ŠKODA AUTO VYSOKÁ ŠKOLA, O.P.S.

Studijní program: B6208 Ekonomika a management
Studijní obor: 6208R088 Podniková ekonomika a management provozu

MODERNÍ METODIKY TEAMOVÉHO VÝVOJE SOFTWARE

Karel NOVOTNÝ

Vedoucí práce: Ing. Pavel Wicher, Ph.D.

Tento list vyjměte a nahraďte zadáním bakalářské práce

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury pod odborným vedením vedoucího práce.

Prohlašuji, že citace použitých pramenů je úplná a v práci jsem neporušil autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Mladé Boleslavi dne .....

Děkuji Ing. Pavlovi Wicherovi, Ph.D za odborné vedení bakalářské práce, poskytování rad a informačních podkladů.

Obsah

Seznam použitých zkratek a symbolů

IT	Informační technologie
SDLC	Systems development life cycle
VCS	Version control system

Úvod

Hlavním cílem této bakalářské práce je navrhnout optimalizaci řízení pracovních teamů v rámci flexibilního vývoje software. Návrh na optimalizaci bude zpracován na základě analýzy současného stavu ve vybrané společnosti.

Dílčím cílem je zpracování základního přehledu jednotlivých metodik v současnosti využívaných pro vývoj software a jejich zhodnocení z hlediska typu, rozsahu a časové náročnosti projektu. Vhodnost výběru metodiky bude porovnána pomocí metody Cynefin.

Hlavním předpokladem je zvolení vhodné agilní metodiky pro vývoj software pro udržení konkurenceschopnosti vybrané společnosti. Konkrétně jsem navrhl zvolit metodiku Scrum a Kaban jako nejvhodnější metodiky. V současnosti jde o nejprogresivnější nejlépe popsané metodiky používané při agilním vývoji software.

V úvodu bakalářské práce jsem definoval základní terminologii a principy využívané při vývoji software. V následujících kapitolách je uveden přehled v současnosti používaných tradičních i agilních metodik. V kapitolách číslo 4, 5 a 6 jsem provedl analýzu současného stavu ve vybrané společnosti a navrhl optimalizace vývoje software s využitím agilních metodik vývoje software. Poslední kapitola pojednává o přípravě aplikačního plánu.

1 Vývoj software

Je soubor činností vedoucí k vytvoření projektu (softwarového produktu), dále jen aplikace. Tento proces v sobě zahrnuje nejen činnosti přímo se podílející na vývoji a údržbě aplikace, které jsou programování, tvorba dokumentace, testování a opravy chyb, ale také implementaci metodik, které mají zajistit hladký průběh vývoje aplikace a dostatečnou kontrolu kvality nad vývojovým procesem (Best Price Computers, 2015).

1.1 Projekt

Definice projetu zní: „Časově omezené úsilí vynaložené na vytvoření unikátního produktu, služby nebo výstupu“ (Schwalbe, 2011).

Abychom však získali lepší představu o projektu, tak uvedu několik atributů, které definici projektu lépe vymezí a upřesní. Prvním z atributů je jedinečný účel, tedy definovaný cíl projektu. Časová ohraničenost projektu jednoznačně určuje jeho konec a začátek. Postupné rozpracování projektu znamená, že se postupně od obecné definice dostáváme ke konkrétním a specifickým detailům. K projektu patří zdroje, může se jednat o zdroje lidské, software, hardware nebo i majetek. Důležitou součástí projektu je jeho vlastník, který jednoznačně určuje cíle a priority. Nakonec musíme u projektů počítat i s nejistotou, protože většina projektů je jedinečných a tak je obtížné jasně definovat cíle a odhadnout časovou dotaci i náklady (Schwalbe, 2011).

Projektový trojimperativ hovoří o ohraničení projektu rozsahem, časem a náklady a výsledkem jejich průniku je úroveň kvality projektu. Rozsah určuje práci nutnou k vykonání, popis výstupu projektu a způsob ověření výstupů. Čas určuje trvání a harmonogram projektu, také podmínky splnění časového kritéria. Náklady určují cenu realizace projektu, rozpočet a způsob sledování nákladů (Schwalbe, 2011).

1.2 Kvalita

Na výslednou kvalitu projektu má vliv funkčnost, tj. míra shodnosti systémů se zadáním. Výstupy systému, tedy podoba dat, které systém poskytuje uživatelům. Efektivita, která je stanovena mírou do jaké systém plní požadavky zákazníka. Spolehlivost určuje schopnost systému chovat se předvídatelně. Udržovatelnost je míra jednoduchosti s jakou lze systém udržovat a dále rozvíjet (Schwalbe, 2011).

1.3 Komunikace

Komunikace je klíčovým aspektem úspěchu každého produktu, proto musíme u každého projektu zavést řízení komunikace, které zajistí generování, shromažďování, distribuci, dokumentaci a archivaci projektových informací (Schwalbe, 2011).

Identifikace zainteresovaných stran nám poskytne obrázek o odesilatelích a příjemcích zpráv souvisejících s projektem a umožní nám vhodně nastavit komunikační kanály. Plánování komunikace zajistí, kdo a jaké zprávy bude dostávat (Schwalbe, 2011).

1.4 Metodiky vývoje software

Někdy také metodiku nazýváme modelem a znamená to strukturované rozdělení vývoje aplikace do oddělených fází. Jednotlivé fáze se dále dělí na činnosti se zaměřením na možnost lepšího plánování, kontroly nad procesem a managementu vývoje aplikace. Většina metodik sdílí kombinaci následujících fází vývoje aplikace (Testování softwaru, 2015):

- Analýza problému,
- shromažďování požadavků pro navrhované řešení,
- návrh plánu vývoje aplikace,
- vývoj aplikace,
- testování aplikace,
- nasazení aplikace,
- údržba a opravy chyb aplikace.

Všechny stadia vývoje aplikace označujeme souhrnně jako životní cyklus vývoje systému (SDLC) a různé metodiky mohou jednotlivé fáze vývoje vykonávat v různém pořadí a v různé časové dotaci, i podrobnost dokumentace v jednotlivých fázích se může lišit v závislosti na použité metodice. Další neméně důležitou odlišností v jednotlivých metodikách může být, zdali se k jednotlivým fázím vývoje v rámci metodiky přistupuje sekvenčně, nebo v cyklech či iteracích (Schwalbe, 2011).

Sekvenční přístup se dá považovat za přístup konvenční a naopak za extrémnější (agilní) lze považovat přístupy využívající cykly a iterace. Jejich extrémnost spočívá zejména v potlačení prvotních konvenčních fází vývoje, jako je plánování a tvorba dokumentace na nutné minimum a v důrazu na čas strávený samotným vývojem a psaním automatizovaných testů aplikace s předpokladem, že může v průběhu vývoje nastat i významná změna v návrhu aplikace. Dobře napsané automatizované testy zajišťují odhalení chyb již při změně návrhu aplikace, nebo refaktoringu[1].

Výběr vhodné metodiky vždy závisí na typu řešeného problému. Konvenční přístupy jsou vhodné k aplikaci zejména v případech, kde je oblast řešeného problému velmi dobře známa a řešení může být efektivně naplánováno dopředu. Extrémní přístupy volíme v případech, kdy řešíme unikátní problém (i na úrovni vývojového týmu) anebo nelze design a plán vývoje aplikace dobře vytvořit či odhadnout dopředu.

Agilní metodiky použité při optimalizaci vývoje software budou z hlediska jejich obsáhlosti popsány v samostatných kapitolách.

