Abstrakt

Ve své bakalářské práci se zaměřím na zpracování a porovnání agilního a vodopádového řízení projektů. Budu se věnovat tomu, co obsahuje samotné řízení projektů. Jaké lze použít postupy a jaké výhody či nevýhody lze najít u agilního či vodopádového přístupu. Rozvinu úlohy jednotlivých členů týmu od vedení přes scrum mastery až po řadové členy týmů. Výsledkem mé práce je odpověď na otázku, zda mnou zkoumaná firma je vhodná agilní přístup řízení projektů.

Abstract

In my thesis, I will focus on the processing and comparison of the waterfall and agile project management methods. I will describe the actual project management, what methods can be used, and what advantages or disadvantages can be found in the agile and waterfall approach. I will talk about the team members starting from leadership roles over the scrum master to ordinary team members. The result of my work will be the answer to the question of whether the company surveyed by me is suitable for the agile project management approach.

Klíčová slova

Projekt, řízení projektů, agilní metody, vodopádové metody

Key words

Project, project management, agile methods, waterfall methods

Bibliografická citace

RAJČAN, M. *Návrh na zavedení agilní metodiky pro řízení projektu*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2017. 54 s. Vedoucí bakalářské práce Ing. et Ing. Pavel Juřica, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 29. května 2017

.....

podpis studenta

Poděkování

Chtěl bych poděkovat vedoucímu své bakalářské práce Ing. et Ing. Pavlu Juřicovi, Ph.D. za cenné rady, vhodné připomínky a odborné vedení při zpracování této bakalářské práce. Poděkování také patří všem, kteří mi svými odbornými připomínkami pomohli.

OBSAH

ÚVOD

Agilní metodika, jako nástroj řízení je poměrně nový přístup k řízení projektu. Oblast agilního přístupu se stále zdokonaluje a vyvíjí. Hlavní myšlenka se ovšem nemění a jedná se o přístup, který se snaží v pravidelných krátkých intervalech dodávat inovace, které si zákazník definoval. Umožňuje vhodný přístup k projektům, u kterých není jistá predikce úspěchu, či není ještě zcela definovaný cíl, či zákazník nemá vizi toho, co chce mít.

Vzhledem k tomu, že dodávka probíhá v pravidelných krátkých časových úsecích, je možné velmi rychle uvést do provozu základní části požadovaného produktu, který má pro zákazníka nejvyšší přidanou hodnotu a navíc mu velmi často začíná ihned generovat zisk.

Velmi důležitou věcí je aktivní angažovanost zákazníka. Nestačí, aby agilně fungovala jen společnost, která má za úkol produkt dodat, ale i zákazník, nebo aspoň část zaměstnanců. V tomto ohledu musí agilně pracovat minimálně zaměstnanci, kteří mají za úkol spolupracovat s dodávající společností.

Agilní metodika sebou přináší nižší pravděpodobnost vyskytnutí chyby softwaru, z důvodů dodávaní menších části nově požadovaného produktu. Tato skutečnost je velmi pozitivně přijímána na straně zákazníka, neboť snižuje šanci špatných dopadů na jeho zákazníky. Navíc kvalita, stálá a správná funkce softwaru je jedním z klíčových hodnotících parametrů.

1 CÍL PRÁCE

Cílem bakalářské práce je popsání rozdílu mezi vodopádovým a agilním přístupem k vývoji softwaru, provedení analýzy současného stavu řízení projektu ve vybrané společnosti, která pro vývoj softwaru využívá vodopádovou metodiku. Následně se zaměřím na zavedení nové metodiky řízení projektů pomocí agilního přístupu. Zhodnocení přínosu změny jak pro vybranou společnost, tak i pro jejího zákazníka.

2 TEORETICKÁ VÝCHODISKÁ PRÁCE

V této kapitole se obecně podívám na funkci a význam projektového řízení. Zaměřím se na jeho zásady a způsoby vedení. Dále se budu podrobně zabývat problematikou vodopádového způsobu řízení projektu. Podíváme se na historii a souvislosti vzniku této metody. Další důležitou a podrobnou částí bude zpracování agilní metody řízení projektu. Tato metoda je poměrná nová, proto se podíváme, co předcházelo jejímu vzniku a kde jsou její výhody.

2.1 Projekt

Projekt je výsledek materiální nebo nemateriální povahy založený na strategickém plánu, navržený, organizovaný a realizovány pod vedením osoby v zájmu vlastníka nebo zadavatele. Projekt je aktivita omezená v čase, realizována pouze jedenkrát bez opakování se značným množstvím rysů, ke kterým patří:

- výsledek musí sloužit po celou dobu, kterou vyžaduje zadavatel,
- úspěch projektu při jeho zadání není jasný,
- trvání je přesně časově omezeno,
- zdroje jsou konečné,
- projekt má jen jeden výsledek (FIALA, 2004, s. 12-13).

