

Činnosti – *Activities* (*jak*)

- jasně definovaný účel s definovaným výsledkem (meziprodukt)

Meziprodukty – *Artifacts* (*co*)

- výsledky projektu (činností)
- model, dokument, zdrojový kód, ...

Pracovní procesy – *Workflows* (*kdy*)

- definuje posloupnost činností a interakce mezi pracovníky
- RUP definuje 9 klíčových procesů: např. *Specifikace požadavků*, *Implementace*

Metodika Rational Unified Process – RUP

Co je RUP

- výsledek výzkumu zkušeností řady velkých firem koordinovaný firmou Rational Software, 1997
- první kniha *The Unified Software Development Process*, 1999
- od roku 2003 je Rational Software součástí IBM
- spíše než konkrétní metodika je chápán jako rozšiřitelný framework, který by měl být uzpůsoben organizaci či projektu (*customizable framework*)
- komerční produkt, dodávaný společně s nástroji

Základní vlastnosti

- objektově orientovaná metodika
- *iterativní vývoj*
- přístupy řízené případy užití (*use-case-driven approach*)
- věnuje se všem otázkám procesu tvorby softwaru (*kdo*, *co*, *kdy* a *jak*)

Metodika RUP – vývojový cyklus

Fáze cyklu

- zahájení (*inception*)
  - rozsah projektu, náklady, základní rizika, základní UC, ...
  - 1–2 iterace
- projektování (*elaboration*)
  - plánování, specifikace požadavků, architektura, analýza rizik, ...
  - zpravidla 2 iterace, může až 4 iterace
- realizace (*construction*)
  - kompletace analýzy a návrhu, implementace, hodnocení výstupů, ...
  - 2-4 iterace
- předání (*transition*)
  - dodání, školení, podpora při zavádění, ...
  - alespoň 2 iterace (betaverze, plná verze)

Metodika RUP – vývojový cyklus

Vývojové cykly

- *Initial Development Cycle* – výsledkem je funkční softwarový produkt
- *Evolution Cycles* – další vývoj, verze, ...

Základní cyklus

- je rozdělen na čtyři fáze
  - zahájení (*inception*) ... 10%
  - projektování (*elaboration*) ... 30%
  - realizace (*construction*) ... 50%
  - předání (*transition*) ... 10%
- každá fáze je rozdělena na iterace
  - délka jedné iterace 2 až 6 týdnů

Metodika RUP – mezníky (Milestones)

Mezníky (Milestones)

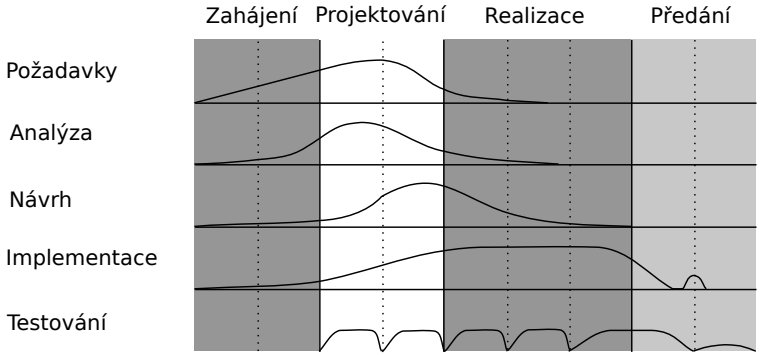
- převzaté ze *Spirálového modelu*
  - **Life Cycle Objectives**
    vyhodnocení záměrů/cílů projektu, rozhodnutí o pokračování projektu
  - **Life Cycle Architecture**
    vyhodnocení výběru architektury, řešení závažných rizik, ...
  - **Initial Operation Capability**
    systém je připraven na distribuci pro uživatelské testování

\+ **Product Release**

- rozhodnutí, zda byly záměry projektu splněny a zda pokračovat v dalším vývojovém cyklu
- *je uživatel spokojen*, *odpovídají náklady plánu*, ...