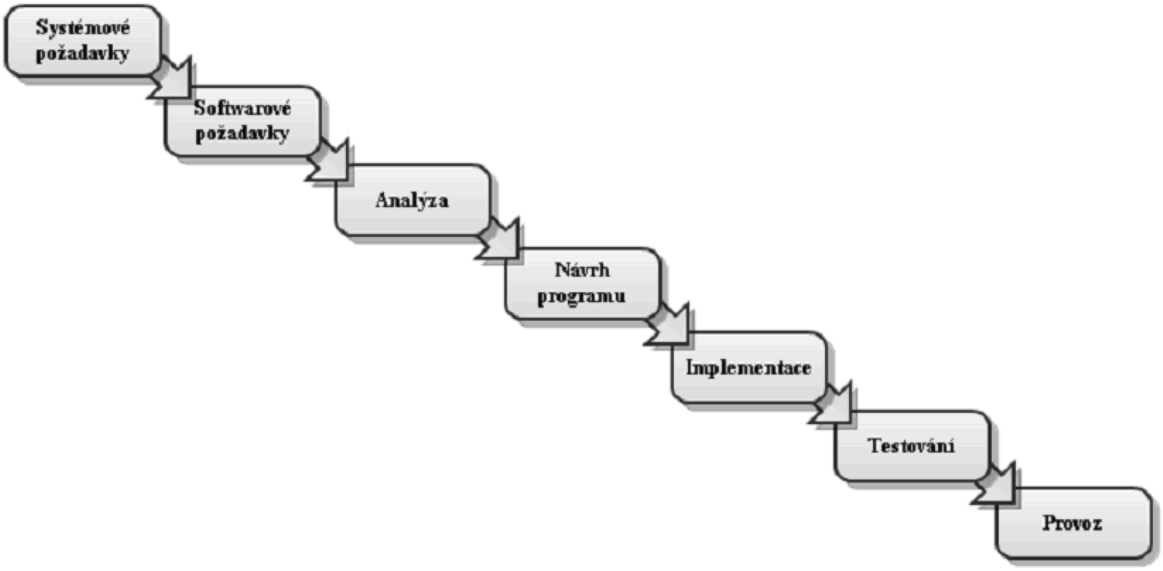
1.4.1 Vodopádový model

Je zástupcem tradičního přístupu k vývoji software. Předpokládá, že všechny body návrhu zůstanou po jejich stanovení neměnné. Jedná o přesně definované postupy a fáze vývoje, u kterých je velmi malý prostor pro přizpůsobení se. Typický průběh vývoje u vodopádového modelu vidíme na obrázku 1 (Testování softwaru, 2015).

Přechod z jedné fáze do druhé, je možný jen a pouze po dokončení fáze předchozí. Z tohoto důvodu není vhodné používat vodopádový model pro projekty, u kterých dochází často ke změně zadání i v průběhu jejich vývoje. Vodopádový model lze doporučit pro jednoduché nebo takové úkoly, u kterých jsme schopni přesně definovat a specifikovat zadání (Testování softwaru, 2015).

[1] Refaktorování je disciplinovaný proces provádění změn v softwarovém systému takovým způsobem, že nemají vliv na vnější chování kódu, ale vylepšují jeho vnitřní strukturu s minimálním rizikem vnášení chyb (Fowler, 2003).

Jeho obliba i přes zmíněné nevýhody je zřejmě zapříčiněna jeho jednoduchostí a přímočarostí při zavádění do vývojových procesů i bez znalostí projektového řízení v oblasti IT.



Zdroj: http://testovanisoftwaru.cz/wp-content/uploads/2011/07/vodopad1.png

Obr. 1 Vodopádový model

1.4.2 Agilní a štíhlé metodiky

S prudkým rozvojem webových technologií a potřebou rychlého vývoje přestaly tradiční metodiky k vývoji software dostačovat. Selhávali v oblasti flexibility a kvality, jelikož se nedokázali dostatečně rychle přizpůsobovat neustále měnícím se požadavkům od zákazníků. Z jejich definice je toto bráno jako chyba návrhu, ale modelování komplexních systémů je velmi náročná a nákladná činnost. Proto byly vyvinuty nové agilní metodiky, které odstraňují tyto nedostatky a soustředí se na rychlost vývoje a udržení kvality i přes absenci detailního návrhu projektu a zapojení zákazníka do procesu vývoje.

Důsledkem vývoje agilních metodik je manifest, který definuje základní pravidla agilního vývoje software (Agile Manifesto, 2001):

- Jednotlivci a interakce před procesy a nástroji,
- fungující software před vyčerpávající dokumentací,

- spolupráce se zákazníkem před vyjednáváním o smlouvě,
- reagování na změny před dodržováním plánu.

Součástí manifestu je také 12 principů agilních metodik (Agile Manifesto, 2001):

1. Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
2. Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje.
3. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
4. Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
5. Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.
6. Budujeme projekty kolem motivovaných jednotlivců.
7. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
8. Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
9. Hlavním měřítkem pokroku je fungující software.
10. Agilní procesy podporují udržitelný rozvoj.
11. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
12. Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
13. Jednoduchost, umění maximalizovat množství nevykonané práce, je klíčová.
14. Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.
15. Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti.

2 Metodika Scrum

Scrum patřím mezi inkrementální a iterativní agilní metodiky, která je založena na principech (Šochová, a další, 2014):

- Samoorganizovaných teamů,
- transparentní komunikaci,
- otevřené kultuře založené na spolupráci,
- a sdílení informací.

Ale pro zaručení jeho funkčnosti jsou zavedeny specifické role. Velmi důležitým aspektem jsou i aktivity a artefakty, které nám podávají informaci o tom co je principem této metodiky a jak jí nejlépe využít. (Šochová, a další, 2014).

2.1 Role

2.1.1 Scrum Master

Funguje jako mezičlánek, mezi teamem a okolním prostředím. Má za úkol vytvořit vhodné prostředí a odstínit team od vnějších vlivů. Stará se o funkčnost teamu jako celku a snaží se zvýšit jeho efektivitu, neměl by vystupovat direktivně, ale formou leadera či kouče. Na druhou stranu musí mít právo na změnu a team musí akceptovat jeho rozhodnutí (Šochová, a další, 2014).

2.1.2 Product Owner

Je vlastníkem produktu (v rámci společnosti). Definuje vize a cíle projektu, funguje jako styčný bod a spojuje team se zhotovitelem projektu. Definuje priority a rozhoduje na jakých úkolech má team pracovat nejdříve, na kterých později anebo vůbec. Product Owner musí detailně porozumět potřebám zákazníka, aby jeho rozhodnutí byla zodpovědná a konstruktivní. (Šochová, a další, 2014).

2.1.3 Zákazník

Jedním z principů Scrumu je zapojení zákazníka do procesu vývoje. Tím je zaručeno, že zákazník sám vidí, jakým směrem se projekt ubírá a může si tak určovat své priority i definovat změny funkcionality již v průběhu vývoje. Všechny

své požadavky komunikuje přes Product Owenera směrem k teamu a vice versa (Šochová, a další, 2014).

2.2 Artefakty a aktivity

2.2.1 Sprint

Neboli iterace je základní entitou Scrumu. Je to časově ohraničený úsek, v rámci kterého teamy implementují novou funkčnost. Jeho délka vždy závisí na typu projektu, obecně je však doporučováno zvolit sprint v délce 2 týdny. Podstatou je, že naplánované úkoly v rámci právě běžícího sprintu nemá právo nikdo měnit (Šochová, a další, 2014).

2.2.2 Product Backlog

Je fronta úkolů, které jsou potřeba provést, aby byl produkt dokončen. Samozřejmě, že úkoly se v průběhu vývoje mohou přidávat, odebírat a měnit dle aktuálních priorit. Podskupinou je Sprint Backlog, který je aktivní pouze v rámci konkrétního běžícího sprintu a vymezuje teamový závazek – jakou funkcionalitu team dokončí v rámci zmíněného sprintu (Šochová, a další, 2014).

2.2.3 Plánování sprintu

Product Backlog zpravidla představuje mnoho hodin práce a není v silách teamu jí dokončit v jednom krátkém sprintu. Před začátkem sprintu je tedy nutné provést plánování a založit nový Sprint Backlog, který již představuje objem práce, který se team zavázal stihnout v průběhu jednoho sprintu (Rubin, 2012).

