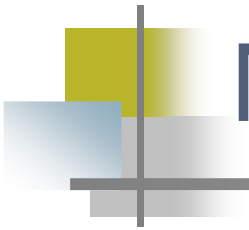
KIV/ASWI 2007/2008

Agilní přístup k tvorbě software

Simplicity – the art of maximizing the amount of work not *done.*

-- Agile Manifesto, principle 11

- Motivace
- Principy
- Důkazy realističnosti
- Metodiky - XP, SCRUM



Motivace

Kolik je $17 mld?

– tucet komerčních letů na Měsíc [google „project apollo cost"]

– 3x cena majority v ČTc

– výše dotace EU do zemědělství na jeden rok [google „14 miliard EUR"]

– 3/4 nákladů na přechod ČR od centrálně plánované ekonomiky na ekonomiku tržní [MFČR]

– cca 1/2 celkové ceny lunárního programu Apollo [google „project apollo cost"]

Software: Mýty vs Realita

Software není automobil

Změna je život

Dinosauři vyhynuli, myši nikoli

Sjíždět vodopád je nebezpečné

Mýty softwarových projektů

- Zákazník ví, co chce
  - pevné vlastnosti produktu
  - předem známý cílový stav
- Dodavatel ví, jak na to
  - predikovatelný postup, náklady, kvalita
  - lineární škálování složitosti projektu

- Tyto předpoklady – platné pro sériovou výrobu – používá příliš mnoho softwarových procesů (a manažerů a klientů)
  - založené na zjednodušeném, a idealizovaném, vodopádovém modelu

Tvorba SW není sériová výroba

Software není automobil

- Sériová výroba
  - » CDčka, koloběžky, pračky, mobily, auta, paneláky
  - pevné a předem známé specifikace, známý cíl
  - známý výrobní postup, přesné odhady na začátku
  - malá míra variability a nutnost reakce na změny
  - problémem je logistika a ekonomie výroby kopií

- Tvorba software nemá (ve většině případů) charakter předvídatelného projektu a/nebo sériové výroby. Naopak: jde o vývoj nového (typu) produktu.
  - » studie vozu, ekologický dům, raketoplán
  - produkt a projekt jedinečný, bez vzoru a modelu

Změny jsou pravidlem, ne výjimkou

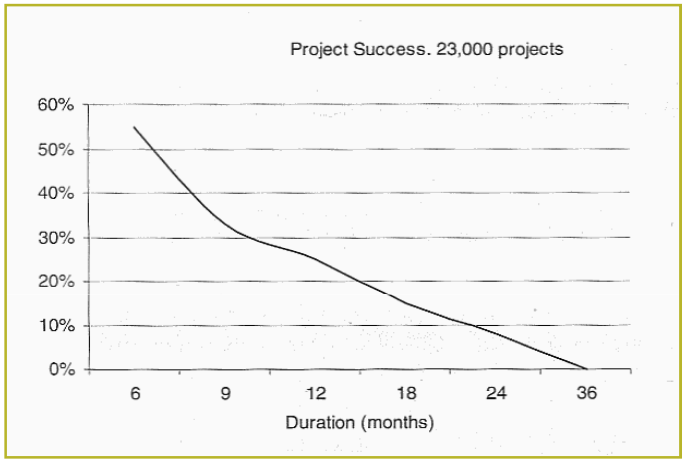
Změna je život

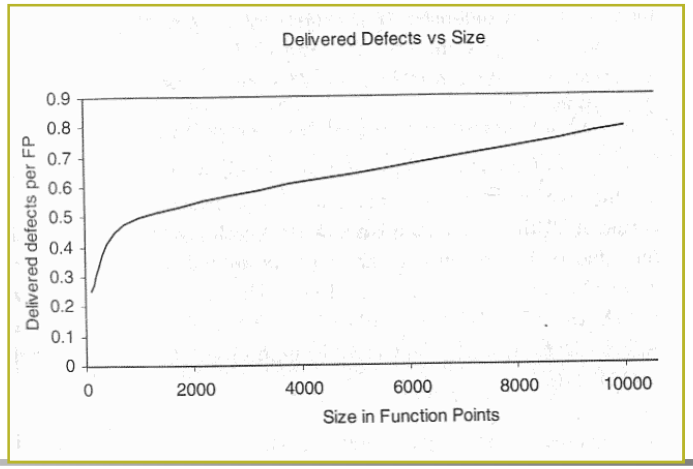
- Změny požadavků: spekulativní funkce/vlastnosti, fenomén IKIWISI, nedostatečná komunikace, ...
  - » „Zákazník neví, co chce, neumí to říct, ale chce to“
  - » typicky se změní zhruba 25% specifikovaných požadavků [Boehm88]
- Změny prostředí: legislativní rámec, akvizice firmy, upgrade systémů zákazníka, nové technologie, ...
- Změny postupu: fluktuace v týmu, chybná architektonická rozhodnutí, změny nástrojů, ...

Velikost pracuje proti nám

Dinosauři vyhynuli, myši nikoli

- Úspěšnost
  - velké plány, velká zklamání
    - » přes 1/2 velkých zrušeno
  - potvrzováno teorií systémů
- Produktivita
  - nepřímá úměra k velikosti produktu
    - » větší pro malé přírůstky, týmy
- Četnost změn
  - 10% malé projekty (100 FP)
  - 35% velké (10000 FP)
- Kvalita
  - nepřímá úměra k velikosti produktu
  - prototyp se 40% → 20% funkčnosti
    ⇒ pokles chybovosti o 10/měs/MLOC