Metodika RUP – model životního cyklu

- iterativní model jednoho vývojového cyklu
- etapy (pracovní procesy) se překrývají (souběžné provádění)



Rapid Application Development (RAD)

Výhody

- flexibilita, schopnost rychlé změny návrhu podle požadavků zákazníka
- více projektů splňuje termíny a ceny (úspora času, peněz a lidských zdrojů)
- vyšší kvalita zpracování *business potřeb* (prototypování)

Nevýhody

- nižší kvalita návrhu, problém s udržovatelností
- flexibilita vede k menší míře kontroly nad změnami
- projekt může skončit s více požadavky, než je nutné (problém s udržovatelností)

Metodika RUP – zhodnocení

Výhody

- robustní, vhodný pro velkou škálu projektů
- iterativní přístup, včasné odhalení rizik, správa změn, ...
- detailní propracovanost
- vazba na UML

Nevýhody

- detailní propracovanost: u menších projektů značná zátěž na zkoumání metodiky; vývoj může postrádat efektivitu
- komerční produkt (obsahuje hodně podpůrných nástrojů)

Další přístupy k procesu vývoje softwaru

Unified Software Development Process (zjednodušeně UP)

- stejné principy a myšlenky jako RUP, není komerční, nenabízí nástroje
- není tak detailně rozpracována, např. pouze 5 pracovních procesů

Modifikované verze vodopádu

- možnost prolínání etap
- vodopád s podprojekty
- ...

Agilní přístupy (metodiky)

- skupina metodik s odlišným přístupem k procesu tvorby softwarového produktu

Metodika Rapid Application Development

Vlastnosti RAD

- James Martin, *Rapid Application Development*, 1991
- rychlý iterativní vývoj prototypů
- funkční verze jsou k dispozici dříve než u předchozích přístupů
- intenzivní zapojení zákazníka/uživatele do vývojového procesu
- zaměřuje se na splnění *business potřeb* (potřeby a požadavky zákazníka), technologické a inženýrské kvality mají menší důležitost
- určen pro menší až středně velké projekty

Fáze (přehled)

- Plánování: rozsah projektu, omezení, systémové požadavky, ...
- Návrh: modelování, prototypování, využívání CASE nástrojů, ...
- Provedení: pokračování návrhu, kódování, integrace, testování, ...
- Uzavření a nasazení: příprava dat, finální testování, přechod zákazníka na nový systém, zaškolení uživatelů, ...

Predikovatelnost procesu vývoje

Prediktivní přístupy

- plánují velké části softwarových procesů velmi detailně pro dlouhý časový úsek
- projekty vyžadující mnoho procedur, času, velké týmy a **stabilní požadavky**

Problém většiny projektů je,
že se požadavky neustále mění.

Predikovatelnost procesu vývoje

- dobrá predikovatelnost procesu vývoje
  - projekty s jasnými a stabilními požadavky
  - např. projekty NASA, ...
- špatná predikovatelnost procesu vývoje
  - projekty s požadavky, které se v čase mění
    - změna okolních podmínek, účelu softwaru...
    - zákazník si požadavky ujasňuje v průběhu vývoje

Heavyweight a Agilní metodiky

Heavyweight methods

- častá kritika „byrokratizace“ metodik – příliš mnoho aktivit, které jsou předepisovány, způsobuje snížení efektivity celého procesu vývoje
- člen vývojového týmu sleduje přesně postup, krok po kroku

Lightweight methods

- nová skupina metodik, dnes nazývána **agile methods** (agilní metodiky)
- kompromis mezi chaotickým přístupem bez procesů (žádná metodika) a přístupem s mnoha procesy (heavyweight metodiky)
- definují základní rámec vývoje, termíny (*mile-stones*), předpokládané výstupy, techniky, ...
- *agilní* = *čilý, aktivní* ⇒ člen vývojového týmu používá procesy *aktivně*, tj. sám přizpůsobuje procesy a techniky potřebám projektu a týmu

Adaptivní přístupy k procesům vývoje

Adaptivní přístupy

- plánují s přiměřenou mírou detailu
- plány se v průběhu procesu vývoje revidují
- jak řídit adaptivní procesy? ⇒ iterativní přístup

