Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky

Možnosti využití agilních metodik při provozu a údržbě SW

Předmět: 4IT421 Zlepšování procesů budování IS
Vyučující: doc. Ing. Alena Buchalcevová, Ph.D.
Semestr: ZS 2011/2012

Vypracoval: Bc. Vlastimil Kouřimský
Datum: 7.12.2011

Obsah

Agilita v kontextu dneška

Dnešní svět informačních a komunikačních technologií se rapidně mění. A tento trend změn probíhá již nějakou dobu, a to nemluvím jen Moorovu zákonu. Dalším faktem poslední doby, alespoň podle akademické sféry je skutečnost, že infomační a komunikační technologie netvoří konkurenční výhodu. Od společnosti je zkrátka vyžadováno, aby její IT bylo maximálně funkční, neboť i malá společnost dnes disponuje CRM a dalšími, ideálně integrovanými, komponenty systému, které ještě před nějakou dobou byly doménou pouze velkých a nadnárodních korporací. S trendem bujarého narůstání objemu a komplexnosti IT vznikala potřeba řízení a poskytnutí nástrojů pro řízení, aby manažeři mohli takto velký kolos zvládnout a aby zaměstnanci měli k dispozici odpovídající vzdělání a nástroje k výkonu své práce.

Tato honba za inovacemi a snahou standardizovat přinesla za několik desetiletí standardizaci do vývoje software a systémů v podobě metodik a také použití frameworků a různých best practice (nejlepších praktik) do údržby a správy jak programového vybavení, tak i celých informačních systémů. Zmíněný posun byl tažený mimo jiné nutností řídit daný proces a mít přehled nad detailním fungováním. Pokud má být systém konzistentní a má být garantována očekávaná spolehlivost, je nutné zajistit určitou míru standardizace.

Výše zmíněný posun můžeme považovat za procesně a technologicky orientovaný, tedy takový, kde člověk jako jedinec hraje roli pouze vykonavatele určité aktivity, a to naprosto standardním, předem definovaným způsobem, který odpovídá normě, nebo předpisu.

V posledních letech oproti tomuto přístupu, především v oblasti vývoje software, je běžnou praktikou a cílem takzvaně odlehčovat těžké metodiky. Toto konstatování v podstatě znamená zbavit se nevýhod a přizpůsobit se dynamickému vývoji a posunout jej blíže k lidem, kteří poté přinášejí díky informačním technologiím přidanou hodnotu společnosti. Zmíníme-li negativa rigorózních metodik, bude to především zaměření na procesy a snaha zabránit změnám v průběhu procesu vytváření. Znamená to tedy investovat spoustu času a prostředků do analýzy stavu v dané chvíli, což se v dnešních podmínkách ukazuje jako neefektivní. To může být způsobeno na straně jedné špatným definováním zadání, nebo nesprávným pochopením problému na straně druhé.

V neposlední řadě musíme vzít v potaz měnící se požadavky byznysu, které se mohou prakticky měnit z týdne na týden, ze dne na den, a lidé na to musí být schopni reagovat, jinak společnost nemusí dnešní dravé prostředí přežít.

Vývoj software je dobrým příkladem, kde se těžiště přesouvá k odlehčování metodik. Bavíme se o přechodu k vyloženě lehčím metodikám, jako je například Scrum, nebo OpenUP. Dle mého je ale lepším zrcadlem snaha velkých společností, jakou je například IBM, o odlehčení, nebo postupné upouštění od těžkých metodik a přesun k agilitě. Agilitu můžeme v kontextu chápat jako míru schopnosti přizpůsobit se změně a reagovat na ni v časovém a kvalitativním měřítku.

Vymezení oblasti práce

Na následujících stránkách se budu věnovat posunutí agilních přístupů a metod i do provozu a údržby software. Tento posun jednak spadá do dnes požadovaného „out-of-the-box-thinking“, tedy nutnosti myslet za hranice běžných postupů a zažitých praktik. Za druhé je poměr mezi výdaji za údržbu a náklady na pořízení nového software výrazně ve prospěch údržby, respektive v její neprospěch, neboť výdaje na údržbu a provoz násobně převyšují výdaje za inovace a pořizování nového software. A tento trend je rostoucí. [1]

Aplikováním agilních praktik na vývoj software se podařilo zkrátit cyklus dodávek software a zvýšit úspěšnost projektů, při zachování, nebo snížení ceny. Dle známého přísloví „čas jsou peníze“ lze tedy předpokládat, že se tento poměr za využití agilních přístupů podaří redukovat v neprospěch údržby.

