ČÍM MOHOU PŘISPĚT NEJZÁMĚJŠÍ AGILNÍ METODIKY KE ZLEPŠENÍ VÝVOJOVÉHO PROCESU?

HOW WELL-KNOWN AGILE METHODOLOGIES CAN CONTRIBUTE TO A SOFTWARE DEVELOPMENT PROCESS?

Robert Pergl, Zdeněk Struska

Abstrakt:
Článek přináší přehled základních principů a zaměření nejznámějších agilních metodik s cílem identifikovat jejich originální přínosy.

Klíčová slova:
metodiky řízení softwarových projektů, agilní metodiky, extrémní programování, SCRUM, Lean Development, Feature Driven Development, metodiky Crystal, Adaptive Software Development, Dynamic Software Development Method

Abstract:
The paper introduces key principles and contents of the most important agile methodologies with the aim to identify their contributions.

Key words:
Software Development Methodologies, Agile Methodology, Extreme Programming, SCRUM, Lean Development, Feature Driven Development, Crystal Methodologies, Adaptive Software Development, Dynamic Software Development Method

ÚVOD

Zhruba od poloviny 90. let jsme svědky pronikání výpočetní techniky a softwaru do všech oblastí lidské činnosti. Spolu s překotným vývojem společnosti a všech odvětví se mění i pohled na řízení softwarových projektů. Typické softwarové projekty z oblasti obchodu a služeb se dnes vyznačují těmito aspekty:

Aby systém mohl být konkurenční výhodou, je třeba **rychle ho dodat.**
V průběhu vývoje se mohou **změnit požadavky**.
Uživatelé **nemají na počátku přesnou představu** o svých potřebách.
Zákazníci žádají **jakostní řešení**.

Zhruba od 2. poloviny 90. let se na poli softwarového inženýrství začínají objevovat snahy o přizpůsobení vedení projektů novým požadavkům doby. Za posledních 10 let vznikla celá řada úprav stávajících metodik či metodik zcela nových, které se snaží adresovat zmíněné problémy. Nejvýraznější snaha je označována jako **agilní přístup** (AP) nebo **rodina agilních metodik**.

Agilní přístup [2] vznikl jako sjednocení snah skupiny významných osobností softwarového inženýrství: *Alistar Cockburn, Kent Beck, Ward Cunningham, Marin Fowler, Ken Schwaber, Jeff Sutherland* a další. V průběhu své mnohaleté praxe tito odborníci identifikovali v oboru SI řadu problémů a přemýšleli o možnostech jejich zlepšení. Začali nezávisle na sobě pracovat na návrhu nového přístupu k programování. Postupně zjistili, že ač se jejich snahy určitým způsobem liší, obsahují zároveň překvapivé množství společných principů. V únoru

2001 se společně sešli v americkém Utahu a cílem jejich setkání bylo podrobně analyzovat jednotlivé nové metodiky, na kterých pracují, a najít společné prvky a formulovat zastřešující teze.
Účastníci byli překvapeni, jak snadno našli společnou řeč a výsledkem setkání je tzv. **Manifest agilního vývoje softwaru** (The Agile Manifesto) [1], který se stal základem rodiny agilních metodik a přístupů. Tento manifest podpořilo jen do roku 2004 svým podpisem již přes 1400 zájemců z kruhů odborné veřejnosti po celém světě. Nejznámější zástupci rodiny agilních metodik jsou Extrémní programování, SCRUM, Lean Development, Feature Driven Development, Agile Unified Process, Adaptive Software Development, Dynamic Software Development Method. Z důvodu velmi omezeného rozsahu tohoto příspěvku zde není prostor pro podrobnější představení jednotlivých metodik, odkazujeme tedy zájemce na literaturu a zaměříme se pouze na cíl příspěvku – jmenovat společné přístupy a vypíchnout specifika jednotlivých metodik.