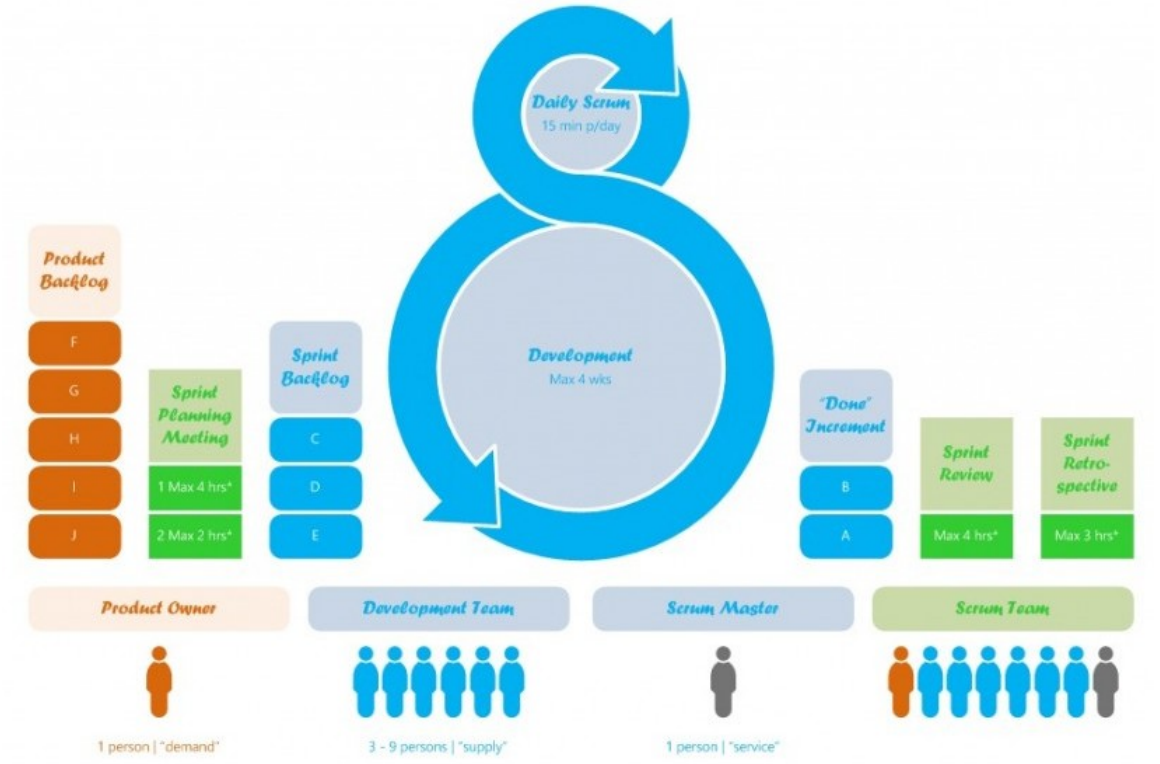
2.2.4 Sprint Review a Retrospektiva

Retrospektiva je nejčastěji provedena ihned po Sprint Rewiew a těsně před plánováním dalšího sprintu. Účel retrospektivy je kontrola a adaptace nastaveného procesu, má odhalit procesní nedostatky, nebo naopak vyzdvihnout a zavést zefektivnění Scrumu (Rubin, 2012).

Na konci každého sprintu jsou důležité dvě aktivity, které slouží ke zhodnocení, jaké pokroky byly udělány v rámci jednoho sprintu. Cílem Sprint Review je kontrola a adaptace produktu, který je vyvíjen. Důležitým bodem Sprint Review je komunikace všech subjektů zainteresovaných ve vývoji produktu. Tématem

komunikace má být právě dokončená funkčnost z hlediska projektu jako celku (Rubin, 2012).

2.3 Průběh Scrumu



Zdroj: http://calvinx.com/2014/05/22/why-scrum-why-agile-development/

Obr. 2 Průběh Scrurmu

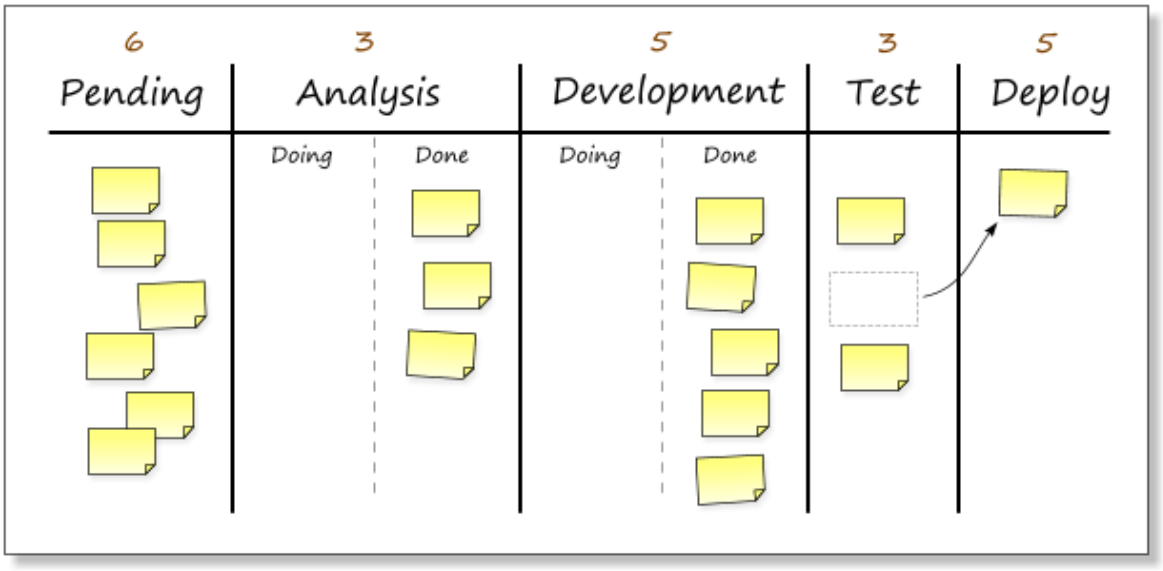
Scrum probíhá tak, že nejprve Product Owner vytvoří Product Backlog. Následně proběhne meeting, kde se dohodne týmový závazek na provedení úkolů z Back Logu za jeden sprint. Úkoly jsou vybírány podle priorit a na základně rozhodnutí Product Ownera. Spustí se sprint v určeném časovém úseku a po dokončení sprintu se provede Sprint Review a retrospektiva, tím zajistíme kontrolu a adaptaci procesu i produktu. Celý proces se opakuje stále dokola v iteracích. Dokončená funkčnost je inkrementální částí Scrumu.

3 Metodika Kanban

Podle definice, je Kanban: „Technika pro vedení a organizování vývoje software s cílem na jeho vysokou efektivitu. Jeho základy jsou v JIT systému společnosti Toyota a navzdory tomu, že vývoj software je kreativní činnost oproti masové produkci automobilů, tak hlavní principy Kanbanu stále mohou být použity při vývoji software.“ (Peterson, 2015)

Kanban v oblasti vývoje software primárně sleduje tyto klíčové principy (Roy, 2014):

1. Vizualizace práce a pracovního postupu za účelem sledování průběhu práce na jednotlivých úkolech napříč systémem Kanban. To nám umožní mít fronty úkolů a vidět, které úkoly zpomalují nebo blokují proces vývoje. Účelem je zefektivnění komunikace a možnosti spolupráce. Příklad takové vizualizace můžeme vidět na obrázku 3.
2. Limitováním množství rozpracovaných úkolů zredukujeme potřebu na neustálé změny priorit úkolů. Limitování času potřebného na projití úkolu systémem Kanban odstraní problémy způsobené přechodem z úkolu na úkol (změna kontextu).
3. Nastavením limitů rozpracovaných úkolů a nastavením standardů vývoje optimalizujeme Kanbanový systém pro hladký průběh vývoje a sběrem dat o průběhu řešení úkolů nám umožní jeho monitoring a analýzu.
4. Dobře nastavený Kanbanový systém přirozeně vede k neustálým zlepšením. Efektivita vývoje je měřena sledováním toku, kvality a průchodnosti úkolů napříč Kanbanovým systém. Závěry měření mají opět vést k neustálému zlepšení.