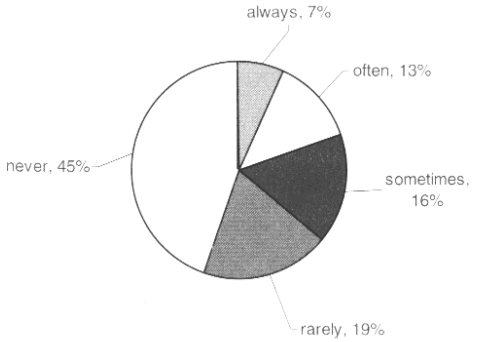


Zjednodušené modely nefungují

Sjíždět vodopád je nebezpečné

- „Pro každý složitý problém existuje řešení, které je jednoduché, elegantní, a špatné" [H.Mencken] – například vodopádový model v „klasickém" vydání ...
  - 4 z 5 faktorů neúspěchu projektů jsou spojeny s VM [Jones95]
  - použití VM nejvíce přispívá ke krachu projektů v 80% [Thomas01]
  - expertní doporučení je vyhnout se VM [Brooks87]
- Například
  - studie DoD 1995: ze systémů za celkem $37 mld, vyvíjených podle DOD-STD-2167, jich 46% nebylo nikdy nepoužito
  - US ATC projekt 1983-1994: vodopád, velký třesk, $2.6 mld, zrušeno
  - Johnson02: využití předem specifikovaných požadavků

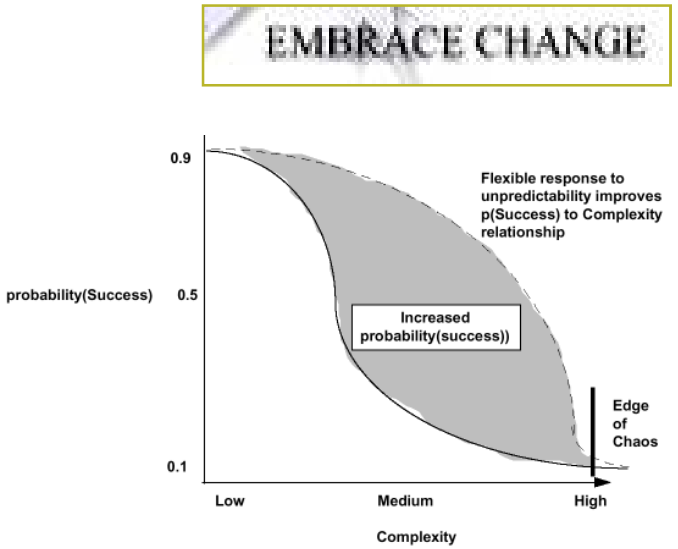
Kolik je $17 mld?



Řešení

- Přivítat změnu
  - » opustit to, co nefunguje – přístup vodopádu

- Iterativní a evoluční vývoj
- Adaptivní plánování
- Agilní přístup



Přístup iterativně-evoluční

Q: Jaké můžeme v nejbližší době čekat nové, vzrušující a slibné myšlenky nebo techniky v oblasti software?

A: Myslím, že [nejslibnější myšlenky] jsou už léta známy, jen nejsou správně používány.

– David Parnas

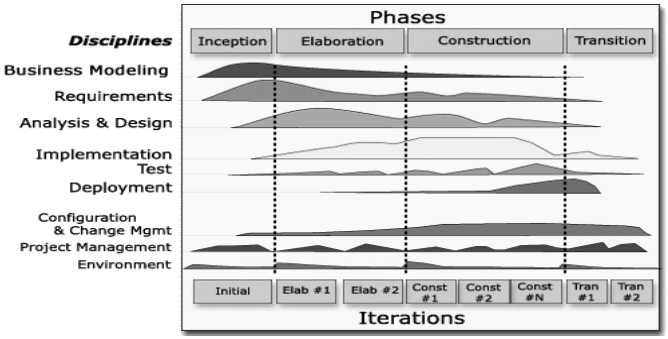
Přehled

- Iterativní vývoj
  - včasná reakce na problémy při vývoji
- Evoluční (adaptivní) dodávky a plánování
  - podchycení změn požadavků
- Empirický proces
  - adaptace na změny týmu a postupu

Iterativní vývoj

- Rámcový plán životního cyklu
  - » milníky – např. RUP

- Řetězec vývojových iterací
  - miniaturní úplný projekt – cca vodopádový model
  - cíl: iterační release (interní)
    - » produkt funkčně neúplný
    - » ale otestovaný a funkční
  - vede na přírůstkový vývoj
  - nevylučuje kompletní počáteční specifikaci požadavků



Časté, krátké, uzavřené iterace

- Délka iterací
  - pevně daná (SCRUM)
  - variabilní v okamžiku plánování (XP, UP, …)
  - malá – blízký cíl, menší složitost/riziko, rychlá adaptace
    - » 1-4 týdny pro malé, 3-6 týdnů velké projekty
- Počet
  - dle potřeby, obvykle alespoň 3
- Běžící iterace uzavřená změnám zvenčí
    - » nutné pro stabilitu projektu
  - možný tlak na změnu: čas, funkčnost, postup
  - neakceptovat ani od šéfů (viz SCRUM)

Timeboxing iterací

- Délka ⇒ datum ukončení pevné
  - omezení plánované funkčnosti možné
    - » viz dále plánování
  - nehotový release, změna datumu neakceptovatelné
  - nesmí být tlak na přesčas
- Výhody
  - zacílení iterace
  - lidé si pamatují překročené termíny, ne opuštěné vlastnosti
  - nutí včas k těžkým rozhodnutím a kompromisům
  - vysoká produktivita: 80 vs 25 FP/měs

SCRUM: 30 dní

XP: 1-2 týdny

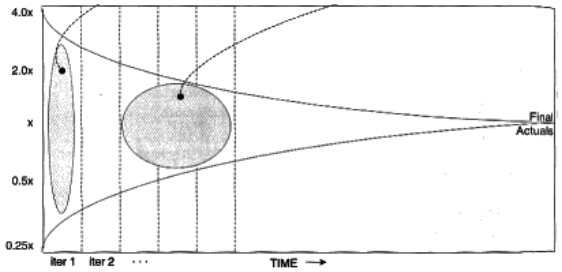
Evoluční a adaptivní vývoj

… jeden z 4 nejčastějších faktorů úspěchu sw projektů

- Evoluční vývoj
  - dotažení iterativního přístupu
  - znalosti o požadavcích, návrhu, odhadech a plánu se vyvíjejí/zpřesňují v průběhu projektu
    - » žádné kompletní, dále neměnné specifikace na začátku (20-80)
    - » míra změny obvykle klesá s postupujícími iteracemi
  - „don't develop software, grow it"
- Adaptivní
  - zdůraznění zpětné vazby v evolučním vývoji
    - » analogie řízení auta
  - zejména evoluční dodávky – zpětná vazba od uživatelů