2.1.1 Řízení projektu

Řízení projektů je soubor modelů, metod, postupů, nástrojů a technik pro plánování a řízení realizace složitých projektů. Projekt často obsahuje tyto rysy:

- projekt má definován začátek a konec,
- existuje vysoká míra nespokojenosti,
- používají se pružné organizační struktury,

- složení řešitelského týmu projektu je proměnlivé (FIALA, 2004, s. 13).

Implementace projektového řízení vede ke snižování nákladů, zvyšování rychlosti a kvality. Snižování nákladů a zvyšování rychlosti spolu úzce souvisí. Všechny tyto pozitivní efekty vyplývají z odstraňování bariér mezi podnikem a jeho partnery. Odstranění bariér vede k eliminaci opakování různých činností, které mohou vznikat z mnoha důvodů. Nejčastější oblastí vzniku nedorozumění jsou nedostatek informací a nerespektování zásad a postupů (ŠMÍDA, 2007, s. 32).

2.1.2 Projektový manažer

Manažera projektu můžeme charakterizovat jako plánovače, organizátora, kontrolora, koordinátora a vyjednávače projektových prací, který řídí pracovníky projektového týmu. Jeho hlavním úkolem je projektové práce řídit, nikoliv vykonávat. Měl by jim být někdo, kdo má s vedením ostatním lidí již dost zkušeností (DOLANSKÝ, MĚKOTA, NĚMEC, 1996, s. 58).

Pro úspěšné zvládnutí projektu musím mít projektový manažer tyto vlastnosti:

- Technické znalosti vztahující se k použití specifických znalostí, postupů a dovedností při jednotlivých úkonech, na úrovni zajišťující kontrolu, zaškolení podřízeného personálu a vedení diskuze v odborném duchu.
- Společenské a lidské faktory se schopností zabezpečit efektivní využití lidských zdrojů. Patří sem také efektivní řízení skupiny za účelem koordinovaného postupu.
- Koncepční schopnosti jsou nutné ke komplexnímu nadhledu na operace v organizaci jako celku, lze sem zařadit schopnost přijímat důležitá rozhodnutí (CEJTHAMR, DĚDINA, 2010, s. 31-32).

Úspěšný manažer projektu musí také zvládat několik velmi podstatných rolí. Kvalita zvládnutí těchto rolí má dopad na cely projekt a nedostatky v této oblasti mohou být indicii, že projekt nebude úspěšný. Role manažera projektu tedy jsou:

- vedoucí skupiny – formální i neformální autorita týmu,
- stratég – předcházení problémům na základě svých zkušeností,
- diagnostik – identifikace potencionálních problému dříve než nastanou,
- poradce – schopnost poradit v případě odborných problémů,
- filtr – zastavování negativních vnějších vlivů mířících na jeho tým,
- plánovač – příprava harmonogramu realizace projektu,
- kontrolor – kontrola provedení jím zadaných úkolů. (DOUCEK, 2004, s. 55-56)

Zjednodušeně je možné tvrdit, že manažer projektů musí každý den něco vyjednávat, sjednávat a dojednávat s cílem zabezpečení toho, co od ostatních potřebuje k dosažení stanovených cílů. Vyjednávací proces má dvě strany jako každá mince, kterou se platí za dosažení dohod. Na jedné straně jsou ti, co něco požadují, a na druhé ti, co to vyrobí nebo vymyslí. (DOLANSKÝ, MĚKOTA, NĚMEC, 1996, s. 61)

2.2 Vodopádová metoda

Vodopádový model životního cyklu projektu patří mezi modely nejstarší, jeho pojetí vychází de facto z tržní situace, kdy poptávka na trhu projekčních prací převyšuje nabídku a kdy vlastně každý projekt informačního systému se stává jedinečným dílem na zakázku. Tedy situace typická pro klasické období řízení projektů, které vyvrcholilo softwarovou krizí. Vliv na její vznik měl i způsob řízení projektů a jeho chápaní (DOUCEK, 2004, s. 69).

2.2.1 Životní cyklus vodopádového modelu

Předpoklady pro nasazení vodopádového modelu životního cyklu projektu jsou následující:

- jeho jednotlivé etapy jsou vykonávány postupně po ukončení etapy předchozí, jejíž výsledky se při ukončení etapy verifikují a slouží jako vstupy další etapy,
- nejlepšího výsledku při řešení projektu se dosáhne při uvedeném pořadí etap

- při žádném jiném pořadí vykonávání etap se nedosáhne lepšího výsledku (DOUCEK, 2004, s. 69-70).

Pořadí etap vodopádového životního cyklu projektu:

- předprojektová příprava
- analýza
- návrh
- implementace testování
- zkušební provoz
- zavádění do provozu
- rutinní provoz, údržba a rozvoj
- ukončení provozu (DOUCEK, 2004, s. 69-70).