Možnostem využití agilních metodik při provozu a údržbě software se věnuji v tomto předmětu spolu s Lukášem Parmou. Abychom mohli obsáhnout danou problematiku blíže a zároveň neduplikovali naše práce, rozdělili jsme si naše téma následovně: Lukáš se ve své práci zaobírá blíže teorií a principy této problematiky, já se naopak na následujících stránkách budu zabývat praktickým použitím a měkkými aspekty agilních metodik.

Jako primární literaturu jsme zvolili knihu Provozujte IT jinak od autorů Jaroslava Procházky a Cyrila Klimeše. Tato publikace vyšla v polovině listopadu tohoto roku. Vzhledem k datu vydání a množství inspirace, které obsahuje, si

myslíme, že odpovídajícím způsobem reflektuje dnešní směr posunu požadavků pro většinu oblastí fungování, a to nejen informačních technologií ve společnosti.

Opomenutý lidský faktor

Ne náhodou jsem se v úvodu zabýval kontextem vývoje přístupu jak k vývoji, tak k údržbě software. O to více rezonuje skutečnost, že klasický přístup nebere ohled na lidský faktor. A to, co agilní přístup odděluje od klasického, je akcent na lidskou stránku týmu a na potřebu učit se s ní pracovat.

V literatuře najdeme bloky věnované vedení, motivaci, komunikaci a týmu, kde se všechny tyto vlastnosti snoubí. Prakticky je potlačen způsob, jakým se daná věc udělá, a je vyzvednuta skutečnost, aby daná věc byla komunikovaná a odpovídala zadání. Trend upřednostňovat lidský faktor se vyvíjí spolu s poznatky z oblasti psychologie, sociologie a neurověd, se kterými vědci přicházejí. Cílem je dosáhnout maximální ergonomie dané práce, tedy práci ulehčit, udělat ji příjemnou a využít nejmodernější dostupné poznatky.

Rigorózní metodiky pocházejí „z dob průmyslové výroby“, kde se člověk choval pouze jako stroj s přesně definovaným zařazením. Dnes je důležitá otevřenost, vlastní přínos a kreativita, tato cesta vede k agilitě s respektem k lidskému faktoru.

Měkké aspekty agility

Pokud jsme v základu zapojili lidský faktor, měli bychom se na tomto místě zmínit více o měkkých aspektech a o osobnostech jednotlivých členů týmu, neboť tento fakt kriticky ovlivňuje právě agilní přístupy, neboť nejsou tak detailně řešeny a existuje v nich vysoká míra odpovědnosti jednotlivých členů k projektu a k jeho výsledku. Primární literatura se tématu měkkých aspektů věnuje velice podrobně a do detailu. Pro potřeby této práce si dovolím tyto oblasti zestručnit.

Techniky a ICT odborníky jsou většinou absolventi vysokých škol se solidním technickým zaměřením, jsou mistry v takzvaných hard-skills, tedy naučených dovednostech. Co jim ale často chybí, jsou soft-skills, tedy měkké dovednosti. Neboť právě předměty jako jsou psychologie, sociologie, nebo dokonce komunikace se povětšinou na jejich oborech nevyučují. Proto ani následující podkapitoly týkající se měkkých aspektů nejsou součástí žádných

norem ani přístupů z „průmyslové doby“. Avšak v dnešních podmínkách jsou právě pro ICT klíčové, protože informační systémy jako takové nemají už tak velkou přidanou hodnotu oproti konkurenci a jejich informačním systémům. To, na čem záleží, jsou lidé, kteří je navrhují, využívají, spravují a provozují.