Adaptivní plánování

- Prediktivní plánování: velká míra nejistoty
  - » „plan work, work plan“
  - neznalost odhadů v době, kdy jsou potřeba
  - měnící se požadavky ⇒ rozsah projektu

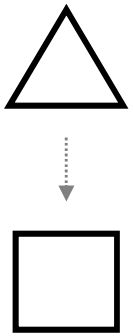


- Řešení
  - přesnější odhady a plán až po několika iteracích
  - detailně plánovat jen na co máme rozumně přesná data
    - » obvykle nejvýše pro následující iteraci
    - » plus hrubé milníky (dodávky zákazníkovi)

Stupně volnosti při plánování

Cheap. Fast. Good.
Choose any two.

- Klasicky: čas, zdroje (cena), kvalita
  - obtížně měnitelné, odhadované
  - kvalita obtížně řiditelná
    - » typický požadavek: „bude to v termínu, s daným rozpočtem, a v bezchybné kvalitě jako vždy“ … „you get crappy SW late“
- Agilně: +funkčnost
  - nejlepší faktor pro řízení projektu
    - » první tři pevné, funkčnost nejsnáze měnitelná
  - vhodná granularita ⇒ snadné a přesné odhady



Riziky a klientem řízený vývoj

- » Kontext: plán iterace (výběr funkčnosti)

- Řízení riziky
  - vyhodnotit rizikové faktory projektu
    - » designová/architektonická rizika, obchodní, legislativní, neznámá funkčnost, použitelnost, …
  - začít s částmi funkčnosti/designu s největší mírou rizika
- Řízení prioritami klienta
  - výběr funkčnosti je na zákaznícovi
    - » množství funkcí omezeno délkou iterace
  - umožňuje pružně reagovat na aktuální potřeby
- Týmové rozhodování: dot voting

Již z dob spirálového modelu (1986)

Empirický proces

» místo definovaného a nařizujícího

- Definovaný proces („rule-based“)
  - předem známé/dané aktivity, jejich návaznost
  - PERT diagram
- Empirický („principle-based“)
  - » uznání, že vývoj software není sériová výroba
  - sada jednoduchých aktivit
  - časté měření procesu a zpětná vazba
  - dynamická adaptace na změny a události
    - » emergent behaviour, samoorganizující tým
  - přizpůsobení typu (závažnosti) projektu

SCRUM: denní setkání týmu, „empowered team“

XP: denní setkání týmu, role „tracker“, plánovací hra

Přístup agilní

Kolik byste zaplatili softwarovému týmu, který by dělal to co chcete vy? Co kdyby vám navíc řekli, kolik to bude stát? Dodali kvalitní produkt, a dávali aktuální přesné informace o stavu projektu? A navíc, co kdybyste mohli kdykoli změnit názor na to, co chcete?

Přehled „hodnot“

- Vše iterativní, evoluční a adaptivní
- Přivítání změny (embrace change)
  - schopnost adaptace na změny je výhodou, ne-li nutností pro přežití
- Komunikace a zpětná vazba
  - co nejvíce mechanismů pro získání podnětů pro změny
- Jednoduchost technik (light touch)
  - komunikačních, programátorských, manažerských

Přivítání změny

- Proč? Vědomá práce se změnami přináší výhody
- Pro zákazníka
  - » implementace změny v požadavcích je výhodou na trhu
  - » možnost pružně měnit chod projektu usnadní nasazení
    - vč. možnosti ukončit „právě teď" pokud release je vyhovující
- Pro vývojáře
  - » modifikace procesu zefektivní práci týmu

Zpětná vazba

- Proč? Nutná pro schopnost adaptovat se
  - snaha získat jí co nejvíc
- Zákazník
  - » problém IKIWISI, neumí specifikovat požadavky
  - » informace o vnější kvalitě produktu (validace)
- Vývojáři
  - » včas zachytit nepředvídatelné události
  - » informace o vnitřní kvalitě produktu
  - » jak vyhovuje proces, jeho efektivita

Komunikace

- Proč? Nejlepší mechanismus pro získání zpětné vazby a kvalitních podkladů pro práci
  - » získání specifikace
  - » zachycení změny
  - » zdroj zpětné vazby
- Preference přímé komunikace

„Za problémy projektů je možno bez výjimky vystopovat chvíle, kdy někdo někomu něco důležitého neřekl" [Beck]

Jednoduchost

- Proč jednoduchost?
  - » pomáhá zaměření na primární cíl
  - » usnadňuje komunikaci
- Zákazník
  - » primární cíl vývoje = dodat kvalitní funkční produkt
- Vývojáři
  - » řízení procesu, techniky, prostředky
    - „light touch“ , odkládání všeho, co nevede přímo k cíli

Techniky naplňující hodnoty

- Jednoduché prostředky
- Intenzivní komunikace
- Člověk středem dění
- Odvaha
- Párové programování
- Test-driven a test-first vývoj
- Refactoring
- Nepřetržitá integrace
- … a další, specifické pro jednotlivé metodiky

Jednoduché prostředky

- „(Use) The Simplest Thing That Could Possibly Work.“
  - » design/kód, dokumentace, mechanismy komunikace, …
  - seznam požadavků na změny: Excel, WikiWiki
  - popis požadavků na funkce: papírové karty
  - statistiky denního buildu: výstup JUnit emailem
  - návrh architektury: ad-hoc obdélníky, tabule + digiťák
  - stav projektu: flipchart se seznamem ~~hotových~~ úkolů
- YAGNI
  - jednoduchost ≠ lajdáckost