Přestože se jedná o model, který se v praxi objevuje již dlouho dobu, nebyl zcela vyškrtnut ze seznamu používaných modelů. Pouze se změnil charakter jeho nasazování. Je možné tento přístup vidět u speciálních projektů na zakázku. Typický je tento postup u firem, které se snaží proniknout na trh s nějakým novým řešením. Tyto firmy, přestože mají vyškolené odborníky v oblasti vývojového prostředí, potřebují svou první zakázku, na níž by získané znalosti využily. Svým charakterem, jednoduchostí, srozumitelností, jednoduchou transformací jeho etap a fází do časové osy, se stal ideálním nástrojem pro sledování průběhu projektu jako celku. (DOUCEK, 2004, s. 71 - 72)

2.2.2 Ekonomické vazby

Ekonomické vazby mezi dodavatelem projektu informačního systému a jeho objednatelem lze shrnout do několika bodů. Objednatel projektu hradí veškeré finanční náklady na tvorbu projektu informačního systému – od etapy předprojektové analýzy až po jeho rutinní provoz a údržbu. Dodavatel dá k dispozici rámcovou metodiku, kterou naplňují konkrétní pracovníci na konkrétním projektu. Podíl na krytí nákladů na projekt a tím i míra rizika úspěšnosti projektu leží na bedrech objednatele. Dodavateli zůstává

pokrytí na vyškolení vlastních zaměstnanců a získání vlastních informačních technologií. (DOUCEK, 2004, s. 70)

2.3 Agilní přístupy

V případě, kdy nelze vytvořit přesnou specifikaci nebo když je jisté, že se průběh projektu bude pravidelně měnit, je vhodné zvolit agilní přístup k tomuto projektu. Mezi základní znaky projektu, který by měl využít agilní přístup lze zařadit tyto:

- Projekt je extrémně inovační a neexistuje spolehlivá báze pro jeho plánování.
- Zákazník (zadavatel) projektu je schopen dodat jen hrubou specifikaci, která sama o sobě neumožňuje odhadnout náklady a délku trvání projektu.
- Lze očekávat, že v projektu nastane v průběhu mnoho změn a bude se možná také měnit cíl projektu. Nezřídka kdy dojde k změně rozsahu i objemu práce. (Doležal, 2016, s. 307-308)

2.3.1 Vztah ke konvenčnímu projektovému řízení

Většina konvenčních projektů začíná tím, že sponzor, případně zákazník, definuje požadovanou změnu nebo cílový stav. Tento cíl zpracuje projektový tým, vytvoří odhady na dobu trvání a náklady. Tímto vznikne plán projektu. Všechno je tedy odvíjeno od specifikace požadavku. (Doležal, 2016, s. 310)

Velmi často se stává, že projekt nabírá zpoždění, čímž vzrůstají náklady. Z těchto praktik jasně plyne, že pro zadavatele je důležitější dodržení všech požadavků, než dodržení prvotně odhadovaných nákladů. V samé podstatě toto řešení není vždy špatné. Ovšem logickým důsledkem těchto postupů je, že rozsah je na prvním místě. (Doležal, 2016, s. 310)

Zato agilní projekt je velmi často započat pouze s hrubým obrysem o konkrétní podobě výsledného projektu. Požadavky bývají často doplňovány, případně se mění. Z tohoto

důvodu nelze předem říci, na čem bude řešitelský tým přesně pracovat. Agilní projekt proto musí již na svém startu mít přesně a již neměnně definované náklady a časový rozsah. Tento stav umožní řešitelskému týmu, aby zadavateli přinesl co největší přidanou hodnotu. I s tím rizikem, že kvůli změnám bude muset část hotové práce předělat či odstranit. (Doležal, 2016, s. 311)

Agilní metody kladnou hlavní důraz na zákazníka. Výsledný produkt (software) musí mít přidanou hodnotu pro zákazníka, tuto přidanou hodnotu definuje a určuje sám zákazník. Poté co zákazník určí všem úkolům přidanou hodnotu, následně se zpracovávají úkoly s nejvyšší přidanou hodnotou. (Jonák, 2015)

Základní agilní metodiky:

- Scrum
- SprintMethod
- Extrémní programování
- Lean Software Development
- Getting Real (Knesl, 2012)

2.3.2 Klíčové agilní principy

„Agilní projektové řízení bývá začasté spojováno s používáním některé ze zavedených metod, nicméně použití metody samo o sobě nevypovídá o tom, zda je projekt řízen agilně. Naopak, nemusí být adaptována žádná specifická metodika, abychom mohli prohlásit, že projekt je řízen agilně.“ Důležité je, aby byly uplatňovány následující principy: (Doležal, 2016, s. 311)

Inkrementální dodávky

Produkty projektu jsou dodávány postupně formou dílčích přírůstků. Projektový tým koncentruje své úsilí jen na malou část celkového rozsahu projektu s tím, aby se minimalizovalo množství rozdělané práce a naopak bylo možné sledovat postupné narůstání dokončených části projektu. Každý inkrement by měl představovat samostatně

funkční jednotku, která má přidanou hodnotu pro zákazníka projektu (např. fungující kalendář jako součást informačního systému, zřízená a zprovozněná kuchyňka při stěhování kanceláří apod.). (Doležal, 2016, s. 311)