Iterativní přístup a plánování procesů

- jedna iterace většinou zahrnuje základní etapy, může se měnit podle zvolené metodiky
- v první iteraci se provádí plánování procesů, tento plán se v dalších iteracích upravuje podle reálného stavu
- otázka délky iterace (týdny, měsíce, ...), určení mile-stones

Heavyweight a Agilní metodiky

Srovnání vlastností

	Heavyweight	Agilní
přístup	**prediktivní**	**adaptivní**
velikost projektu	velká	malá
velikost týmu	velká	malá (kreativní)
styl řízení	centralizovaný	decentralizovaný
	příkaz-kontrola	vedení–spolupráce
dokumentace	**velký objem**	**malý objem**
zdůraznění (důraz na)	**process-oriented**	**people-oriented**
fixní kritéria	*funkcionalita*	*čas a zdroje*
proměnná kritéria	*čas a zdroje*	*funkcionalita*

Agilní metodiky

Základní teze

- Minimum formálních a byrokratických artefaktů.
  - *důležitou součástí dokumentace je i zdrojový kód*
- Člen týmu je schopen rozhodovat technické otázky své práce.
  - *důraz na složení týmu a komunikaci uvnitř týmu*
  - *komunikace jako jedna z forem vývoje*
  - *techniky vyžadující komunikaci*
- Ověření správnosti navrženého systému zpětnou vazbou
  - *iterativní inkrementální vývoj, časté uvolňování průběžných verzí*
  - *předložit zákazníkovi a na základě zpětné vazby upravovat*
  - *zákazník je členem vývojového týmu*

Process-oriented přístupy

Předpoklady

- lidé jsou zdroje, které jsou dostupné v několika rolích: analytik, programátor, tester, manažer, ...
- procesy by měly fungovat za všech okolností (změna týmu, ...)

Důsledky

- podstatná je role, nikoliv individualita lidí
  - není důležité *jaké* analytiky máte, ale *kolik* jich máte
  - člověk je predikovatelná (a tedy jednoduše nahraditelná) komponenta vývojového procesu
- procesy by měly fungovat za všech okolností
  $\Rightarrow$ velký objem procesů, detailní specifikace procesů, velká míra režie
- za standardní prostředek komunikace se považuje dokumentace
  $\Rightarrow$ zvýšení režie

Agilní metodiky

Základní teze

- Důraz na rigorózní, průběžné a automatizované testování.
  - *zejména kvůli neustálým změnám v kódu i návrhu*
- Princip jednoduchosti
  - *návrh odráží aktuální potřeby uživatele*
  - *do systému vložíme to, co potřebujeme, když to potřebujeme*

People-oriented přístupy

> Design and programming are human activities; forget that and all is lost.
> Bjarne Stroustrup, 1991

Předpoklady

- lidé nepracují konzistentně v průběhu času
  - pokud by člověk dostal každý den stejný úkol, vytvoří *podobné* výsledky, ale nikdy ne *stejné*
  - schopnost pracovního nasazení/soustředění se mění
- lidé jsou komunikující bytosti
  - fyzická blízkost – gestikulace, hlasový projev, intonace
  - otázky a odpovědi v reálném čase
- žádný proces nikdy nevytváří dovednosti (znalosti) vývojového týmu

Důsledky

- podstatná je individualita lidí
  - důležitá je kvalita a osobní rozvoj členů týmu
  - role člena týmu se může měnit
  - kvalitní člen týmu je hůře nahraditelný

XP: Charakteristické vlastnosti

Komunikace

- *Teze: programátoři, zákazníci a manažeři musí spolu komunikovat*
- XP využívá takové techniky, které komunikaci vyžadují (testování, párové programování, odhady úkolů)
- *člen týmu (kouč)*, který udržuje komunikační toky, pomáhá programátorům s technickými dovednostmi, komunikuje s manažery na vyšších úrovních
- *člen týmu (velký šéf)*, který provádí zásadní rozhodnutí