Komunikace

- Raději osobní než email a podepsané specifikace
  - » lepší vztah a vyšší zodpovědnost
- On-site zákazník
  - » „kompletní tým" = vývojáři, reprezentant zákazníka, QA
- Denní schůzka týmu
  - » stav projektu, dnešní úkoly, problémy, zkušenosti, …
  - » přebírání práce, deklarování zodpovědnosti
  - » manažer: umožnit práci, nikoli rozdělovat práci
- Otevřený prostor, tabule na čmárání
  - » takže snadno komunikuje kdokoli kdykoli s kýmkoli

Člověk (ne proces) středem dění

- Programování je lidská (ne strojová) činnost
  - potřeba tvořivosti, komunikace, ocenění
  - velká variabilita výkonnosti (1:10)
- Kvalitní a soudržný tým
  - » z talentovaných a schopných jednotlivců
  - » důležitější než komplexní proces
- Rychlé ocenění, lehké varování
  - » programování řízené testy, denní build
- Udržitelné tempo
  - » přesčasy známkou vážných problémů
- Přímá komunikace, jednoduché prostředky

„We methodologists and process designers have been designing complex systems without characterizing the [most important] active components of our systems, known to be highly non-linear and variable: people…“ [A.Cockburn]

Odvaha

- Jednoduchost vyžaduje odvahu
  - zahodit kód, na kterém jsem dělal týden, když nefunguje
  - důvěřovat, že jednoduché řešení je nejlepší
- Řízení, práce na projektu vyžaduje odvahu
  - kompletně změnit plán v další iteraci
  - požádat o pomoc s problémem mladšího kolegu
  - důvěřovat týmu, že vyřeší problém bez direktivního řízení
  - oželet důležitého člena, který se rozhodne odejít
- Přímá komunikace posiluje odvahu

Párové programování

- Pár = řidič + navigátor
  - » společný cíl, plné nasazení
  - » komunikace (dialog), víc hlav víc ví, on-the-fly oponentura
- Řidič
  - řídí (vykonává+udává směr), sleduje situaci, detaily
  - komunikuje, vysvětluje, naslouchá navigátorovi
- Navigátor (partner)
  - pomáhá, dodává odvahu, poskytuje informace
  - kontroluje, opravuje – je „svědomím" páru
- Výhody
  - produktivnější, zábavnější
    - » Jensen 2003: produktivita 175x, chyby 0.001x
    - » Williams et al 2000: produktivita o 40-50% vyšší
- Problémy
  - ne každému vyhovuje

Programování řízené testy

» Test-driven, test-first development

- Zadání → Test(y) → Implementace
  - implementace, která způsobí, že testy nenajdou chybu
  - *test everything that could possibly break*
- Důsledky
  - testy jako specifikace chování, technická dokumentace
  - prověření návrhu před implementací, produktu po implementaci
    - » včasná zpětná vazba
  - rychlá odměna

Refactoring

- Cesta, jak udržet kód zdravý a kvalitní při častých změnách
  - změna kódu bez změny (vnější) funkčnosti
  - cíl: čištění kódu
    - » výsledkem je lepší design ⇒ méně chyb, lepší porozumění
    - » nutné při menší míře úvodního návrhu
  - ověření pomocí jednotkových testů

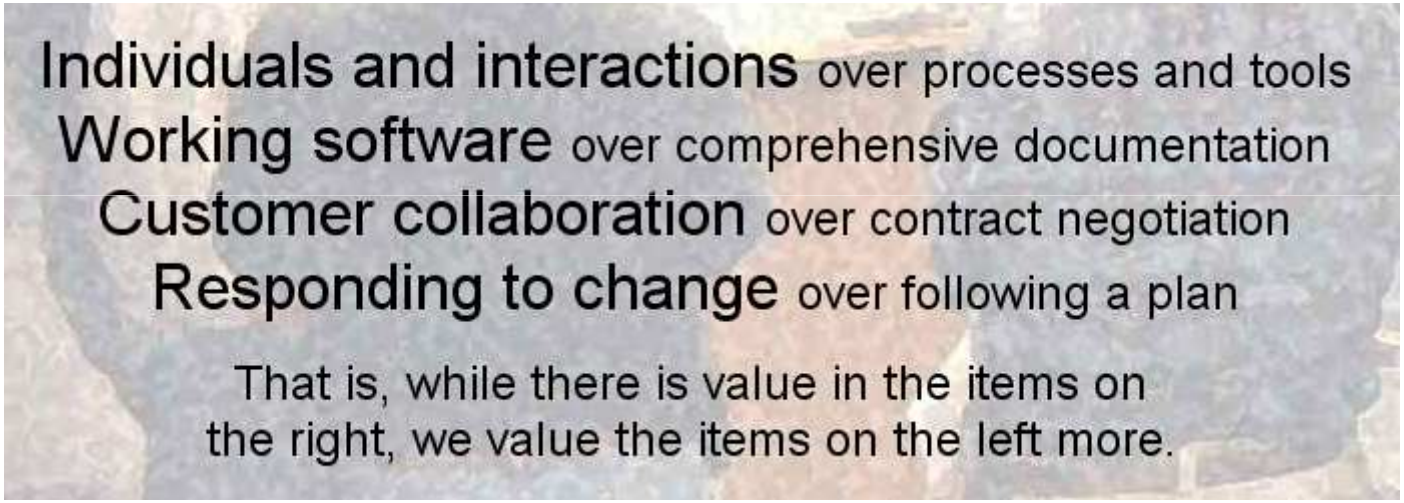
```
switch ($stav) {
case "volna":
    return " AND t.je_prideleno = 0 AND t.d
    break;
case "rezervovana":
    return " AND t.je_prideleno = 0 AND t.d
    break;
...
```

```
switch ($stav) {
case STAV_VOLNO:
    return " AND t.je_pridele
    break;
case STAV_REZERVOVANO:
    return " AND t.je_pridele
    break;
...
```