Iterativní postup

Práce na projektu je rozfázována do etap, které jsou charakteristické shodnou délkou. Každá iterace je tedy milník. Obvykle je výhodné, když jsou iterace stejně velké. Předpokladem pro iterativní postup je dodání požadavků ve velikosti, která se dá za jednu iteraci stihnout. Preferována je kratší perioda, typicky malé jednotky týdnů. (Knesl, 2016)

Další principy logicky navazují nebo podporují předcházející dva:

Multifunkční týmy

Z důvodu omezeného času na dodání dílčích výstupu je potřeba velmi úzké spolupráce mezi jednotlivými členy týmu. Proto se například testeři zapojují do práce ihned na začátku programování softwaru. Začínají ihned psát testovací scénáře a testují první funkční části a prototypy, a ne jako při řízení vodopádem, kdy dostávají k testování až celou naprogramovanou aplikaci. (Doležal, 2016, s. 312)

Zapojení zákazníka/byznysu

Vzhledem k častým změnám a úpravám projektu je nutné, aby zákazník byl pravidelně k dispozici týmu k upřesňování či měnění zadání. Navíc zákazník poskytuje týmu zpětnou vazbu na již odvedenou práci a je možné ihned na jeho podněty reagovat. V agilním přístupu nelze jít cestou, kdy zákazník na počátku předloží zadání a za několik měsíců si nechá předvést hotový projekt. Zákazník se aktivně podílí na tvorbě a koriguje veškeré své požadavky. (Doležal, 2016, s. 312)

Pravidelná revize požadavků (scope)

Agilní projekty začínají s hrubou představou konečné podoby projektu. Zároveň jsou očekávány velké změny případně další upřesnění. K úspěšnému splnění projektů je potřeba pravidelně kontrolovat a seřazovat další požadavky, které vedou k dosažení cíle projektu. (Doležal, 2016, s. 312)

Agilní chování

Od každého člena týmu se předpokládá, že se bude sám aktivně zapojovat, rozebírat si úkoly, snažit se vše dodat ve slíbený čas a kvalitě. Toto chování vyžaduje jisté zkušenosti s fungováním v tomto prostředí. (Doležal, 2016, s. 312-313)

2.4 Metoda Scrum

„Nejpoužívanější metodikou agilního řízení je v dnešní době jednoznačně Scrum. Jedná se o přizpůsobivou metodu produktového vývoje, zvláště přínosnou při inkrementálním a setrvalém dodávání inovací.“ (Takeuchi, a další, 1986, cit. podle Doležal, 2016, s. 314)

„Termín Scrum se používá v souvislosti s agilním projektovým řízením od devadesátých let, kdy v roce 1993 proběhl první zdokumentovaný projekt řízený touto metodikou. Samotný název Scrum v sobě skrývá metaforu s hrou rugby, kde scrum (zkratka ze „scrumage“, česky „mlýn“) představuje herní situaci, kdy se shromáždí celý tým a společně se ve mlýně snaží získat/udržet míč.“ (Doležal, 2016, s. 314)

Ve scrumu se vše podřizuje tomu, aby úkol zadaný na začátku sprintu byl na konci sprintu celý hotový. Vzhledem k práci v iteracích (sprintech) je možné velmi přesně odhadnout náročnost práce. Díky speciální roli produktového vlastníka je zajištěno, že zadání píše přímo zákazník. Bere mu to minimum práce a je aktivním účastníkem procesů. (Knesl, [b.r.])

2.4.1 Základní přehled metodiky Scrum

Scrum využívá iterace nazývané sprint. Tyto sprinty mají přesně danou délku a pravidelně se opakují. Nejčastěji se jedná o dva týdny, maximální délka může být až jeden měsíc. Na konci každého jednoho sprintu musí být hotov produkt, tedy častěji jedna jeho funkční část. Tuto část musí být možné ukázat zákazníkovi či komukoliv jinému. Jestliže by vývoj jedné části trval déle než jeden sprint, nelze jej do sprintu zapojit. V takovém případě je nutné úkol rozdělit na menší části, které lze v jednom sprintu splnit. Toto opatření umožňuje lepé zvládat riziko a také zpřesňuje odhady trvání a náročnosti práce. Na začátku každého projektu dochází ke sběru požadavků na výsledný produkt. Tyto požadavky projdou prioritizací, čímž vznikne „product backlog“. Tímto vznikne seznam všech veškerých požadavků na projekt a navíc jsou tyto požadavky seřazeny podle důležitosti. Z tohoto seznamu jsou postupně vybírány úkoly do „sprint backlogu“. Zde jsou shromažďovaný a evidovány úkoly pro nadcházející jeden sprint. Během sprintu se celý tým pravidelně schází na operativních poradách (daily stand-up). Na této schůzce všichni členové týmu vyjádří progres ve své práci, v případě problému či nejasnosti lze ihned reagovat. (Doležal, 2016, s. 314-315)

2.4.2 Scrum role

Metoda scrum rozeznává několik základních rolí, které musí být obsazeny, aby vše fungovalo jak má a projekt byl veden do úspěšného cíle. (Doležal, 2016, s. 315)

Vlastník produktu:

- Musí se vždy jednat jen o jednu osobu
- Nese zodpovědnost za to, že projekt maximalizuje návratnosti investic. Snaží se toho docílit tím, že nechá tým pracovat nad částí projektu, která má nejvyšší přidanou hodnotu.
- Jedná se o vlastníka produkt backlogu, tudíž určuje prioritu jednotlivým úkolům.
- Poskytuje vývojovému týmu přesnou specifikací produktu. Komunikuje s třetími stranami, které budou projekt využívat či se na něm podílí.