SPOLEČNÉ RYSY

Lze říci, že všechny agilní metodiky určitým způsobem naplňují výše zmíněné teze. Mají do značné míry společné jádro a většina jejich praktik si je velmi podobná. Autoři metodik se však často uchylují k nestandardní (místy až exotické) terminologii a není tedy na první pohled vidět, čím je která metodika opravdu originální či nově inspirující. U všech agilních metodik lze v nějaké formě nalézt následující teze, principy a praktiky:

- **Orientace na zákazníka.** Hlavní snahou je dodat zákazníkovi produkt, který opravdu chce a potřebuje a nikoliv se zákazníkem bojovat.
- **Důraz na komunikaci**. Kladen je důraz na neformální, otevřenou komunikaci všech účastníků projektu (týmu i zákazníka).
- **Jednoduchost, neformálnost.** Ve všech metodikách je patrná snaha o zjednodušení a zefektivnění procesu vývoje, odstranění zbytečných nákladů a činností.
- **Inkrementální vývoj s krátkými iteracemi**. Vývoj vždy probíhá po malých funkčních celcích a hotový systém (či jeho prototyp) je dodáván zákazníkovi co nejdříve.
- **Využívání moderních technologií**. I když to není obecnou podmínkou, agilní metodiky sázejí na moderní objektově-orientované programovací jazyky a možnosti moderních vývojových prostředí.

SPECIFIKA

Výše zmíněné společné rysy lze v nějaké podobě nalézt v každé agilní metodice – např. základem iterativního vývoje je v Extrémním programování tzv. *User Story*, ve SCRUMu *Sprint*, ve Feature Driven Development *Feature*, v Dynamic Software Development Method *Timebox*, atp. Při podrobnějším studiu však lze pro každou metodiku nalézt cosi, čím vybočuje a co ji může činit atraktivní.

Extrémní programování (XP)

Metodika Extrémní programování [3] je poměrně propracovaná metodika. Její jádro tvoří 12 praktik, které se navzájem doplňují a podporují [4]. Její výhodou je její popularita, díky níž existuje dostatek literatury (i v českém jazyce). Díky většímu počtu uživatelů lze (na internetu) nalézt i obsáhlá diskusní fóra, postřehy z praxe a lze tak obdržet radu i konzultaci.
Nevýhodou metodiky je její velká striktnost, co se týká pravidel. Metodika by se měla implementovat celá, což často nebývá možné. Od přijetí metodiky také může odrazovat její „extrémní“ terminologie, která může v managementu (a hůře v zákaznících) evokovat představu divokého, nekontrolovatelného procesu (či chaosu) s nejasnými zárukami.
Metodika je v praxi úspěšně používána (v zahraničí i u nás), což svědčí o tom, že přednosti proklamované jejími příznivci se zakládají na skutečnosti.

SCRUM Development Process

SCRUM [5] je oproti XP konzervativnější v ohledu vlastnictví kódu – místo kolektivního vlastnictví je definována zodpovědnost za každý objekt či množinu objektů. Výhodou je zodpovědnost a možnost specializace, na druhou stranu, pokud vlastník opustí tým, může nastat problém s předáním objektu kompetentnímu následovníkovi.

Výhodou metodiky SCRUM je především větší orientace na *řízení týmu* a na zásahy vedení organizace. Je tedy vhodný pro týmy, které potřebují pevnější vedení než v XP.

Další velkou výhodou je důraz na řízení rizik v průběhu celého vývoje.

Lean Development

Metodika Lean Development [6] je zajímavá svým původem – prvotně vznikla po válce v Japonsku pro potřeby zefektivnění výroby automobilů. I přes odlišnost disciplíny softwarového inženýrství od ostatních inženýrských oborů lze nalézt mnoho analogií a inspirace. Ve svém důsledku je tato metodika naprosto koherentní s principy agilního vývoje.

Lean Development je zaměřena strategičtěji než ostatní agilní metodiky. Vyznává hodně principů společných s XP (vývoj řízený testováním, refaktoring, zpětná vazba, krátké iterace, atd.), nicméně je v některých ohledech opatrnější (např. svěřování pravomocí pracovníkům).