Každý týmový pracovník je osobnost

Tento bod se na první pohled může zdát naprosto banální. V kontextu agilních přístupů, kde se pracuje v malých týmech a v menším, ale v dynamičtějším prostředí, je osobnost člověka klíčová. Pokud člověk nefunguje v rámci velkého rigorózního týmu, neboť se v něm „schová“, může existovat vcelku spokojeně, nebude však přínosem pro tým. V menším týmu je tato možnost téměř eliminována.

Samotným uvědoměním to však nekončí, následují ještě další body, které je třeba vzít v potaz.

Jednání v kontextu

Stejně jako je každý člen týmu osobností a individualitou, má s sebou rovněž i rozdílnou výbavu zkušeností: pracovních, životních a jiných. V neposlední řadě vyznává i jiné hodnoty, které mohou mít u různých lidí odlišný stupeň důležitosti.

Motivace člena týmu

Motivace je prakticky téma samo o sobě. Mezi vnější motivaci členů týmu můžeme zařadit především finanční ohodnocení výší platu. Další součástí odměny a motivace je nefinanční odměna, což může být i postavení jedince v rámci týmu. Dnešní posun v Maslowově pyramidě potřeb je až kritické. Seberealizace a potřeba uznání jsou jedny z nejsilnějších motivačních sil z těch vnitřních, které oproti těm vnějším rozhodují.

Vedení týmu

Vedení a formování týmu je v podstatě jiné, než u klasických týmů, a to je dáno především jeho menší velikostí a přesvědčením, že samo-organizující se týmy přinášejí nejvyšší přidanou hodnotu.[3]

Komunikační problémy

Nejlepší systém komunikace v týmu samotném je osobní kontakt. Pokud jednotlivce zavalíme spoustou dokumentace, předpisů a komunikačních pravidel, komunikace v rámci týmu se stane těžkopádnou. A takováto neefektivita se musí zákonitě projevit i v komunikaci směrem k zákazníkům, nebo partnerům. Další problémy vznikají nezkušeností a nedostatečným tréninkem komunikace. Sice všichni mluvíme stejným jazykem, v prostředí IT a v byznysu však všichni často nemluví stejnou řečí.

Očekávání

Mylná očekávání, nebo představy, následované špatnou komunikací jsou častým kamenem úrazu neúspěšnosti projektů. Rozsáhlé dokumentace v případě rigorózních metodik jsou dobrým argumentem v ruce, při negociaci výsledků a úspěšnosti projektu, reálné použití software a úspěch může však ležet naprosto jinde. Jasná očekávání jsou důležitá i v rámci práce v týmu samotném, nejen ve vztahu je ostatním pracovníkům a zákazníkům mimo projekt.

Zpětná vazba

Zakončení zpětnou vazbou není náhodné. Zpětná vazba je zvláště v agilních metodikách a přístupech klíčem k neustálému zlepšování a k reflexi stavu, tak jak se danou problematikou zabývá a zavazuje se k ní agilní manifest. Zpětnou vazbu je třeba se v našich končinách naučit. V podnikové praxi se setkáváme s jejím nedostatkem, v horším případě s naprostým nepochopením jejího účelu, smyslu a pravidel. Není zvykem, aby se v průběhu práce nebo po jejím dokončení někdo vyjadřoval, diskutoval, nebo dokonce se snažil zlepšit daný postup, nebo celé fungování. Naplnit poslání manifestu agility, a nejen jeho, není možné bez neustálé touhy po zlepšování jak sebe, tak i okolí.

Testování členů týmu

Všechny zmíněné oblasti spolu velice úzce souvisí. Již nějakou dobu existují testy týmových rolí, nebo typologie vůdců a různé další psychosociální testy, které dokáží určit, jak se daný člověk bude v týmu projevovat a co od něj můžeme čekat. Je zásadní, aby členové týmu disponovali určitými vlastnostmi a chápali jejich význam a smysl právě na agilním poli. Zavádět agilitu a snažit se ji

aplikovat v organizaci ať už ve výrobě software, nebo poté v jeho správě a provozu se totiž často míjí účinkem.