- S konečnou platností rozhoduje o tom, zda je splněn úkol podle zadání a vizí či nikoliv. To znamená, že akceptuje/odmítá hotový produkt či jeho část. (Doležal, 2016, s. 315)

Projektový tým:

- Jedná se o skupinu přibližně sedmi spolupracujících lidí. Jestliže je v projektu zapojeno více lidí je potřeba je rozdělit do více týmu právě v počtu přibližně sedmi lidí. Tento počet zachovává přínosy osobní interakce a tým je lépe řiditelný.
- Multifunkčnost – složen z lidí s různou znalostí a zaměřením, aby tým mohl dodávat úkol jako celek a byl samostatný. Důležité je sestavení tak, aby nevznikaly prostoje v práci.
- V ideálním případě by tým měl sdílet pracoviště. Není to nutná podmínka, ovšem umožnuje rychlejší jednání a zvyšuje efektivitu.
- Tým jedná jako celek, čili jako celek bere na sebe závazek, že práci, kterou řekne, že dodá, v daném sprintu opravdu zvládne. (Doležal, 2016, s. 315-316)

Scrum master

- Usnadňuje vytvoření co nejvhodnějšího pracovního prostředí.
- Zodpovědný za průběh veškerých scrum procesů. To znamená, že organizuje porady, moderuje týmové porady, kontroluje dodržování termínů, provádí údržbu scrum nástrojů (například nástěnky).
- Odstraňuje veškeré překážky a rušivé elementy, aby se tým mohl soustředit pouze na svou práci.
- Nemá rozhodovací pravomoc
- Může pomáhat vlastníkovi produktu s některými jeho úkony
- Jedná se o nepovinnou roli. U zkušených a zaběhlých týmu není už potřeba. (Doležal, 2016, s. 316)

2.4.3 Scrum postupy a porady

Scrum si zakládá na jednání tváři v tvář. Z toho důvodů se většiny schůzek má účastnit celý tým, aby každý věděl, co se bude dít a zároveň může každý vyslovit svůj názor. Každý sprint musí obsahovat několik chronologicky po sobě jdoucích jednáních. Některé jsou jen jednou, jiné se opakují každý den. (Doležal, 2016, s. 316)

Plánování sprintu:

- Cíl porady: Stanovit úkoly pro nadcházející sprint a jejich podobu zafixovat
- Trvání: Maximálně 8 hodin pro sprinty trvající 30 dní. Pro čtrnáctidenní sprint nejdéle 3 hodiny.
- Účastnící: Vlastník produktu, projektový tým, případně scrum master

Plánování sprintu má za úkol, aby analytik a vlastník produktu představili své user story a projektový tým se zaváže k dodání konkrétní části produktu. Projektový tým nemusí vzít vše, co vlastník produktu představí, pokud si myslí, že mu na vše nebudou stačit jeho kapacity. Vlastník produktu zodpovídá za to, že celý produkt backlog je seřazen podle priorit. To znamená, že nahoře jsou všechny úkoly, které přináší projektu největší užitek. (Doležal, 2016, s. 316)

Důležitým pravidlem je to, že se odsouhlasený rozsah za celý sprint nezmění. V případě, že úkol byl zadán chybně, je nutné jeho opravu zařadit opět do backlogu na horní pozice, aby v dalším sprintu došlo k přepracování. Neměnnost úkolu je důležitá z důvodů ochrany projektových týmů. Je nutné, aby týmy nebyly přetěžovány a neměnilo se jim zadání v době, kdy již na něm pracují. Změna zadání by mohla způsobit, že předělány úkol se už nestihne včas udělat a navíc může vyčerpat čas i pro práci na dalších úkolech ve sprintu. Důsledkem tohoto jednání by se stalo, že tým nesplní ve sprintu to, co se zavázal dodat. (Doležal, 2016, s. 316-317)

Denní stand-up

- Cíl: Sledování postupu prací, stanovení pracovního plánu na daný den, identifikování aktuálních a předpokládaných překážek.