Zdroj: http://kanbanblog.com/explained/image/kanban-board-1.png

Obr. 3 Vizualizace Kanban systému

Pokud srovnáme Scrum a Kanban, tak výhody Kanbanu jsou následující: flexibilita, zaměření na kontinuální řešení úkolů, vysoká kvalita, produktivita a efektivnost vývoje při dobře nastaveném Kanbanu, odstranění neproduktivní práce (Roy, 2014).

Kanban je obtížnější na implementaci, protože samotný Kanban nelze definovat jako metodiku, metodika se z něj stává právě až po implementaci do vývojového procesu, je to jen sada základních doporučení (Šochová, a další, 2014).

4 Současný stav ve vybrané společnosti

Nežli přistoupíme k samotnému plánování přechodu a posléze i implementaci nevhodnější metodiky vývoje software je nutné nejprve analyzovat současný stav ve vybrané společnosti, dále jen společnosti. Analýza současného stavu je důležitá, abychom měli přehled v následujících bodech:

- Jak společnost funguje a jaké procesy ve společnosti probíhají,
- jakou oblastí vývoje se společnost zabývá
- jak velké projekty se ve společnosti řeší
- jaké jsou požadavky zákazníků.

4.1 Představení společnosti

Společnost, ve které jsem navrhoval proces optimalizace vývoje software, se soustředí na malé až středně velké klienty a zabývá se zejména tvorbou webových stránek, vývojem aplikací, portálů, e-shopů a informačních systémů. Společnost zatím působí pouze na tuzemském trhu necelý 1 rok od svého založení, tedy relativně krátce, ale i přesto si dokázala získat dostatek klientů, pro které provádí buď přímý vývoj a údržbu, nebo zajišťuje jejich outsourcing[2].

4.2 Analýza současného stavu

Společnost je v neustálém růstu a klienti, potažmo projekty stále přibývají, tak vznikají nové komunikační problémy na jedné straně mezi společností a klienty a na straně druhé začíná být problematická i vnitrofiremní komunikace. Největší problém však spatřuji, že s rostoucím počtem interních i externích vývojářů a subdodavatelů se vytrácí možnost jak sledovat a predikovat průběh projektu a s tím spojená postupná ztráta v schopnosti pružně reagovat na požadavky klientů a zároveň udržet efektivitu vývoje i dostatečnou kvalitu kódu a dokumentace v rámci jednotlivých projektů.

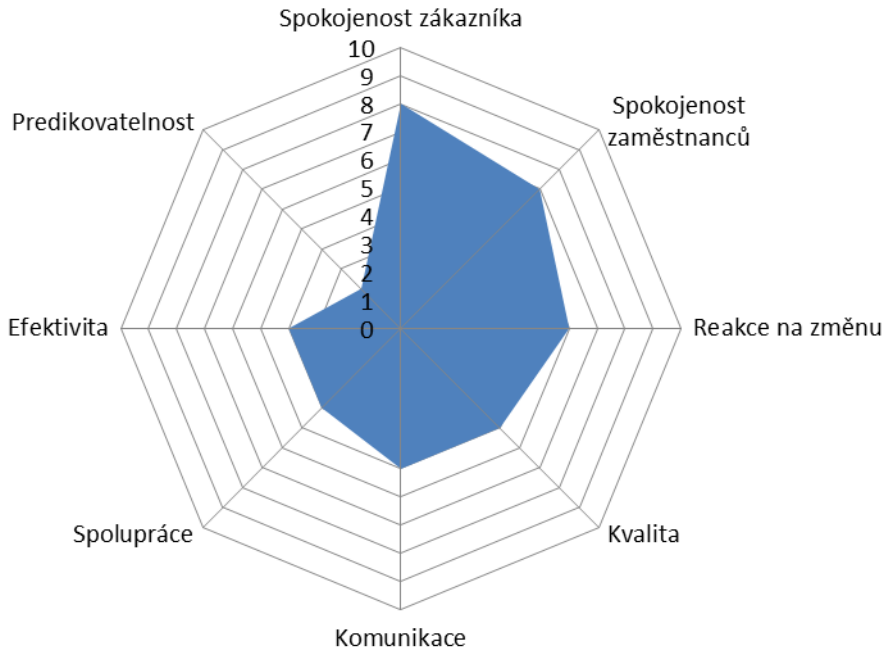
Pro zhodnocení současného stavu společnosti jsem použil metodu, která spočívá v ohodnocení následujících osmi oblastí:

- Kvalita,

[2] Zajištění provedení činností, které si je společnost schopna provést sama, externí společností.

- efektivita,
- spokojenost zaměstnanců,
- reakce na změnu,
- předvídatelnost,
- spolupráce,
- komunikace,
- spokojenost zákazníka.

Jednotlivé oblasti ohodnotíme ve škále 1 až 10 bodů, kde ohodnocení 1 bodem odpovídá velmi nízké spokojenosti a ohodnocení 10 body znamená, že jste s danou oblastí velmi spokojeni. Následně stačí ohodnocené oblasti seřadit sestupně podle obodování a vynést jednotlivé oblasti do síťového grafu (Šochová, a další, 2014).



Obr. 4 Zhodnocení současného stavu ve společnosti

Obrázek 4 zobrazuje současný stav v mnou vybrané společnosti. Jak si můžeme povšimnout, tak zákazníci jsou zatím s výsledky jejich projektů spokojeni, jelikož

interní problémy se dostávají směrem k zákazníkům velmi pomalu a z jejich pohledu se zdá, že je vše v pořádku.

Dalším důležitým aspektem jsou zaměstnanci, potažmo programátoři. Bez spokojených a nadšených zaměstnanců nemůže fungovat žádná úspěšná společnost. Vysoká spokojenost zaměstnanců v této společnosti je nyní především zapříčiněna jejich vysokou autonomií, protože ve společnosti nejsou zatím vytvořené pevné standardy a nastavené procesy, včetně těch kontrolních, tak aby každý zaměstnanec odváděl zodpovědně, kvalitně a efektivně svojí práci.

S tím souvisí i snížená schopnost reagovat na změny v zadání od klientů. Jelikož nikdo nenese přímou zodpovědnost za dokončení konkrétního a již přiřazeného úkolu, proto se staré úkoly často nechávají rozpracované a bez dokumentace jejich stavu a přistupuje se k řešení nových úkolů, které se zdají být momentálně důležitější.

Tímto velmi trpí kvalita, protože přesouvání starších, tak i aktuálních rozpracovaných úkolů mezi programátory zanáší značné množství chyb do aplikací. Další příčinou, která snižuje kvalitu výsledného produktu, je i absence komplexních testů aplikace, která by odhalila většinu chyb před nasazením aplikace do ostrého provozu, aby se aktivně předcházelo nalezení chyby zákazníkem nebo potažmo jeho klientem.

Komunikace jdoucí ruku v ruce se spoluprací je integrální součást každé společnosti a to zejména té softwarové. V současnosti je komunikace ve společnosti zajištěna osobně, přes e-mail, instant messaging[3] a pomocí telefonu.