Zpětná vazba

- *Teze: stav a kvalita vývoje se nejlépe zjistí od zákazníka a testováním*
- snaha mít co nejdříve implementované nejdůležitější části systému, nejlépe nasazené přímo v provozu
- odpovědí na otázku „Funguje to?“ je testování
- zpětná vazba musí být rychlá
- *člen týmu* (zákazník), který vyhodnocuje dosaženou funkcionalitu

Agilní metodiky

Metodiky označované jako agilní

- Extreme programming (XP)
- Crystal
- Scrum
- Feature Driven Development (FDD)
- Test Driven Development (TDD)
- Dynamic System Development Method (DSDM)
- ...

XP: Charakteristické vlastnosti

Jednoduchost

- *Teze: Jednoduché věci se realizují a upravují rychleji s menším počtem chyb*
- je dobré udržovat si přehled o tom, co bude
- ale je nutné soustředit se na to, co je potřeba právě teď
- jednoduché věci je třeba vytvářet jednoduše
  $\Rightarrow$ úspora času na opravdu složité věci
- v případě potřeby není problém jednoduché věci rozšířit

Odvaha

- *Teze: pokud je to potřeba, nebát se provést zásadní změny*, a to i za cenu dočasného snížení úspěšnosti testů a tedy zvýšeného úsilí
  - nebát se zahodit naprogramovaný kód
  - nebát se zkusit neznámé (když nevíš, že to nejde, může se to podařit)

Extrémní programování (XP)

Kořeny XP

- Kent Beck, Ward Cunningham
- 80. léta – Smalltalk
- 90. léta – získávání zkušeností v různých projektech, rozšiřování idejí agilního přístupu

Reference

- `http://www.extremeprogramming.org`
- *Beck, K., and Andres, C., Extreme Programming Explained: Embrace Change, 2nd ed. Addison-Wesley, 2004*
- *Ambler, S. W., AM Throughout the XP Lifecycle*
  `http://www.agilemodeling.com/essays/agileModelingXPLifecycle.htm`

XP: Základní techniky

Metriky

- důležitá součást určení kvality softwarových procesů
- např. poměr plánovaného času a skutečného času
- přiměřený počet metrik (3 – 4)
- pokud přestane metrika plnit svůj účel ⇒ nahradit jinou
  např. metrika testů funkcionality se blíží 100% ⇒ nahradit jinou s menší úspěšností
- existují pravidla udávající, kdy a jak často by se měly jednotlivé techniky používat

Motivace vývojářů

- lidé lépe pracují, pokud je práce baví
- jídlo, hračky, vybavení pracoviště, ...

XP: Základní techniky

Přírůstkové (malé) změny

- návrh a implementace se mění v čase jen pozvolna
- uvolňování malých verzí systému (nejpodstatnější požadavky, postupně vylepšované a doplňované)

Testování

- *Co nelze otestovat, to neexistuje.*
- ke každé funkci píšeme testy, někdy i před tím, než začneme programovat
- zautomatizovaný systém testů
- jednotkové i integrační testování

XP: Proces vývoje

1. **Zkoumání** (*Exploration*)
   - tvorba vysokoúrovňových požadavků
   - tvorba základního návrhu prostřednictvím prototypů
2. **Plánování** (*Planning*)
   - odhad času, výběr minimální možné množiny požadavků, ...
   - plánování iterací
3. **Iterace** (*Iterations to First Release*)
   - iterativní vývoj s využitím specifických pravidel a technik XP
4. **Produkce** (*Productionizing /to put into production/*)
   - verifikace a validace, nasazení systému
5. **Údržba** (*Maintenance*)
   - implementace zbývajících požadavků a nových potřeb do běžícího systému (iterování fází 2 až 4)
6. **Uzavření** (*Death*)
   - uzavření a zhodnocení projektu

XP: Základní techniky

Párové programování

- jednu věc programují vždy 2 programátoři (ale pouze 1 skutečně píše)
- ten, kdo píše, se soustřeďuje na nejlepší způsob implementace problému
- druhý se soustřeďuje na problém z globálnějšího pohledu
  bude to fungovat, jaké další testy, možnost zjednodušení, ...
- páry jsou dynamické