Agilní praktiky uplatnitelné v údržbě

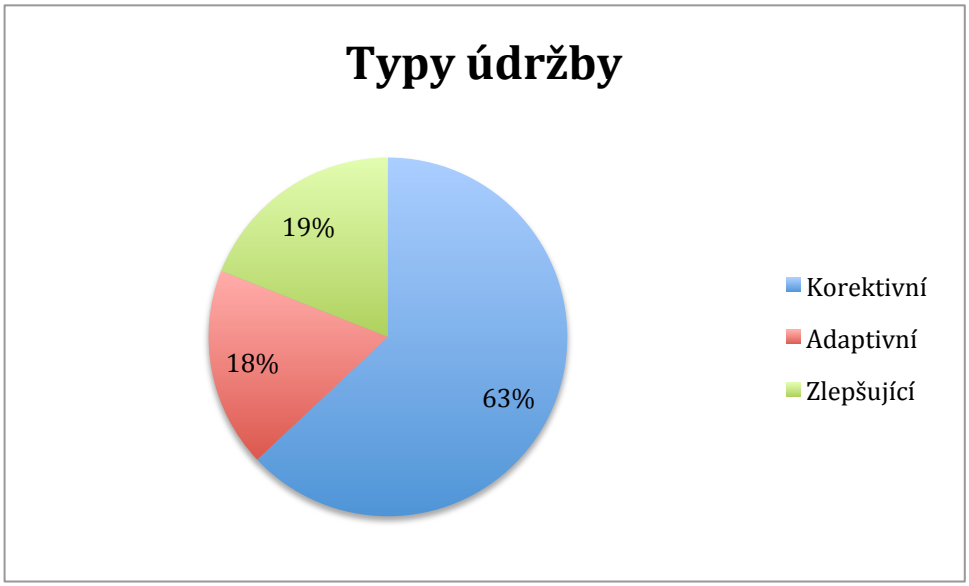
Po představení předpokladů a uvedení do kontextu rozdílnosti rigorózního a agilního přístupu můžeme přejít ke konkrétním realizacím a problémům, které v rámci údržby a provozu musí administrátoři a konzultanti řešit.

Zásahy vyskytující se v údržbě software

Nejprve si rozdělme údržbu na 3 typy:

- Korektivní údržba – řeší reaktivně korekci chyb, které se vyskytly v provozu.
- Adaptivní údržba – řeší požadavky na změnu funkcionality, které se vyskytnou v rámci provozu na základě změny podmínek v podniku.
- Zlepšující údržba – řeší zlepšování výkonnosti produktu. [1]

Následně se můžeme podívat na procentuální graf zastoupení daných chyb v systému.



Zdroj: [4]

Z obrázku vyplývá, že jasnou převahou disponují korektivní chyby, tedy chyby, které se vyskytují v provozu a měly by být opraveny. Tyto chyby jsou podobného rázu, jako chyby, které vznikají i při vývoji software. Přistupovat

k těmto chybám a pracovat s nimi může dané oddělení v duchu agilních přístupů. Stejně tak ostatní chyby. Proces může být velice podobný procesům vytváření software s tím rozdílem, že údržba a provoz probíhají povětšinou u zákazníka. Zároveň zde může mít aplikace agilních přístupů akcelerující účinek na zvýšení spokojenosti koncových uživatelů a snížení čekacích dob na opravu chyb, především díky iterativnímu vývoji a posílené komunikaci.[5]

Využitelné agilní přístupy

Následující výběrový výčet budu věnovat technikám, které jsou založené na agilitě, ale nebudu tyto techniky rozdělovat dle metodik. Bližší teoretický rámec k metodikám uvádí ve své práci Lukáš. Tento výčet má sloužit jako komentovaný pohled na techniky, které je možné využít v rámci projektu a implementovat z hlediska jejich praktického uplatnění u zákaznického projektu, realizovaného k údržbě a provozu software, vycházím z [6].

Přítomnost zákazníka

Jednou z klíčových praktik agilního přístupu je mít zákazníka/klienta k dispozici. V našem případě se jedná o přítomnost zákazníka při údržbě a provozu. Důvod je jednoduchý, pokud je vývojový a servisní tým v úzkém kontaktu s konkrétním zástupcem zákazníka, má tak větší šanci lépe reagovat na vzniklé podněty a situace a může dodávat tak své služby v optimální kvalitě. Pokud přijde na údržbu požadavek k opravení chyby, nebo je tato nahlášena, je možné daleko lépe doplnit další informace, a to jen díky možnosti obrátit se na přiděleného zástupce, který je též ze strany zákazníka částečně zodpovědný.