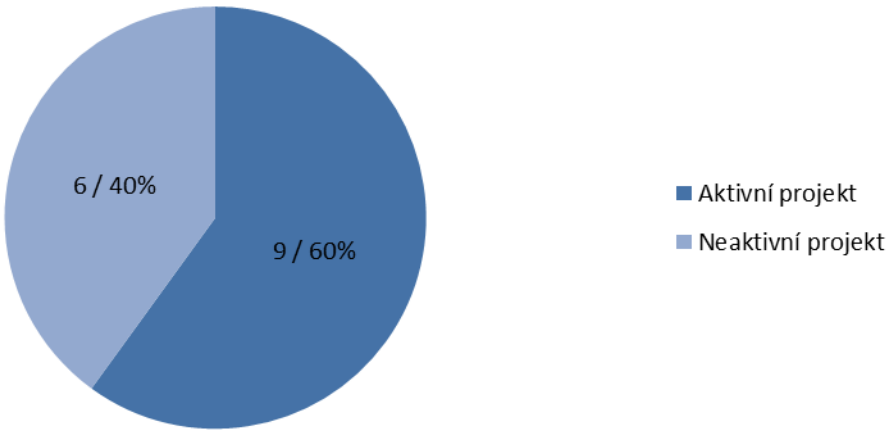
Nedostatečná komunikace mezi programátory, nízká úroveň teamové spolupráce a nízká kvalita zdrojových kódů aplikací vede k redundantním činnostem, které velmi značně snižují efektivitu práce a všechny tyto výše zmíněné oblasti působí tak, že predikovatelnost (časová, nákladová, atd…) projektů je velmi obtížná a vznikají značné posuny v termínech dokončení projektů, tak i velmi časté překročení rozpočtů u projektů.

[3] Komunikace přes internet v reálném čase pomocí textových zpráv, ale i jinými způsoby (například pomocí videohovoru).

4.2.1 Struktura projektů ve společnosti

Na obrázku 5 můžeme vidět zastoupení aktivních a neaktivních projektů ve společnosti. Celkem společnost od svého založení řešila anebo řeší 15 projektů, z toho je 9 projektů aktivních, tj. 60% a 6 projektů již neaktivních, tj 40%.

Podle vnitřní směrnice je aktivní většina projektů, vůči kterým má společnost závazky vůči klientovi. U těchto projektů probíhá aktivní vývoj nebo údržba, anebo alespoň občasná údržba. Neaktivní projekty jsou takové projekty, které jsou vedením společnosti prohlášeny za dokončené, tedy bez závazků vůči klientům, nebo pozastaveny z různých důvodů, například to může být soudní spor se zadavatelem projektů.



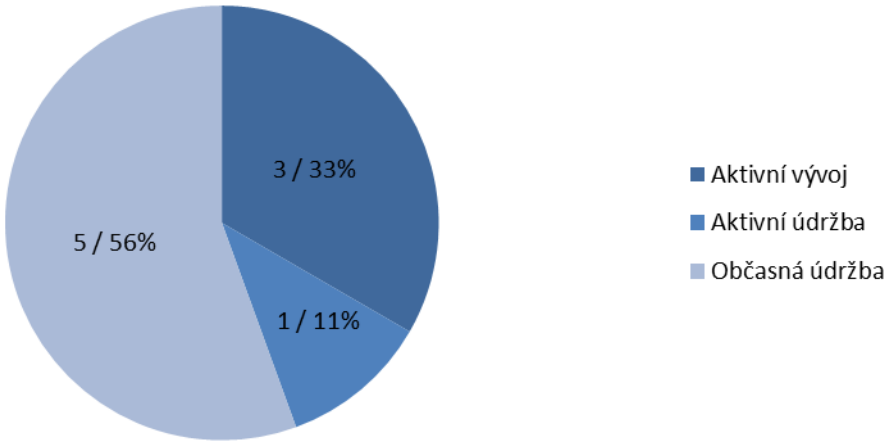
Obr. 5 Zastoupení aktivních a neaktivních projektů

Z obrázku číslo 6 můžeme vyčíst strukturu projektů aktivních, projektů s aktivní údržbou a projektů s občasnou údržbou. Společnost má v současnosti 3 projekty, tj. 33% aktivní, 1 projekt, tj. 11% s aktivní údržbou a 5 projektů, tj. 56% s občasnou údržbou.

Projekt, u kterého se neustále přidává nová funkcionalita podle zadání, vnitřní směrnice označuje jako projekt s aktivním vývojem. Aktivní údržba podle vnitřní směrnice znamená, že projekt je již podle zadání dokončen, je spuštěn v produkčním režimu[4] a dochází pouze k opravám nalezených chyb. U projektů s občasnou údržbou se jedná o projekty, u kterých stejně jako u projektů s aktivní údržbou nedochází k implementaci nové funkcionality a již nejsou ani kritické

[4] Produkční režim znamená, že aplikaci již používají zákazníci za účelem, pro který byla vyvinuta.

z hlediska rychlosti oprav, protože tyto projekty jsou již otestovány v reálném provozu a k nalezení chyb dochází jen velmi zřídka.



Obr. 6 Zastoupení jednotlivých fází projektů

4.2.2 Aplikované metodiky vývoje software

V současnosti je u všech projektů zavedeno verzování zdrojových kódu aplikace centralizovanou formou pomocí software Subversion a všechny projekty jsou řízené velmi omezenou a upravenou formou vodopádového modelu, který však v současnosti selhává.

5 Návrhy na optimalizaci

Na základě analýzy současného stavu ve společnosti jsem optimalizaci vývoje software rozdělil do následujících dílčích kroků:

1. Převedení zdrojových kódů aplikace do distribuovaného VCS,
2. zavedení vhodného nástroje pro projektový management,
3. implementace vhodných agilních metodik vývoje software.

Všechny výše uvedené kroky jsou klíčové pro nastavení správných procesů ve společnosti, které zajistí odstranění slabých stránek u oblastí uvedených v kapitole 4.2.

Implementace vhodných metodik vývoje software je uvedena níže v samostatné kapitole, jelikož se jedná o obsáhlejší problematiku.

5.1 Převod zdrojových kódů

Jelikož budeme zavádět metodiky vývoje software, které plně těží z týmové spolupráce na projektech, tak musíme mít nástroj, který nám umožní efektivně a kolaborativně pracovat na úrovni zdrojových kódů aplikace.

Jako nevhodnější nástroj jsem zvolil Git. Hlavní výhody pro nahrazení Subversion za Git spatřuji v jednoduchosti větvení a slučování změn ve zdrojových kódech aplikace, znalosti Gitu u většiny současných programátorů ve společnosti oproti Subversion, pro jeho přívětivější uživatelské rozhraní a kvalitu nástrojů využívajících Git. Zde se především jedná o podpůrné nástroje využívané při vývoji aplikace a zejména široká škála možností jak aplikaci nasadit na server a spustit v produkčním režimu. Výše uvedené výhody pro nasazení Gitu byly diskutovány napříč společností a jsou závěrem kolektivní dohody.

Další důvody proč nasadit Git jsou umožnění distribuované správy verzí zdrojových kódů, proto jsou operace nad zdrojovými kódy aplikace velmi rychlé. Většina stávajících komponent (knihoven) používaných při vývoji aplikací je založena jako open-source software a je spravována právě také pomocí Gitu a umístěna na Githubu, odkud mohou být velmi jednoduše spravovány pomocí nástroje pro správu závislostí aplikace.

V současnosti se jako jediná nevýhoda hovořící proti nasazení Gitu jeví nemožnost řídit přístupová práva na úrovni jednotlivých souborů a adresářů v rámci standartní distribuce Gitu. Z hlediska společnosti se ukázala tato nevýhoda lichá. Současné projekty totiž tuto funkcionalitu nevyžadují a v budoucnosti jí lze jednoduše zajistit externími nástroji.

Samotné převedení správy zdrojových kódů aplikace pod Git je v současnosti velmi jednoduchý a přímočarý krok, protože historie změn projektu je nyní využívána spíše jako záloha a tak není potřeba její zachování. S tím souvisí i změna ve stylu zapisování změn do Gitu.