Refaktorizace

- úprava stávajícího programu – zjednodušení, zefektivnění návrhu
- odstranění (úprava) nepotřebných částí
- změna architektury (pravidlo přírůstkové změny)
- *při refaktorizaci se nemění funkcionalita!*

XP: Collective-Code-Ownership

Collective-Code-Ownership

- aplikováno na procesy analýzy, návrhu, programování, testování a integrace
- sdílení kódu, kdokoliv má možnost měnit kód (nová funkcionalita, odstranění chyb, refaktorizace)
- nutnost zavést principy *test-driven development*
  - ke každému kódu musí existovat jednotkové testy (*unit tests*)
  - testy se sdružují do sady (*test suite*)
  - při každé změně kódu musí být provedena (automatizovaně) sada testů
  - vkládání nového kódu (integrace) je chráněno sadami testů (nový kód musí testy projít)

XP: Proces vývoje

Zkoumání (Exploration)

- Utváření týmu
- Návrh počáteční množiny *User Stories*. *User Story*
  - definuje vlastnost systému z pohledu zákazníka/uživatele
  - je psána uživatelem terminologií problémové domény
- Tvorba systémových metafor (*Metaphor)* a prototypů (*Spikes*)
  - prototypy pomáhají definovat metafory
  - metafora je jednoduchý popis jak má systém pracovat, srozumitelný pro všechny členy týmu ⇒ pochopení požadavků a architektury

Development Engine

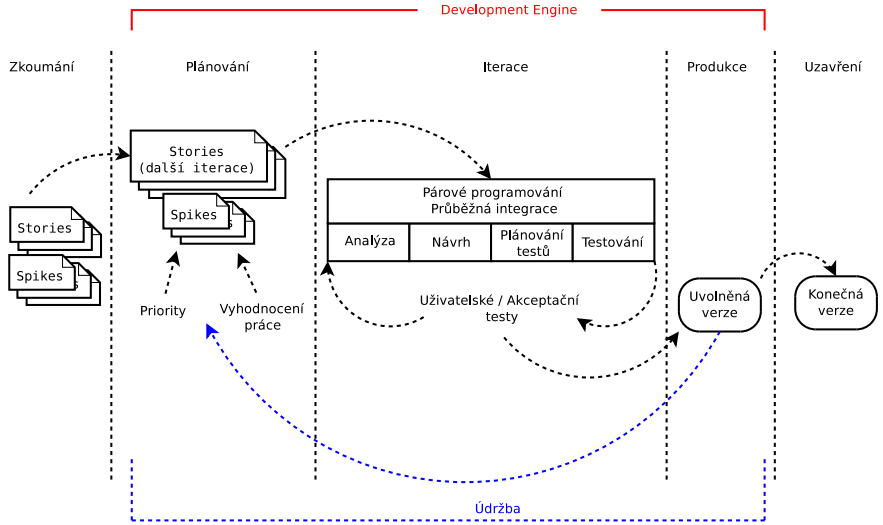
- aktivity spojené s fázemi *Plánování*, *Iterace* a *Produkce*.
- každý jeden běh těchto fází produkuje novou verzi
- v jednom běhu *enginu* probíhá vývoj v iteracích (fáze *Iterace*)
- po prvním nasazení (fáze *Produkce*) se kroky fáze *Údržba* provádějí iteracemi v *enginu*

XP: Collective-Code-Ownership

Základní techniky

- využívání CRC karet
  – nalezení nejjednodušího návrhu
- jednotný styl programování (standard)
  – zlepšení komunikace
- vývojáři se postupně účastní všech prací
  – získání znalostí o všech částech systému
- vývojáři mají pracovat udržitelným tempem; nikdo nesmí pracovat přes čas příliš dlouho
- párové programování
- průběžné provádění refaktorizace
- *průběžná integrace* (*continuous integration*)

XP: Proces vývoje



Studijní koutek – IT a studium (1/4)

Naše postřehy

- Úroveň přijatých studentů na FIT (měřeno percentilem Národních srovnávacích zkoušek od SCIO) v posledních letech roste.
- Přesto se studijní výsledky našich studentů spíše zhoršují.
- To nedává smysl! Někde musí být chyba! Ale kde?