Nepřetržitá integrace

- Viz předchozí přednáška

Shrnutí



www.agilemanifesto.org

Agilní metodiky

agile ['ædʒaɪl] **(adj)**

1. hbitý, čilý, svižný
2. bystrý

„Definice“ agility

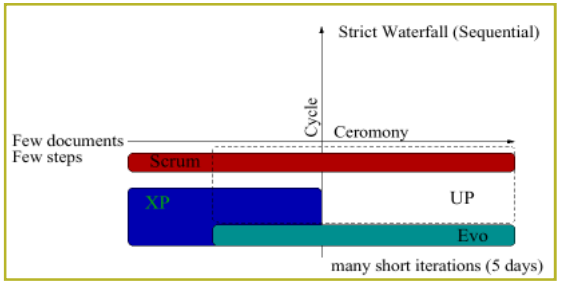
- Použití timeboxovaného iterativního a evolučního vývoje, adaptivního plánování, evolučních dodávek a dalších hodnot a technik, které podporují čilost – rychlou a pružnou odezvu na změny
  - motto: změna je vítána
  - strategie: co největší manévrovatelnost

Seznam metodik

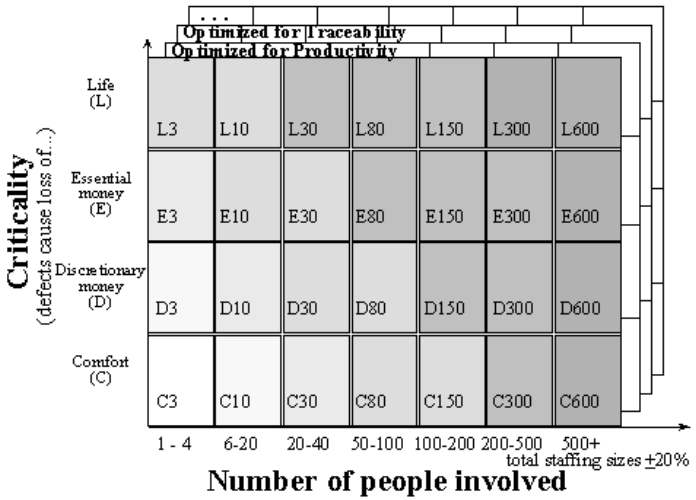
- Nejčastěji používané
  - Extrémní programování (XP), SCRUM
- Mnohé další
  - Feature-Driven Development (FDD) – J.De Luca, P.Coad
  - Adaptive Software Development (ASD) – J.Highsmith
  - Dynamic Solutions Delivery Model (DSDM)
  - Lean Development – Poppendieck
  - EVO – T.Gilb (1976!)
- A také
  - Unified Process (UP)
  - Microsoft Solutions Framework (MSF)
  - Spirálový model

Škálování

- Míra formálnosti („ceremony“)
  - minimalistické – XP, SCRUM
  - umožňující klasické artefakty – UP, SCRUM
- Tep projektu
  - velmi rychlé – EVO (1 týden)
  - delší – SCRUM (30 dní), XP (1-4 týdny)
    - » ale už ne vodopád (bez tepu)



- Velikost a míra kritičnosti projektu
  - Crystal family
    - » projekt D6 velmi rozdílný od L40
  - 1-6 – 20 – 40 – 100 – … členů týmu
  - C(omfort) – D(iscretionary money) – E(ssential) – L(ife)



SCRUM

SCRUM

Podstatné rysy

- Sada hodnot a postupů řízení projektu
  - » nikoli softwarově vývojářských technik
  - » důsledně empirický a adaptivní proces
  - » samoorganizující tým
  - » dobře škálující (C6-L600)

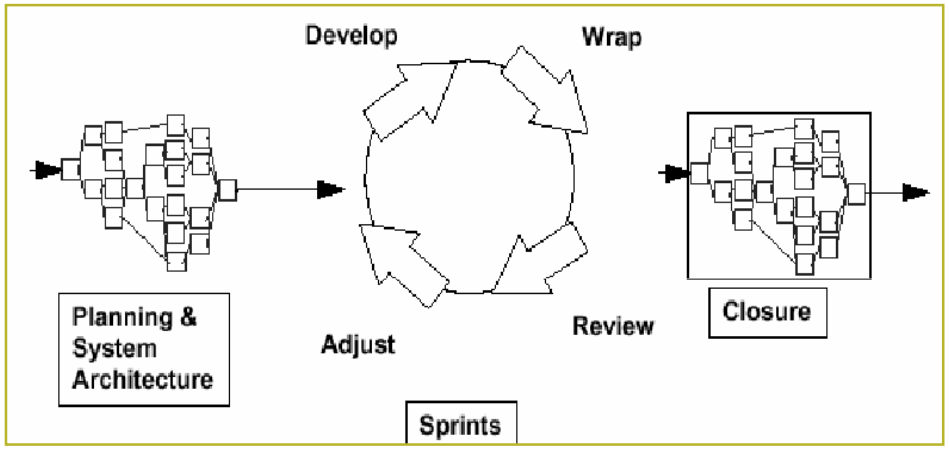
Hodnoty ve SCRUM:
důvěra
komunikace

- Ken Schwaber, Jeff Sutherland
  - » prvně cca 1993
  - » K.Schwaber, M. Beedle: *Agile Software Development with Scrum*, 2001
  - » K.Schwaber: *Agile Project Management with Scrum*. MS Press 2004