Každá zapsaná změna do Gitu musí následovat následující pravidla (Beams, 2014):

- Měnit pouze jednu a konkrétní věc, například přidávat nebo odebírat určitou funkčnost.
- Obsahovat popisný titulek.
- V těle změny obsahovat popis co a proč bylo změněno.

S převodem zdrojových kódů souvisí i standardizace stylu zdrojových kódů pomocí vnitřního předpisu ve společnosti a jejich automatický převod do této formy ještě před samotným převodem. Jelikož se jedná spíše o technické řešení, tak se tomuto tématu nebudu věnovat podrobněji.

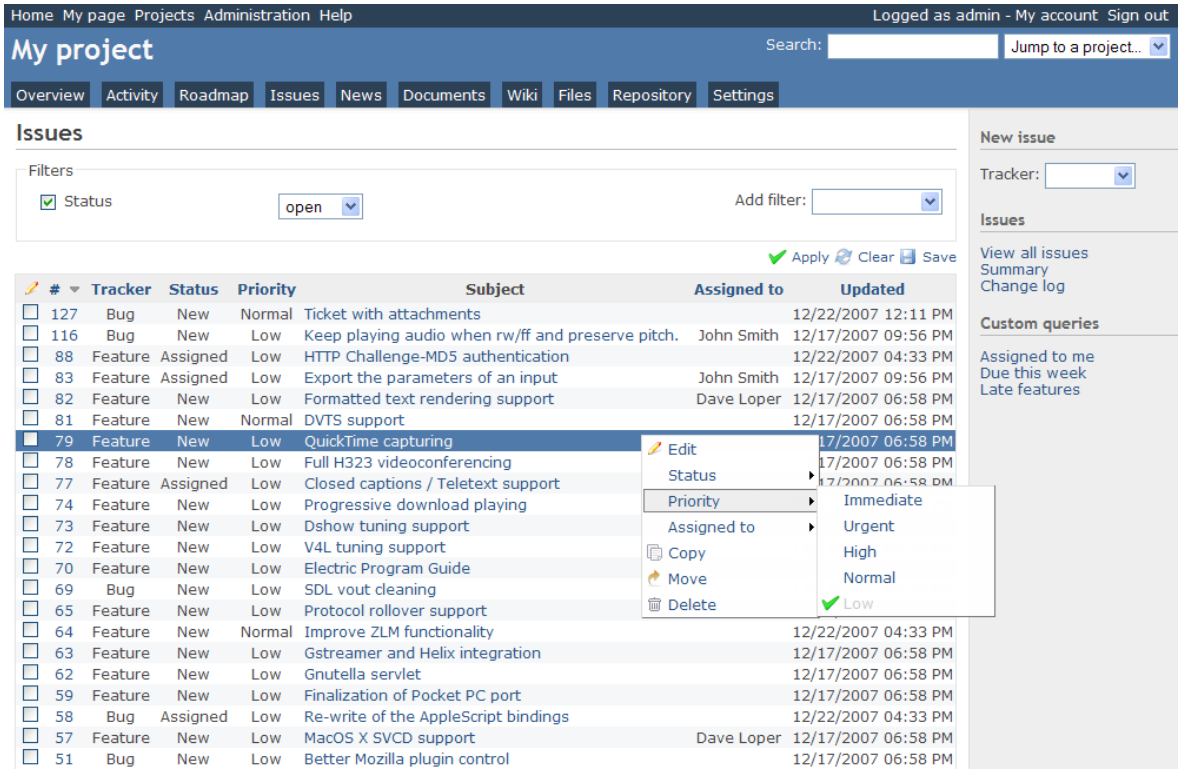
5.2 Nástroj pro projektový management

Vhodný podpůrný nástroj pro řízení vývoje projektu (aplikace) je pro vedoucí pracovníky klíčový nástroj pro sledování všech aspektů z hlediska efektivity, kvality a komunikace. Jako nejlepší možnost jsem zvolil aplikaci Redmine, protože splňuje všechny naše požadavky na projektový management, které jsou popsány v následujících odstavcích.

Díky podpoře více projektů, nám odpadá potřeba mít pro každý projekt vlastní nástroj pro projektový management a všechny projekty jsou přehledně spravovány centrálně z jedné aplikace. Aplikace je přístupná všem jako webová stránka, tím se velmi zjednoduší přístup do této aplikace.

Velmi důležitá vlastnost je i detailní nastavení přístupových práv pro každého uživatele. Tím můžeme umožnit přístup do aplikace i klientům a ti mají možnost nahlížet pouze do svých projektů. Na základě toho mohou klienti získávat informace o svých projektech. Takto se velmi zvýší efektivita komunikace mezi společností a klientem. Klienti totiž například vidí, v jaké fázi se projekt nachází, jaké úkoly jsou nové, hotové nebo rozpracované včetně hodinové dotace.

Úkolům můžeme nastavit prioritu a přiřazení úkolů na konkrétního zaměstnance zajistí, že někdo za úkol nese zodpovědnost, aby byl vyhotoven ve stanoveném termínu a nastalé komplikace komunikoval směrem k nadřízenému, nebo klientovi. U konkrétního úkolu musí být vedena kompletní historie změn, komunikace, všech souvisejících dokumentů a případné cenné poznatky z průběhu řešení úkolu mají být zaznamenávány formou projektové Wiki[5]. Příklad takového seznamu s úkoly projetu, včetně jejich priorit, stavů a vlastníků můžeme vidět na obrázku 7.



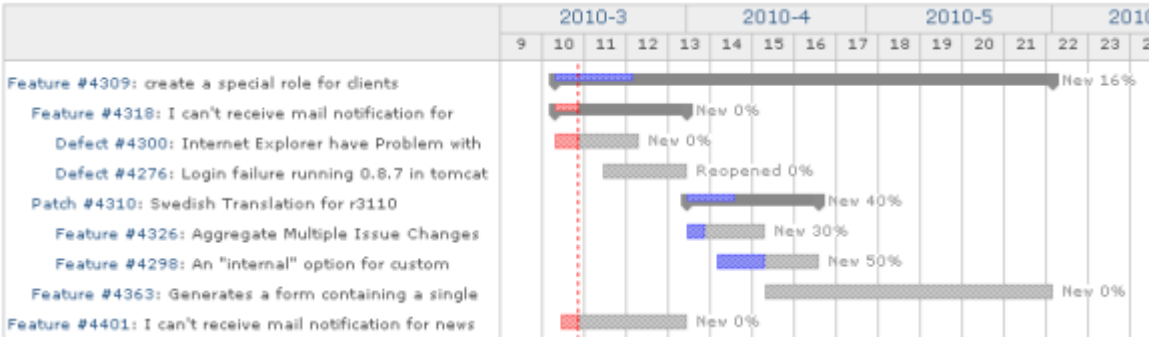
Zdroj: http://www.redmine.org/screenshots/issue_list.png

Obr. 7 Seznam úkolů v Redmine

[5] „Wiki je označení webů (nebo obecněji hypertextových dokumentů), které umožňují uživatelům přidávat obsah podobně jako v internetových diskusích, ale navíc jim také umožňují měnit stávající obsah“ (Wikipedie, 2015)

Propojení Gitu se systémem hlášení chyb v software, který je standardní součástí Redmine, umožňuje jednoduše sledovat, kolik chyb se v aplikaci nachází a jak oprava pokračuje. Tato funkčnost slouží jak interním zaměstnancům, tak zákazníkům, kteří mohou nahlásit chyby v software. Na nahlášené chyby je v Redmine nahlíženo podobně jako na úkoly. Propojení s Gitem umožní chybu propojit s konkrétní změnou ve zdrojových kódech aplikace a tak velmi jednoduše dohledat co, proč a jak bylo opraveno.

Ganttův diagram v Redmine (viz Obr. 8) přehledně zobrazuje jednotlivé úkoly v projektu na časové ose, tak můžeme jednoduše kontrolovat, jak projekt celkově postupuje vůči stanoveným termínům dokončení úkolů.