Hlavním přínosem metodiky je zaměření na pojmy *hodnota* a *plýtvání*, které jsou cestou k zefektivnění procesu a odstranění zbytečných činností a artefaktů. Velkým přínosem metodiky je též explicitní diskuse otázky efektivních *subdodávek a používání komponent*.

Ačkoliv metodika na první pohled ve srovnání např. s XP působí strohým, chladným a odlidštěným dojmem, zdůrazňuje důležitost příjemného motivujícího prostředí, ve kterém se všichni budou cítit dobře a odvádět maximální výkony.

Feature Driven Development

Feature Driven Development [7,8] je mezi agilními přístupy považována za „konzervativnější“ metodiku a má blíže ke klasickému přístupu než ostatní metodiky. Přesněji definuje své pojmy a proces a klade důraz na modelování.

Základním stavebním kamenem metodiky Feature Driven Development jsou *features*, tedy vlastnosti systému. Oproti např. metodice SCRUM a jejím pojmu s podobným významem „backlog” je feature v FDD přesně specifikována. Díky tomu je usnadněno provádění průběžných kontrol a kvantifikace vývoje. V porovnání s případy užití jsou velikosti jednotlivých vlastností na relativně stejné úrovni a nejsou tedy tak velké rozdíly v délce implementace. Metodika však počítá s menším prostorem pro změny během vývoje (v řádu procent).

FDD se klasickým metodikám přibližuje i v tom, že ve fázi plánování stanovuje pevné datum ukončení vývoje. Nepočítá tedy s flexibilním harmonogramem jako např. SCRUM.

FDD zavádí vlastnictví tříd a přiřazuje tak konkrétní zodpovědnost za určitou třídu konkrétnímu programátorovi (oproti společnému vlastnictví tříd v XP).

FDD pracuje s dynamickým vytvářením týmů, díky čemuž je možná škálovatelnost projektů a zvládá tedy větší projekty než ostatní agilní metodiky.

Metodiky Crystal

Jedná se o rodinu metodik, kde každá metodika je určena pro projekt určité důležitosti a rozsahu. Jedntolivé metodiky jsou potom navíc zaměřené na maximalizacích různých parametrů (produktivita, sledovatelnost, ...). Rodina metodik Crystal se tedy vyznačuje především svojí *konfigurovatelností*.

Metodika neobsahuje explicitní řízení rizik, ale nahrazuje je organizováním pravidelných schůzek, jejichž náplní je právě diskuse o případných problémech v projektu a jejich řešení.

Metodika Crystal je ze všech agilních metodik nejlépe *škálovatelná*: počítá s rozsahem projektů od několika lidí až do několika set. Z tohoto důvodu jsou některé úkony pojaty více formálněji.

Metodika Crystal je oproti jiným agilním metodikám více podrobná a konkrétní ve svých doporučeních. Je tak vhodná pro méně zkušené vedoucí projektů, kteří ještě nemají vyvinut dostatečný cit pro nastavení procesu, stanovení pravidel a dokumentů.

Adaptive Software Development

Metodika ASD [9] je nejdynamičtější metodikou z rodiny agilních metodik, přesto však zachovává procesní přístup. Jejím základním motorem je *změna* a přizpůsobení změně.

Metodika postupuje stejně jako ostatní agilní přístupy iterativně, nicméně na koncích jednotlivých iterací si neklade za cíl dodávat fungující produkt (jako např. Extrémní programování), účelem je pouze dát zákazníkovi dostatek podkladů k ověření postupu, maximálně betaverze produktu a prototypy uživatelského rozhraní. Hotový a odladěný produkt dodává až po skončení vývoje. Metodika je tedy vhodná pro zadání, které svoji podstatou není možné fragmetovat a dodávat postupně (např. bezpečnostní a řídicí systémy a systémy založené na workflow).