SCRUM

Proces

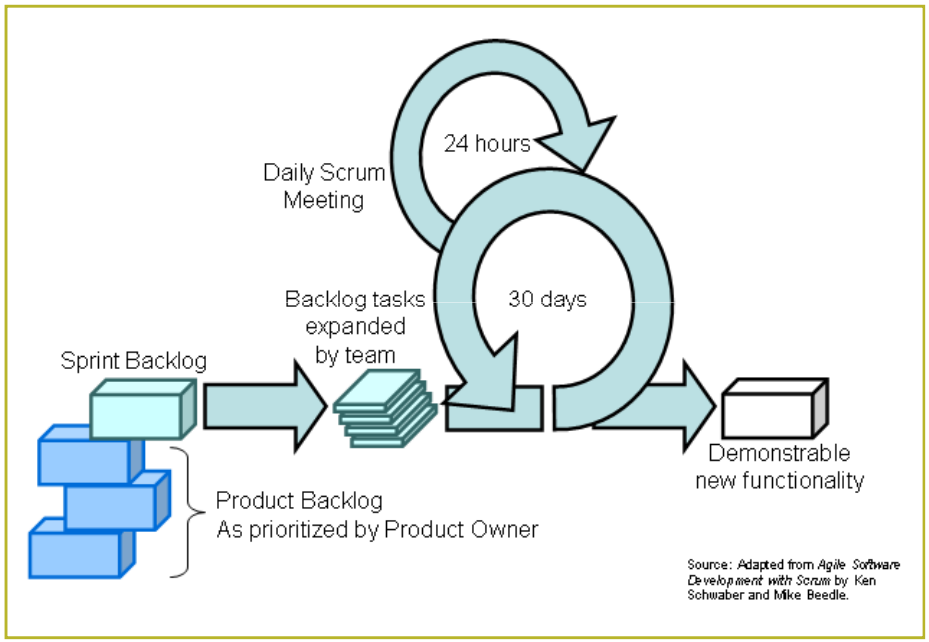
- Pre-game
  - plánování (vize)
  - jádro požadavků
  - architektonický prototyp
- Vývoj
  - » sprinty
  - přírůstková implementace
- Post-game
  - dodávka



SCRUM

Sprint

- Timebox (30dní) iterace
- Plánování
  - » product → sprint backlog
  - zákazník vybírá
  - tým odhaduje
- Práce
  - Daily scrum
  - libovolné technické postupy
  - žádné změny plánu
- Demonstrace přírůstku
  - » Sprint review
  - „demonstrace" nikoli „prezentace"
  - zákazník+tým hodnotí WRT plán sprintu

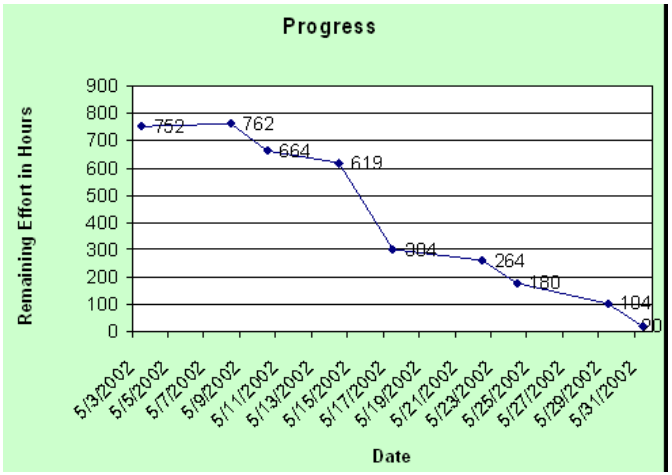


SCRUM

Backlog

- Backlog
  - features, úkoly, chyby, ...
    - » release backlog: 1-3 dny práce
    - » sprint backlog: 4-16 hod
  - tým identifikuje položky
  - zákazník stanovuje priority
- Graf postupu prací
  - „Sprint/release burndown“

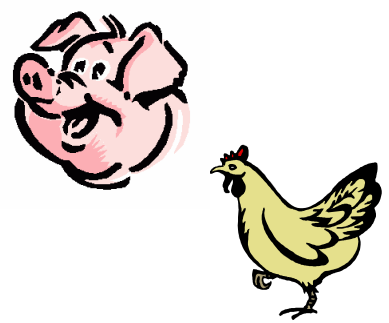
		Days Left in Sprint	15	13	10	8	
Who	Description		7/22/2002	7/24/2002	7/26/2002	7/31/2002	
		Total Estimated Hours:	554	458	362	270	0
-	User's Guide		-	-	-	-	-
SM	Start on Study Variable chapter first draft		16	16	16	16	
SM	Import chapter first draft		40	24	6	6	
SM	Export chapter first draft		24	24	24	6	
	Misc. Small Bugs						
JM	Fix connection leak		40				
JM	Delete queries		8	8			



SCRUM

Daily scrum

- Pravidelné setkání týmu
  - komunikace, zpětná vazba, empirický proces, průhlednost
- Pravidla
  - každý den, stejný čas a místo
  - 15minut vestoje
    - » Co jste dělali od posledního daily scrum?
    - » Co budete dělat do dalšího?
    - » Co stojí v cestě k cíli iterace?
    - » + Jsou nové položky do backlogu?
    - » + Naučili jsme se něco nového?
  - slepice a selata



SCRUM

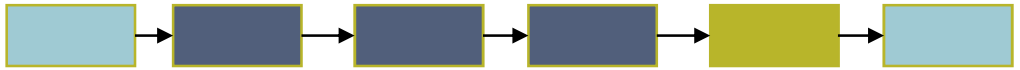
Tým

- „Empowered, self-organizing“
  - » function getTeamConstraints() { return null; }
- Členové týmu
  - všechny profese vč. QA
  - stabilní během sprintu
- Scrum Master
  - „manažer projektu“
    - » dbá na hodnoty a postupy
    - » firewall týmu, odstraňuje překážky
  - člen – 50% úvazek na implementaci
- Zákazník
  - „product owner“

SCRUM

Makro proces

- Větší projekt = více přírůstků / více týmů
- Jeden přírůstek



  - zahájení
    - » vize, hledání cesty
  - vývoj
  - stabilizace
    - » QA, odstranění chyb, příprava dodávky
- Meta-Scrum
  - Scrum master členem hierarchicky vyššího týmu

Extrémní programování

Software development fails to deliver. (…) We need to find a new way to develop software.