- Trvání: Maximálně 15 minut.
- Účastníci: Projektový tým, scrum master, případně vlastník projektu

Už podle názvu je jasné, že se jedná o každodenní schůzku. Každý projektový tým tuto schůzku pořádá ráno před začátkem prací. Účelem této schůzky je sdílet informace o tom, co se za předchozí den stihlo udělat, dále každý sdělí plán jeho prací na daný den a následně je možnost sdělit překážky na které bylo naraženo, případně včas odhalit překážky, které by se daný den mohly vyskytnout. Zde se jednotliví členové týmu dovědí, co dělají ostatní a navíc je možné předat pomyslný štafetový kolík dalšímu jedinci v případě, že na jednom úkolu pracuje více lidí a jsou na sobě navzájem závislí. Každý člen projektového týmu odpovídá na tři otázky:

1. Co mám hotovo?
2. Co udělám dnes?
3. Vidím nějaké překážky?

Důležité je, že denní stand-up neslouží k řešitelským poradám nad vzniklými problémy. Pouze se na této schůzce určí lidé, kteří se budou řešením zabývat a také čas a místo, kde daný problém budou řešit. V případě, že by se denní stand-up proměnil v řešitelskou poradu, ztrácí význam. (Doležal, 2016, s. 317)

V případě práce na náročnějších projektech, na kterých pracuje najednou více týmu, následuje ještě schůzka zástupců všech týmů. Průběh je obdobný jako u stand-up schůzky, jen s tím rozdílem, že se řeší pouze úkoly, které mohou ovlivnit ostatní týmy, případně na ně mají ostatní týmy navázat. (Zikmund, 2010)

Sprint review

- Cíl: Předat a akceptovat hotový produkt nabraný do sprintu
- Trvání: Velice individuální podle složitosti úkolu, ne však delší než 3 hodiny
- Účastníci: Projektový tým, vlastník produktu, případně další zainteresované strany

Hlavní podstatou sprint review je předložení a předání hotového projektu, který se tým zavázal za jeden sprint dodat. Projektový tým předvádí, co vytvořil a vlastník produktu rozhodne o akceptaci či neakceptaci úkolu. Žádný úkol nelze akceptovat s jakýmikoliv výhradami. To znamená, že úkol je buď splněn na 100%, nebo vůbec. V případě, že úkol nebyl akceptován a do konce sprintu se jej nepodaří opravit a znovu akceptovat, je zařazen zpět do backlogu a vlastník produktu opět tomuto úkolu přidává znovu prioritu. S velkou pravděpodobnosti je úkolu daná jedna z nejvyšších priorit a tým si jej znovu nabere do dalšího sprintu a dokončí jej. (Doležal, 2016, s. 318)

Retrospektiva

- Cíl: Sestavení zpětné vazby za uplynulý sprint a poučení se z ní
- Trvání: Maximálně 4 hodiny
- Účastníci: Vlastník produktu, projektový tým, scrum master

Poslední porada právě končicího sprintu. Účelem této porady je ohlédnutí se za uplynulým sprintem a zhodnocení věcí, které se povedly či naopak nepovedly. V rámci retrospektivy tým může hodnotit úplně vše, co se za dany sprint událo. Výsledkem je seznam věcí, které týmu fungují a má se jich držet i seznam neúspěšných věcí s návrhem, jak jím příště lépe čelit či jak se jim zcela vyvarovat. Závěry tedy lze jednoduše rozdělit mezi čtyři skupiny výsledků:

- Co ponechat stejné - jde o činnosti nebo věci, které správně fungují a týmu vyhovují, není důvod tyto úkony nepoužívat dále.
- Co změnit - činnosti, jejichž princip je správný, jen postoj nebo postup nebyl dokonalý. V tom případě činnost zachováme, jen změníme přístup k ní.
- Co zrušit - veškeré skutečnosti, které působily negativně a měly neblahý vliv na průběh sprintu a fungování týmu.
- Co zavést – jestliže jsme se v průběhu sprintu objevili novou činnost, nebo se o ní dozvěděli, a je-li tým přesvědčen, že mu tato činnost přinese užitek, pak lze tuto činnost zakomponovat do dalšího sprintu a využít její přínosy. Může se jednat o nové principy, nové technologie, nové procesy či nové postupu. (Myslín, 2016, s. 117-118)

Nástroje scrumu

Jedná se o několik nástrojů, které ulehčují přehled a řízení scrumu. Použití těchto nástrojů je velmi podstatné pro správné fungování celého projektu. (Doležal, 2016, s. 318)

Produktový backlog

Produktový backlog je souhrn všech informací vztahujících se k dodávanému produktu. Jedná se o hlavní a nenahraditelný seznam informací v agilním vývoji. Jednotlivé položky produktového backlogu tvoří jako celek celý projekt. Jedná se o seznam všech úkolů, které je zapotřebí udělat, aby výsledný produkt fungoval jako celek. Produktový backlog je přístupný všem, ovšem jen vlastník produktu, případně jim pověření lidé mohou přidělovat jednotlivým úkolům prioritu. K určení priority se používají tři hlavní zásady:

- Nejhodnotnější věci jako první. Vzhledem k tomu, že agilní projekt má jako svůj imperativ dodání co největší hodnoty, je nutné právě tyto nejhodnotnější věci zařadit na špici produktového backlogu. Protože čím dřív budou dodány, tím dřív mohou být využity.
- Nejrizikovější věci jako první. Jestliže existuje pochybnost, že z důvodů nějakého úkolu má projekt skončit nezdarem. Ať už z důvodů vysoké technické náročnosti (nad možnosti projektového týmu), případně složitého až nemožného zavedení do praxe, je nutné, aby úkoly které mohou tento nezdar způsobit, byly zařazeny na špici produktového backlogu. Nejlepší je proto zjistit, že projekt nemůže být úspěšný již na jeho začátku.
- Nejrychleji vyrobitelné věci jako první. Jestliže všechny komponenty mají pro zákazníka stejný přínos, je proto vhodné odbavit na začátku co nejvíce věcí, aby přinášely co nejvyšší užitek. (Doležal, 2016, s. 318-319)

Sprintový backlog

Obdobný jako produktový backlog, jen s tím rozdílem, že zobrazuje pouze věci, které se tým zavázal v daném sprintu dodat. (Doležal, 2016, s. 318-319)

User story

Jedná se o jednotlivé položky, které tvoří backlog. Jedná se o funkční specifikaci, která obsahuje tyto informace:

- Kdo je příjemcem přidané hodnoty
- Co konkrétně má produkt dělat
- Jaké očekávání je tím plněno

Jednotlivé user story musí splňovat základní charakteristiku:

- Unikátní: Důležité je, aby každá user story řešila právě jedno očekávání. Ovšem z pohledu výsledného produktu. Proto na jedné user story může pracovat více lidí odpovědných za různé technologie.
- Přínosná: Není dobře, aby tým pracoval na úkolu, který nemá pro zákazníka žádný přínos. V tom případě nemá smysl úkol plnit. Není rozdíl v tom, když je či není vyhotoven.
- Odhadnutelná: U každé user story musí být možné odhadnout kolik, zabere práce a navíc množství práce nesmí přesáhnout dobu trvání jednoho sprintu. V případě, že odhad přesáhl dobu trvání jednoho sprintu, je nutné tuto user story rozdělit na dvě menší.
- Testovatelná: Musí se jednat o úkol, který lze řádně otestovat, aby byla zajištěna kontrola jeho kvality. (Doležal, 2016, s. 319-320)

Scrum board

Všechny user story se dále rozpadají na jednotlivé úkoly. Tyto úkoly jsou ve své podstatě zajímavé pouze pro projektový tým, který zajímá, co vše a v jakých technologiích je potřeba udělat pro dokončení user story. Nejjednodušší forma je scrum board, kde jsou každé user story přiděleny veškeré podúkoly. U těchto podúkolů je možné sledovat, zda již jsou hotové, nebo se na nich právě teď pracuje či stále čekají na zpracování. Právě tyto podúkoly řeší denní stand-up. (Doležal, 2016, s. 320-321)

Plánovací poker

K hodnocení náročnosti jednotlivých user story lze použít techniku plánovacího pokeru. Každý člen týmu obdrží balíček karet s čísly, tyto čísla znamenají náročnost daného úkolu. Každou user story někdo znalý představí. Představení by mělo obsahovat informace o tom, co je potřeba v dané user story udělat, za jakým účelem a jaké budou potřeba znalosti k jejímu dokončení. V případě nejasností může proběhnout krátká diskuze. Poté se odehraje samotný plánovací poker. Každý účastník schůzky odhalí kartu, na které má číslo, které značí náročnost daného úkolu. Tento úkon musí provést všichni najednou. Ve chvíli kdy má někdo velmi vysokou nebo nízkou hodnotu vůči ostatním, je vyzván, aby své rozhodnutí vysvětlil. Po ukončení této diskuze se zapíše k dané user story hodnota, na které se celý tým následně shodl. (Doležal, 2016, s. 323)

3 ANALYTICKÁ ČÁST

Analytickou část budu zpracovávat o vybrané společnosti. Tato společnost působí na českém trhu více než dvacet let. Hlavní činností společnosti je návrh a vývoj nového finančního softwaru. Vybraná společnost spolupracuje výhradně s jednou značkou, které působí napříč po celém světě. Pro svého zákazníka software pouze nevyrábí, ale také provozuje veškeré servery, na kterých systém běží a stará se také o jeho neustálé monitorování a spravování. Po celém světě má vybraná společnost až dvacet datových center.

Do společnosti docházím už více než dva roky. V posledním roce mi bylo dovoleno intenzivněji sledovat své nadřízené a kolegy za účelem pochopení jejich práce. Získal jsem detailní přehled o postavení jednotlivých roli v týmu, který využívá vodopádové projektové řízení.

Vybraná společnost je rozdělena do několika pracovních týmů. Jedná se o týmy zajišťující provoz serveru pro zákazníka, hardwarovou a softwarovou podporu zaměstnanců, tým zajišťující bezpečnost, vnitřní fungování firmy, lidské zdroje a hlavně týmy vyvíjející software. Tyto týmy jsou rozděleny podle toho, do které světové země pro zákazníka dodávají.