Práce při studiu

- Jistě, když jsou studenti v práci, nemohou být na přednáškách.
- Apelujeme na studenty, aby rozumně vyvážili práci a studium. Nejen prostřednictvím projektové praxe nabízíme studentům možnost vydělat si v našich projektech.
- Naši firemní partneři se musí zavázat, že budou studentům poskytovat dostatečný prostor pro studium. Ostatní firmy ke studentům nepouštíme.
- Je to ale skutečná příčina? I dříve studenti při studiu pracovali či podnikali a na studijních výsledcích se to neprojevilo.

Studijní koutek – IT a studium (2/4)

Digitální média

- Sociální sítě vytvářejí závislost prostřednictvím dopaminové vazby. Čím je uživatel déle připojen, tím víc reklamy se prodá.
- Mladí lidé ztrácejí schopnost navazovat reálné vztahy. Mají podstatně méně sexu než předchozí generace.
- Klesá doba, po kterou je mozek schopen se soustředit.
- Průměrná inteligence přestala růst a začala klesat.
- Zato roste počet lidí s úzkostmi a depresemi.
- Na lékařské fakulty přichází gramlaví medici.
- Závislost na moderních médiích vznikne snáze než závislost na alkoholu.

XP: Průběžná integrace

Co je průběžná integrace

- automatizované a reprodukovatelné sestavování (*build*)
- obsahuje automatizované testování, které probíhá mnohokrát za den
- umožňuje průběžně integrovat změny a tím redukovat problémy s integrací

Základní procesy průběžné integrace

- integrace zdrojového kódu
  - sdílené repozitáře, ...
- automatizovaná správa sestavování (*build management*)
  - sestavování se provádí často, několikrát za den
  - sestavení se provádí při změně kódu, v naplánovaném čase, ...
  - vývojář musí být informován o výsledku
- automatizované ověřování (testování)
  - po sestavení je nutno ověřit, že nová verze splňuje všechny testy

XP: Vyhodnocení

Silné stránky

- iterativní inkrementální proces
- proces se *ladí* na základě zpětné vazby
- požadavky se *ladí* během celého vývoje
- průběžná integrace
- zapojení uživatelů
- vývoj založený na testování

Slabé stránky

- definuje rámec, principy a praktiky, ale nedefinuje přesný postup
- nepředepisuje modely pro návrh, často se od *User Stories* a *Metaphor* přechází na implementaci
- hůře akceptovatelný pro vývojáře – vyžaduje striktní dodržování základních principů a procesů

Studijní koutek – IT a studium (3/4)

Co s tím dělá společnost?

- osvěta (např. MUDr. Martin Jan Stránský)
- zákazy mobilních telefonů ve školách (např. Francie)
- úprava Mezinárodní statistické klasifikace nemocí
- budování center pro léčení závislosti na internetu (např. Německo)
  - počítačové (zejména on-line) hry
  - on-line komunikace (např. sociální sítě)
  - stránky s pornografickým obsahem
  - získávání co největšího množství informací

Studijní koutek – IT a studium (4/4)

A co s tím můžete dělat vy?

- Dbejte na dostatek spánku.
  Je důležitý pro utřídění informací a zahození „smetí“.
- Využívejte fyzická média (papír).
  Zlepšují učení (rámování). Aktivují více center v mozku současně.
- Nenoste si mobily a notebooky na přednášky.
  Rozptylují Vaši pozornost (i vypnuté).
- Probíranou látku si zopakujte.
  Různé úhly pohledu na téma pomáhají dlouhodobějšímu zapamatování.
- Bojujte se svými závislostmi.
  Vydržíte třeba dva týdny bez internetu a mobilu (detox)?