Kent Beck

XP

Kladný extremismus

Hodnoty v XP:
komunikace
jednoduchost
zpětná vazba
odvaha

- Protože kontroly nezaujatým čtenářem jsou dobrá věc, **budeme kontrolovat kód nepřetržitě.**
- Protože testování je dobrá věc, **všichni budou testovat neustále, dokonce i zákazník.**
- Protože návrh je dobrá věc, **uděláme z navrhování software každodenní chléb všech programátorů.**
- Protože jednoduchost je dobrá věc, **návrh systému bude vždycky ten nejjednodušší možný pro zachování aktuální funkčnosti.**
- Protože krátké iterace jsou dobrá věc, **budeme iterace mít opravdu krátké – vteřiny, minuty a hodiny, ne týdny, měsíce a roky.**

XP

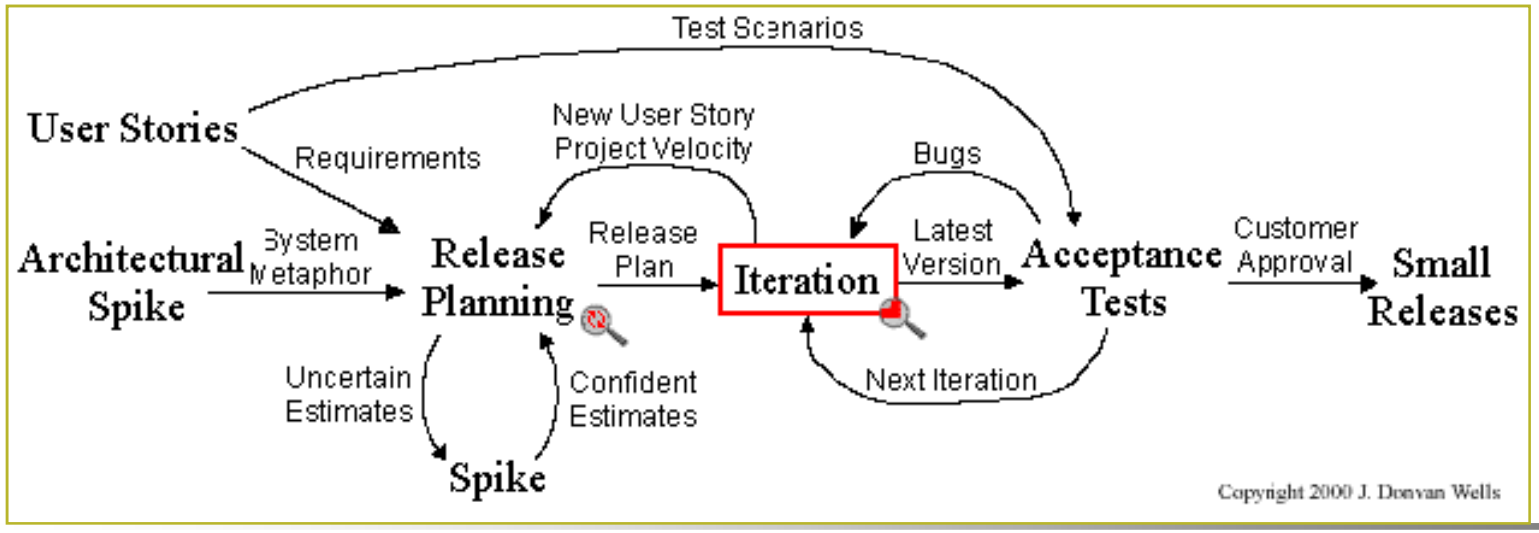
Podstatné rysy

- (Téměř) kompletní metodika
  - » programování jako hlavní aktivita ...
  - » ... ale synergie hodnot a technik klíčová
  - » dokumentace minimální, jen pokud je opravdu potřebná
  - » důraz na komunikaci
  - » horší škálování (C6-E20)
- Kent Beck, Ron Jeffries et al
  - » prvně C3 projekt u Chrysler 1996
  - » Kent Beck: *Extreme Programming Explained*. Addison Wesley 2000
  - » R.Jeffries, A.Andreson, C.Hendrickson: *Extreme Programming Installed*. Addison Wesley 2001

XP

Proces

- Průzkum
  - jádro požadavků
  - feasibility, odhad
- Plánování
  - release planning game
- Vývoj
  - iteration p.g.
  - přírůstková implementace
- Nasazení



XP

Role v týmu

- Vývojář
  - programátor
  - tester
- Zákazník
- Management
  - kouč
  - tracker
- Konzultant (externí)

XP

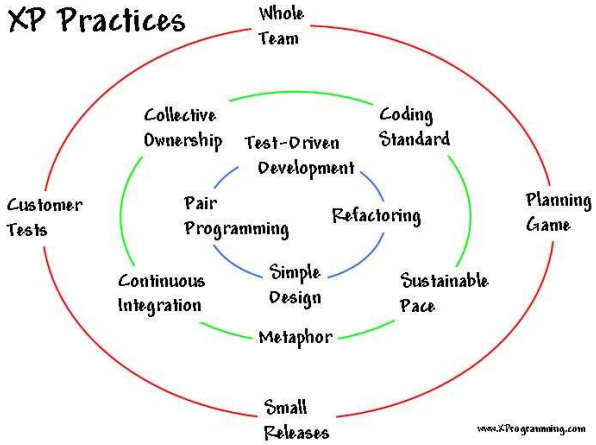
Nejdůležitější techniky

Vývoj

- metafora
  - » pro design
- refactoring
- test-first
- párové programování
- sdílený kód
  - » coding standard
- nepřetržitá integrace
- jednoduchost