Krátké iterace

Rizikem dlouhého vývoje je protahování opravy aktualizací, které jsou pro zákazníka důležité. S častými a bezproblémovými úpravami roste zákazníkova spokojenost. Zkrácení času iterací působí i pozitivně na objem práce a počet chyb, které mohou vzniknout při uvolnění velkých kusů záplat daného software.

Testování

Rigorózní přístupy testování se odsouvají na konečné fáze projektu, kde je zpravidla minimum času zabývat se potřebným testováním funkcionality. Trend u agilních přístupů je testovat co nejvíce. V tomto případě mají vývojáři starající

se o údržbu a provoz stejnou zodpovědnost testovat jako vývojáři, kteří připravovali původní software pro zákazníka. Výsledkem je minimum nechtěných chyb při změně funkcionality, nebo při nechtěném zásahu do kódu, kde vývojář změní nevědomky nějakou návaznost na jinou funkcionalitu.

Párové programování

Párové programování dle uváděného projektu [6] nebylo uplatněno. Dá se předpokládat, že u malého týmu programátorů, kteří mají na starosti údržbu, nebude nutné vyloženě trvat na uplatňování tohoto agilního postupu. Předpokládá se, že interakce mezi nimi bude kvůli dennímu chodu potřebná a žádoucí.

Denní stand-up meetingy

Potkávání a komunikování je jedna z klíčových věcí vůbec. Tato technika pochází ze Scrumu, kde se vývojáři sejdou každý den na půl hodiny a mají za úkol se navzájem informovat, co se událo, jaké mají problémy a co se bude dělat. Základní myšlenkou je synchronizovat tým, vyloučit dlouhé lhůty týmových reakcí a minimalizovat možná rizika projektu tím, že v rámci kompletního projektu a jeho návazností se odchytí případná vada, nebo nekonzistence v co nejranějším stádiu. Pro týmy, které se starají o údržbu, může být denní interval příliš častý, záleží poté hodně na aktuálních preferencích, umístění týmových kolegů geograficky a dalších okolnostech.

Optimalizace až na konec

Tato technika je podle zmíněné studie [6] pro tým údržby a provozu zbytečná. Už samotný vývoj daného software na základě agilních principů obsahuje tuto techniku. Přílišný zásah a změny v kódu v produkční fázi jsou dle zjištění kontraproduktivní. Tyto změny stojí obvykle hodně času a v našem případě by přinesly jen mizivé přínosy na úkor vysokých nákladů. Zvláště v případě, prováděl-li by údržbu tým, který nebyl přítomen programování a vývoji daného software.[8]

Žádné přesčasy

Zde se opět vracíme k psychologické problematice práce, zapojení nových poznatků, výkonnosti a motivace. Přesčasy v delším horizontu, nebo nárazové

nepůsobí na motivaci a aktivitu týmu dobře. Na druhou stranu čas strávený na projektu je vykompenzován větší efektivitou. To platí v případě vývoje software, tedy ještě předtím, než je odevzdán zákazníkovi. Při plném nasazení systému závisí na nastavení SLA (dohod o kvalitě služby) smluv a v určitých případech se přesčasům není možné vyhnout. Proto bych tento agilní přístup neviděl jako relevantní pro oblast provozu a údržby.

Refaktorování

Za touto praktikou agilního přístupu se skrývá úprava vnitřku kódu při zachování stejné funkcionality. Dalo by se tedy říci, že refaktorovat má smysl pouze tehdy, je-li kód v nasazeném prostředí opravdu špatně napsaný. Tyto zásahy by stejně jako technika Optimalizace až na konec neměly spadat do kompetencí týmu. Větší smysl dává sbírat tyto postřehy (pokud nejsou kritického rázu) a předat je po jejich kumulaci dodavatelskému týmu, který by měl být schopen je zapracovat do své aplikace.[6][7]

Společné vlastnictví kódu

Kód vlastněný všemi členy malého týmu je smysluplný pro využití i v oblasti údržby a provozu. Můžeme předpokládat, že tým, který se bude starat o údržbu a provoz, bude vždy menší, než tým, který daný software připravoval, proto by tím spíše, pro lepší přehled a snížení reakční doby na vyvstalý problém, měli mít všichni vývojáři vlastnická práva na kód.