Zdroj: http://www.redmine.org/attachments/download/3481/subtasking.png

Obr. 8 Ganttův diagram v Redmine

Využitím Redmine získáme schopnost efektivně a centrálně řídit všechny aktivní projekty a to jednak z hlediska zdrojů materiálních i zdrojů lidských. Jsme tedy schopni lépe a zodpovědněji predikovat průběh vývoje projektu a to až na úrovni jednotlivých úkolů. Historie komunikace a přiložená dokumentace zvyšuje efektivitu a umožňuje každému jednoduše zjistit všechny podrobnosti náležící k úkolu.

6 Vhodné metodiky vývoje software

Pro výběr agilních metodik vývoje software bylo rozhodnuto na základě oblastí, které jsou integrální součástí agilních metodik vývoje software, jsou to zejména flexibilita, efektivita, předvídatelnost a kvalita, přestože to znamená změnit kompletní přístup k vývoji aplikací oproti nyní využívanému vodopádovému modelu. Musíme totiž společnost přeorientovat z hierarchického řízení na společnost orientovanou na problém a nemáme nikde přesně specifikován postup jak agilní metodiku zavést (Šochová, a další, 2014).

Výhoda agilních metodik právě spočívá v teamovém přístupu, principu tahu konvergujícího k funkčnímu software přednostně před dokumentací, spoluprací se zákazníkem místo lpění nad přesným zněním zadání projektů, bezproblémovým akceptováním změn v zadání v průběhu vývoje aplikace a širokých možnostech přizpůsobení projektu, pro které jsou zaváděny (Scrum Alliance, 2015).

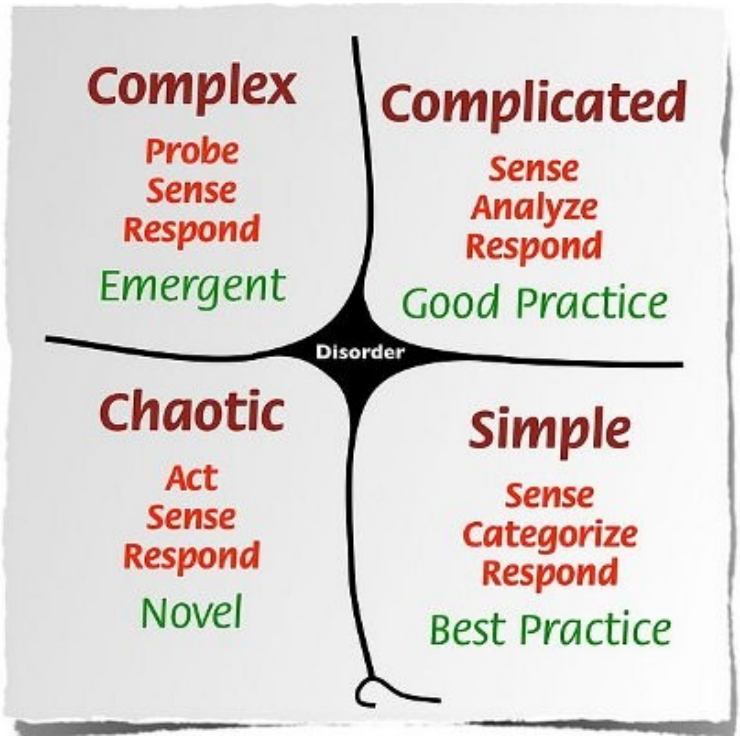
Akceptace změn v průběhu vývoje je hlavním důvodem, proč ve společnosti zavést agilní metodiky vývoje software, protože v současnosti využívaný vodopádový model označuje toto jako chybou návrhu aplikace a není tomuto uzpůsoben, ale kvůli typům a struktuře projektů ve společnosti se stává, že změny v zadání přichází často, i v průběhu vývoje aplikace.

Z hlediska struktury projektů (viz Obr. 6) jsem navrhl optimalizaci vývoje software tak, že projekty ve fázi občasné údržby budou v současnosti ponechány bez změny, jelikož z hlediska společnosti nepředstavují takřka žádné riziko z nedodržení termínů dokončení, či nákladů na vývoj. Projekty s aktivní údržbou budou vyvíjeny užitím metodiky Kanban a aktivní projekty metodikou Scrum.

6.1 Metoda Cynefin

Rozhodnutí, které metodiky budou použity, je založeno na metodě Cynefin. Tato metoda funguje na principu racionálního vyhodnocení stavu, ve kterém se právě projekty nachází. Metoda definuje a porovnává charakteristiky 5 oblastí (viz Obr. 9): snadná (simple), komplikovaná (complicated), chaotická (chaotic), komplexní (complex) a zmatečná (disorder). Ve zmatečné oblasti se nacházíme, pokud

nemůžeme zodpovědně rozhodnout, že stav projektu patří do jedné z předchozích čtyř (Rubin, 2012).



Zdroj: http://scrumandkanban.co.uk/wp-content/uploads/2013/11/Cynefin-405x400.jpg

Obr. 9 Pět oblastí Cynefin metody

Podle Lowea (2013) se ve snadném stavu nachází projekty u kterých je jasný vztah mezi cílem a jeho dopadem. Vhodným scénářem jak tyto projekty řešit, je porozumění problematice (sense), rozdělení do jednotlivých úkolů (categorize) a provedení těchto úkolů (respond). Toto odpovída modelu vodopádového přístupu k vývoji aplikací.

U komplikovaných projektů je potřeba nejprve analyzovat vztahy mezi cíly a jejich důsledky. Ideální scénář řešení je porozumění problematice (sense), analýza možných přístupů k vyřešení úkolů (analyze) a jejich následné vyřešení (respond). Zde je vhodné nasadit agilní metodiku typu Kanban.

U komplexních projektů je vhodné nasadit metodiku Scrum, jelikož nejlépe odpovída postupu, kde dochází ke zjištění informací o úkolech (probe), jejich následnému pochopení (sense) a vyřešení (respond). Zpravidla zde nelze zjistit vztah mezi cíly a jejich dopady před dokončením úkolů a jejich otestování v produkčním nasazení.

Pro projekty v chaotickém stavu je příznačné, že vztah mezi cíly a jejich dopady nelze determinovat. V takovémto případě je doporučeno projekt stabilizovat (act) a pak převést do jedné z výše zmíněných oblastí (sense, respond). Nelze doporučit konkrétní metodiku, zde je pouze důležitá rychlá akce vedoucí ke stabilizaci, například odstranění nečekaného výpadku kritické služby.

Zmatečný stav nastává, pokud se nemůžeme rozhodnout v jaké oblasti se nacházíme, většinou je to způsobeno rozhodováním založeném na osobních preferencích.

6.2 Metodika Scrum

Bude zavedena pro projekty s aktivním vývojem, protože svojí charakteristikou odpovídají komplexní oblasti v metodě Cynefin. Programátoři budou reorganizování do 3 teamů. Velikost teamů bude 2 programátoři + 1 Scrum Master. Product Owner bude z hlediska nedostatku lidských zdrojů zastřešovat všechny tři projekty najednou. Iterace budou na začátku 2 týdenní a poté se u každého projektu provede jejich optimalizace.

6.3 Metodika Kanban

Je ve společnosti vhodná nasadit tam, kde se jedná o komplikovanou oblast metody Cynefin a to splňují projekty s aktivní údržbou. Navíc se zde nejedná o kontinuální proces vývoje, ale práce je prováděna nárazově s hlášením o chybě. To je další fakt hovořící ve prospěch Kanbanu.

Pro vizualizaci pracovního postupu se ve společnosti bude využívat software Trello. Limity jednotlivých front budou nastaveny při přechodu na novou metodiku. Pro kontinuální optimalizační proces limitů front se bude pravidelně sledovat čas dokončení úkolu v závislosti na počtu úkolů rozpracovaných. Jelikož je ve společnosti pouze jeden projekt s aktivní údržbou, bude na tuto činnost vyhrazen pouze jeden programátor. Jeho další pracovní náplní bude udržovat i projekty s občasnou údržbou a provádět případné testování nové funkcionality u projektů s aktivním vývojem.