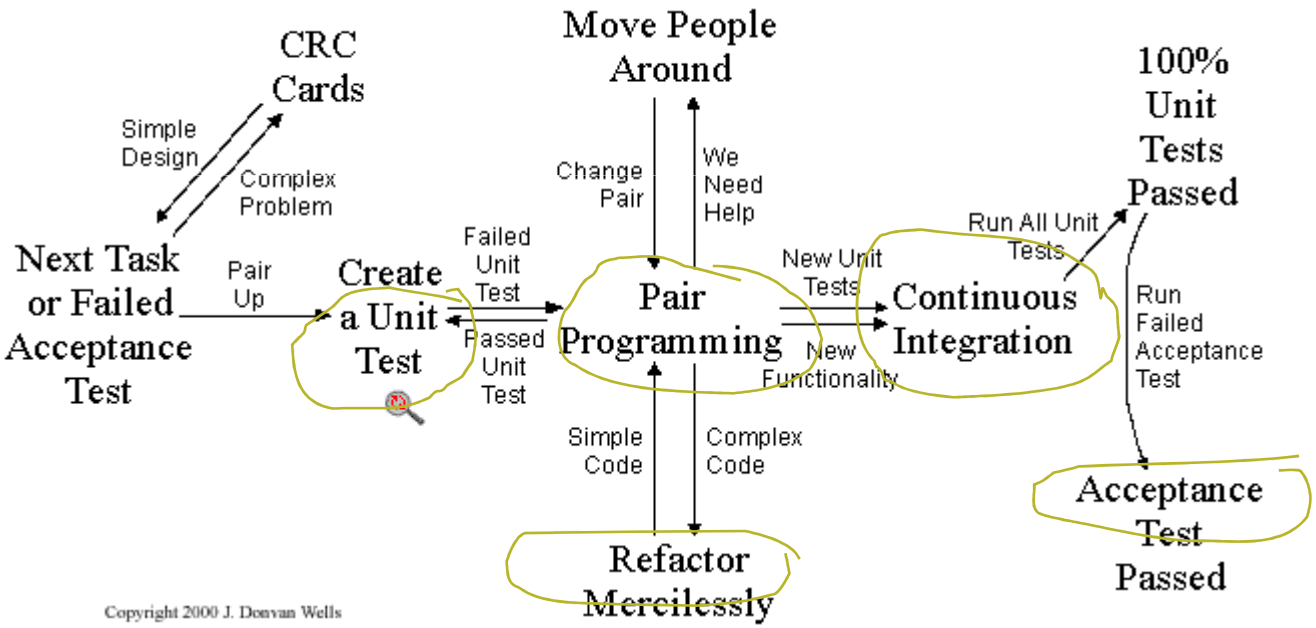
Management

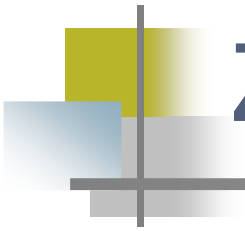
- udržitelné tempo
- kompletní tým
  - » on-site zákazník
- kartičky
  - » story cards (2-10 dní)
  - » tasks (1-2 dny)



XP

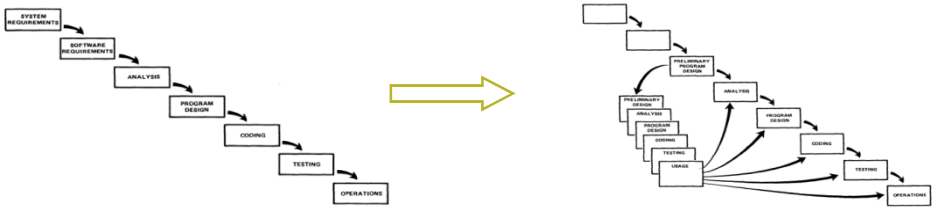
Synergie technik





Závěrečné poznámky

Původní vodopád, spirála



- „I believe in this concept, but *the implementation described above is risky* and invites failure. (…) The required design changes [due to errors found during testing] are likely to be so disruptive that the software requirements upon which the design is based and which provide the rationale for everything are violated. (…) If the program in question is being developed for the first time, *arrange matters so that the version finally delivered … is actually the second version* insofar as critical design/operations areas are concerned. (…) It is *important to involve the customer* … at earlier points before final delivery."
  - Winston Royce: *Managing the Development of Large Software Systems*. 1970 (článek s původním popisem vodopádového modelu; zvýraznění PB)

- „I feel that more and more what I and others are doing with agile comes back to Boehm's Spiral Model first postulated in 1986 and refined in 1988. Boehm described Spiral as *evolutionary* rather than incremental."
  - David J. Anderson, http://www.agilemanagement.net/Articles/Weblog/SpiralModelRevisited.html

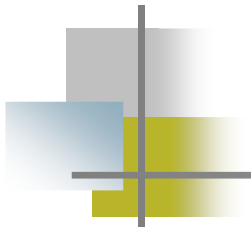
Projekt Mercury

- NASA 1958-1963
  - » člověk na oběžné dráze, bezpečný návrat
  - » poprvé řízeno počítačem
- Iterativní vývoj
  - » 1/2 denní iterace
  - » test-first vývoj, integrace

> We were doing incremental development as early as 1957 … I do remember Herb Jacobs (primarily, though we all participated) developing a large simulation for Motorola, where the technique used was, as far as I can tell, indistinguishable from XP. (…) All of us, as far as I can remember, thought waterfalling of a huge project was rather stupid, or at least ignorant of the realities…
>
> – Gerald Weinberg

Osobní poznámka

- Proč se mi to líbí
  - jednoduché, uchopitelné, představitelné
    - » na rozdíl od RUPu apod
  - vyzkoušený princip rychlého uspokojení
    - » programování řízené testy s JUnit
  - konečně odpověď na „problém s palačinkou“
    - » If you are made to wait, it is to serve you better, and to please you.
  - škáluje nahoru i dolů
    - » doložené na velkých (E200+) projektech
    - » ihned aplikovatelné na soukromých „jednomužných“



Přes to všechno…

- **Varování: Neopravovat, co není rozbité**

Zdroje

- Craig Larman: *Agile and Iterative Development. A Manager's Guide.* Pearson 2004
- Kent Beck. *Extreme Programming Explained.* Addison Wesley 2000
  - » existuje též český překlad
  - » http://interval.cz/clanek.asp?article=1792
- Václav Kadlec: *Agilní programování. Metodiky efektivního vývoje softwaru.* Computer Press 2004