Závěr

Na závěr bychom se měli shodnout na tom, že klasický/rigorózní přístup (orientovaný na procesy) od agilního přístupu odděluje právě zájem především o lidi a respektování lidských oblastí fungování za použití a zakomponování moderních poznatků psychologie, sociologie a neurověd do ověřených postupů a uzpůsobení se jim. Dále z celého vyplývá, že velice důležitá je i dimenze lidská, a to obzvláště pro agilní přístupy, kde jsou pracovníci nuceni spolupracovat daleko intenzivněji a intenzivněji se setkávat, než je tomu u klasických přístupů, a to ať už se jedná jak o vývoj, tak o údržbu a provoz. Je to jistě nový přístup, který bude prorůstat stále více do fungování všude tam, kde jsou lidé nuceni společně komunikovat a spolupracovat. Ve světě informačních technologií bude toto setkávání nejintenzivnější, protože lidé zde jsou velice vzdělaní, zároveň mají přístup k posledním technologiím a toto je doplněné o další „lidské“ prvky.

Zajímavou aplikací agilních přístupů je i jejich uplatnění v provozu a v údržbě software. Na základě předložených faktů a studia literatury daleko více než aplikování konkrétní metodiky, dává smysl využít principy přesně podle toho, v jaké situaci se nachází servisní tým, který provozuje software. Navíc v návaznosti na postupné upouštění, nebo odlehčování přístupů k samotnému vývoji software se jako další krok jeví přechod i k jeho správě na základě agilních přístupů. Věřím, že se mi, na příkladech v části věnované konkrétním přístupům, podařilo několikrát nastínit, kde provoz a údržba přímo navazují na vývoj, proto by se i do budoucna mělo toto pouto mezi provozem a údržbou utužovat. Na konci obou aktivit se totiž nachází nejen byznys pro společnost, ale i spokojený zákazník, bez kterého nemůže v dnešním světě společnost přežít.

Zdroje

- [1] PROCHÁZKA, Jaroslav; KLIMEŠ, Cyril. *Provozujte IT jinak : Agilní a štíhlý provoz, podpora a údržba informačních systémů a IT služeb*. 1. Praha : Grada Publishing, 2011. 288 s. ISBN 978-80-247-4137-6.

- [2] Best Management Practice [online]. 2010 [cit. 2011-12-05]. *IT Service Management – ITIL®.* Dostupné z WWW: <http://www.best-management-practice.com/IT-Service-Management-ITIL/>.

- [3*] Manifesto for Agile Software Development* [online]. 2001 [cit. 2011-11-25]. Twelve Principles of Agile Software. Dostupné z WWW: <http://agilemanifesto.org/>.

- [4] JAIN, Naresh . *SlideShare* [online]. 2008 [cit. 2011-12-02]. Agile Maintenance. Dostupné z WWW: <http://www.slideshare.net/nashjain/agile-maintenance>.

- [5] *Tools For Agile Blog* [online]. 14.3.2008 [cit. 2011-12-07]. Agile for maintenance projects. Dostupné z WWW: <http://toolsforagile.com/blog/archives/92>.

- [6] SHAW, Steve. Using Agile Practices in a Maintenance Environment. *Intelliware Development Inc*. [online]. 2007, 1, [cit. 2011-12-07]. Dostupný z WWW: <http://i-proving.ca/download/Articles+and+White+Papers/shaw_paper_agile2007.pdf>.

- [7] Refaktorování. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 21. 6. 2004, last modified on 13. 9. 2011 [cit. 2011-12-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Refaktorován%C3%AD>.

- [8] VASHISHTHA, ShriKant. *Xebia Blog* [online]. 14.8.2008 [cit. 2011-12-07]. Preparing for Agile Maintenance – Knowledge Management. Dostupné z WWW: <http://blog.xebia.com/2008/08/14/preparing-for-agile-maintenance-knowledge-management/>.