Metodika nedefinuje konkrétní postupy, pouze se zaměřuje na samotný adaptibilní učící proces. Konkrétní obsah procesu je třeba naplnit z jiných metodik, agilních i klasických. Je např. možné ve fázi spolupráce využívat párové programování z Extrémního programování a Scrum Meetings z metodiky SCRUM.

Metodika ASD má za sebou léta praktického používání. Oproti většině ostatních agilních metodik byla úspěšně použita i u rozsáhlých projektů.

Dynamic Software Development Method

Za vývojem metodiky DSDM stojí celé konsorcium šestnácti organizací. Metodika je zajímavá tím, že autoři k ní dodávají i *podpůrné vývojové prostředí (framework)*.

Metodika DSDM je značně propracovaná a je intenzivně vyvíjena již bezmála 20 let. Během vývoje do ní byly zapracovány moderní trendy a zkušenosti z vývoje. Metodika obsahuje řadu propracovaných technik spolu s doporučeními o jejich použitelnosti a zacílení.

Metodika se vyznačuje propracovaným vývojovým cyklem, který probíhá iterativně jak na úrovni hlavních fází, tak i uvnitř fází. Mezi hlavními fázemi je možné se i vracet, takže práce jsou vždy směrovány do místa, které je v dané chvíli klíčové.

Nevýhodou je, že metodika není zcela „zdarma“ – je třeba zaplatit členský příspěvek konsorciu. V ceně příspěvku je i podpůrný vývojový framework. Výše členských poplatků se liší podle charakteru členství (akademický, vláda, firemní, ap.) a pohybuje se od několika desítek liber do tisíců liber.

ZÁVĚR

Okolo agilních metodik a souvisejících otázek se vede v posledních letech řada diskusí, existuje poměrně mnoho literatury a tyto metodiky se postupně dostávají do povědomí odborné veřejnosti jako vhodná alternativa, která “může dosti pomoci” zlepšit vývoj softwaru. Cílem tohoto článku bylo přispět k této diskusi přehledným shrnutím toho, co je pro všechny metodiky společné a čím jsou jednotlivé metodiky “zajímavé”. V agilních metodikách lze nalézt mnoho inspirace i pro projekty vedené klasicky.

LITERATURA

1. Beck K. at al.: *Manifest agilního vývoje*, http://agilealliance.org
2. Buchalcevová A.: *Agilní metodiky*, sborník konference OBJEKTY 2002, ČZU Praha, ISBN 80-213-0947-4
3. Beck K.: *Extrémní programování* (český překlad) Grada 2002, ISBN 80-247-0300-9
4. Pergl R.: *Analýza vnitřních vazeb principů metody extrémního programování*, In: Sborník příspěvků z doktorandského semináře, ČZU Praha, 2005, ISBN 80-213-1314-5
5. Schwaber K., Beedle M.: *Agile Software Development with SCRUM*, Prentice Hall 2001, ISBN 0130676349

6. Poppendieck M., Poppendieck T.: *Lean Software Development: An Agile Toolkit for Software Development Managers*, Addison-Wesley 2003, ISBN 0321150783
7. Buchalcevová, A.: *Metodika feature-driven development neopouští modelování a procesy, a přesto přináší výhody agilního vývoje*, sborník konference Tvorba softwaru 2005, Ostrava
8. Felsing J. M., Palmer S. R.: *A Practical Guide to Feature-Driven Development,* Prentice Hall 2002, ISBN 0130676152
9. Highsmith J.: *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Publishing Company, Inc. 1999, ISBN 0932633404

KONTAKTNÍ INFORMACE

Ing. Robert Pergl, Ing. Zdeněk Struska
Katedra informačního inženýrství, Provozně ekonomická fakulta
Česká zemědělská univerzita. Kamýcká 129,
165 21 Praha 6. Czech Republic
tel.: +420-2-2438-3244
email: {pergl, struska}@pef.czu.cz