7 Aplikační plán

Spočívá ve vypracování časového harmonogramu pro zavedení změn do společnosti a z jejího hlediska je na něj nahlíženo jako na samostatný a specifický projekt. Agenda projektu bude vedena v software MS Project, jelikož s tímto software mám zkušenosti již z jiných projektů a umožňuje v rámci řízení projektu flexibilně měnit všechny jeho parametry.

V současnosti má aplikační plán jen hrubé obrysy a je ve fázi detailní specifikace. Pouze byla vedením odsouhlasena možnost začít s přípravou na převod zdrojových kódů a s tím spojené proškolení všech zaměstnanců. Po dokončení plánování se přistoupí k nasazení nástroje pro projektový management a dojde k převedení veškeré projektové agendy do tohoto software. Poté nastane nejtěžší fáze optimalizace – zavedení nových metodik vývoje software.

Závěr

V této bakalářské práci byly porovnány metodiky pro vývoj software vhodné pro optimalizaci procesu vývoje software ve vybrané společnosti. Dále byla zpracována analýza současného stavu s návrhem na optimalizace procesu vývoje software ve vybrané společnosti. Doporučení k zvolení konkrétních metod bylo založeno na metodě Cynefin.

Na základě hypotézy uvedené v úvodu této bakalářské práce byla potvrzena teze, že v současnosti nejvhodnějšími agilními metodikami vývoje software ve vybrané společnosti jsou Scrum a Kanban. Závěr tedy je, že agilní metodiky doplněné o vhodné nástroje pro podporu vývoje software jsou vhodné pro zavedení v této vybrané společnosti a nahrazením v současnosti použitého vodopádového modelu se zvýší efektivita, kvalita, flexibilita a predikovatelnost vývoje software. Tento závěr se opírá zejména o výstupy z analýzy současného stavu a metody Cynefin, pomocí které jsem vybral vhodné metodiky vývoje software vyhovující současné struktuře projektů ve vybrané společnosti.

Jako další možnost rozvoje řešené problematiky spatřuji ve zpracování standardů pro zavedení procesu automatického testování aplikací. V současnosti je ponechána každému jistá míra autonomie, jak testování aplikace provádět. Další oblastí, ve které je prostor pro zlepšení, jsou nástroje pro automatické nasazení aplikací do produkčního režimu, či přímo implementace systémů pro kontinuální integraci aplikací.

Seznam literatury

Agile Manifesto. 2001. Manifest Agilního vývoje software. *Agile Manifesto.* [Online] 2001. [Citace: 25. 9 2015.] http://www.agilemanifesto.org/iso/cs/.

Beams, Chris. 2014. How to Write a Git Commit Message. [Online] Chris Beams, 31. 8 2014. [Citace: 23. 10 2015.] Dostupné z: http://chris.beams.io/posts/git-commit/.

Best Price Computers. 2015. Application Development. *Best Price Computers.* [Online] 2015. [Citace: 30. 8 2015.] http://www.bestpricecomputers.co.uk/glossary/application-development.htm.

Fowler, Martin. 2003. *Refaktoring: Zlepšení existujícího kódu.* Praha : Grada, 2003. ISBN 80-247-0299-1.

Lowe, David. 2013. Cynefin. *Scrum & Kanban.* [Online] 14. 11 2013. [Citace: 18. 9 2015.] http://scrumandkanban.co.uk/cynefin/.

Peterson, David. 2015. What is Kanban. *Kanban Blog.* [Online] 2015. [Citace: 1. 12 2015.] http://kanbanblog.com/explained/.

Roy, Vandana. 2014. Scrum Versus Kanban. *Scrum Alliance.* [Online] 10. 7 2014. [Citace: 1. 12 2015.] https://www.scrumalliance.org/community/articles/2014/july/scrum-vs-kanban.

Rubin, Kenneth. 2012. *Essential Scrum.* 1. vyd. Ann Arbor : Edwards Brothers Malloy, 2012. ISBN 0137043295.

Scrum Alliance. 2015. Scrum Values. *Scrum Alliance.* [Online] 2015. [Citace: 12. 8 2015.] https://www.scrumalliance.org/why-scrum/core-scrum-values-roles.

Schwalbe, Kathy. 2011. *Řízení projektů v IT: kompletní průvodce2011.* 1. vyd. Brno : Computer Press, 2011. ISBN 978-80-251-2882-4.

Šochová, Zuzana a Kunce, Eduard. 2014. *Agilní metody řízení projektů.* 1. vyd. Brno : Computer Press, 2014. ISBN 978-80-251-4194-6.

Testování softwaru. 2015. Testování softwar. *Vodopádový model.* [Online] 2015. [Citace: 2. 12 2015.] http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/vodopadovy-model/.

Wikipedie. 2015. Wiki. [Online] Wikipedie, poslední aktualizace 6. 12 2015. [Citace: 5. 11 2015.] Dostupné z: https://cs.wikipedia.org/wiki/Wiki.

Seznam obrázků a tabulek

Seznam obrázků

ANOTAČNÍ ZÁZNAM

AUTOR	**Karel Novotný**		
STUDIJNÍ OBOR	**6208R088 Podniková ekonomika a management provozu**		
NÁZEV PRÁCE	**Moderní metodiky teamového vývoje software**		
VEDOUCÍ PRÁCE			
KATEDRA	**KLRK - Katedra logistiky a řízení kvality**	**ROK ODEVZDÁNÍ**	**2015**

POČET STRAN	**33**
POČET OBRÁZKŮ	**9**
POČET TABULEK	**0**
POČET PŘÍLOH	**0**

STRUČNÝ POPIS	**Záměrem této bakalářské práce je návrh na optimalizaci řízení pracovních teamů v rámci flexibilního vývoje software.** **Práce obsahuje rešerši v současnosti nejpoužívanějších metodik používaných pro vývoj software. Ukázkově zpracovává analýzu současného stavu ve vybrané společnosti a představuje metodu Cynefin, která je vhodná pro racionální výběr vhodné metodiky vývoje software v závislosti na typu projektu.** **Součástí práce je také popis implementace nástrojů pro projektový management, bez kterých nelze projekty adekvátně řídit.** **Závěrem práce je, že v současnosti jsou pro vývoj komplexních a komplikovaných projektů vhodnější agilní metodiky vývoje software typu Scrum a Kanban.**
KLÍČOVÁ SLOVA	**Vývoj software, Analýza, Agilní metodiky, Scrum, Kanban, Cynefin, Projektový management, VCS**

PRÁCE OBSAHUJE UTAJENÉ ČÁSTI: Ne

ANNOTATION

AUTHOR	**Karel Novotný**		
FIELD	**6208R088 Business Management and Production**		
THESIS TITLE	**Modern Methodologies of Software Development in Teams**		
SUPERVISOR			
DEPARTMENT	**KLRK - Department of Logistics and Quality Management**	**YEAR**	**2015**

NUMBER OF PAGES	**33**
NUMBER OF PICTURES	**9**
NUMBER OF TABLES	**0**
NUMBER OF APPENDICES	**0**

SUMMARY	**The purpose of this bachelor thesis is „Optimization proposal of the software development in teams with focus on flexibility“.** **The thesis include research of current most used software development methodologies and example analysis of current state in selected company. Thesis also show Cynefin method which is appropriate for rational selection of suitable software development methodology in dependency on project type.** **Part of thesis is description of project management tools. This tools is key for efficient project management.** **In conclusion, thesis shows that agile software development methodologies like Scrum and Kanban are more efficient for complex and complicated projects.**

KEY WORDS	**Software development, Analysis , Agile methodologies, Scrum, Kanban, Cynefin, Project management, VCS**

THESIS INCLUDES UNDISCLOSED PARTS: No