3.1 Práce týmu

Dostal jsem se do vývojového týmu. Jeho hlavním úkolem je vytvoření nových části softwaru, který bude spojen s již používaným softwarem. Zákazník si tedy pravidelně chystá své poznatky a vymýšlí nové věci, které chce nabídnout svým zákazníkům. Aby jim nové služby mohl nabídnout, potřebuje k tomu často novou část softwaru. S tímto se obrací na vybranou společnost – konkrétně na vývojový tým.

Vzhledem k tomu, že je využíváno řízení projektu pomocí vodopádů, tak doba od úmyslu až po finální nasazení trvá opravdu dlouho. Jeden release trvá nejméně osm měsíců.

3.2 Průběh jednoho release

Úplně na začátku všeho je nápad zákazníka, který chce buď nějakou novinku, nebo potřebuje vylepšit stávající funkčnost. Zákazník sepisuje své poznámky do ucelené formy, kterou pak vybrané společnosti představuje a řeší, zda je možné jeho představy naplnit.

Následně se zákazník setkává s jedním či více analytiky ze společnosti. Jejich společných úsilím je navrhnout, jak daný požadavek má fungovat a vypadat. Jakmile jsou tyto procedury ukončeny, je úkolem analytika vypracovat zprávu, podle které budou moci vývojáři dané řešení napsat a nakonfigurovat.

Ve chvíli, kdy jsou vývojáři se zadanou práci hotoví, začíná se nový software testovat. Testování probíhá na zkušebních testovacích prostředích, kde musí každý tester zkoušet mnoho kombinací, které systém umožňuje a ověřit, že se vše chová jak má.

Jestliže je vše důkladně otestované může se daný release uvolnit na produkci. Toto je nejdůležitější část projektu, všichni netrpělivě očekávají, jestli vše co prováděli, bude fungovat tak jak má a zákazník bude spokojen.

3.2.1 Analýza

Analýzu provádí analytik společně se zákazníkem. Účelem jejich pravidelného setkávání je pochopení veškerých požadavku zákazníka na novou část softwaru nebo vylepšení či doplnění stávající funkčnosti.

Když má analytik veškeré podklady od zákazníka, začíná analyzovat proveditelnost těchto změn v systému. Dohledává veškeré dopady na stávající funkčnost a zjišťuje, zda je možné změny provést či nikoliv. Velmi často se stane, že analytik objeví dopady do systému, které předtím jeho ani zákazníka nenapadly a opět se vrací k jednání, jakým způsobem projekt upravit, aby byl proveditelný a zároveň, aby byly zachovány veškeré požadavky zákazníka.

Veškerá analytická práce musí být provedena poctivě a kvalitně. Předejde se tak budoucím velkým problémům a dlouhým časovým prodlevám. Tyto časové prodlevy mají za následek snížení spokojenosti zákazníka, vyšší časovou zátěž na zaměstnance, což vede k jejich přetěžování a v neposlední řadě snížení výše platby za dodaný software. Navíc časový nátlak sebou přináší také vyšší chybovost.

Ve vybrané společnosti působí několik analytiků a v rámci jednoho release řeší každý z nich i více úkolů. Hledání řešení provádí s více než jedním oddělením u zákazníka a zde často dochází k velkým průtahům. Každé oddělení u zákazníka má trošku jinou představu o budoucím fungování softwaru a je na umu analytika, aby všechny tyto požadavky spojil do všemi přijatelného kompromisu.

3.2.2 Návrh řešení

Návrh se provádí sepisováním dokumentace. Tuto dokumentaci sepisuje analytik a obsahuje veškerou technickou dokumentaci. Účelem je sepsání veškerých změn a způsob jejich provedení a navázání na stávající funkčnost. Návrh obsahuje veškeré informace, které jsou potřeba pro kvalitní provedení vývoje, umožňuje předání informací vývojářům, ale také zajištění veškeré technické dokumentace do minulosti. Díky těmto informacím lze dohledat i starší změny, díky čemuž i noví zaměstnanci vědí co, jak a proč funguje.

Návrhové dokumentace jsou velmi důležité pro budoucí fungování společnosti. Za pomocí starší návrhové dokumentace lze navázat na starší práci i u lidí, kteří ve společnosti nepůsobí. Je to taková sběrnice systémového know-how.

Najdou-li se nesrovnalosti během navrhování řešení, je možné se opět vrátit do analytické části, kde jsou na tyto otázky hledány odpovědi. Opětovně proběhne analýza a pak se přestoupí znovu k návrhu. Tyto časové prodlevy mají nepříjemné dopady na časový harmonogram celého projektu.

Vytvoření správného návrhu je velmi zdlouhavá a složitá práce. Analytik nesmí opomenut žádný detail, veškeré informace musí být logický seřazeny a musí být zajištěno jejich propojení. Důležité je zaznamenat veškeré podklady pro rychlé a úspěšné vyvinutí vývojáři, kteří potřebují informace o validacích softwaru, rozhraní, napojení a mnoho dalších.

3.2.3 Vývoj

V týmu na který se zaměřuji, působí několik desítek vývojářů, kteří ovládají různé programovací jazyky. Každý vývojář se specializuje na jednu část softwaru, to ovšem neznamená, že nedokáže pracovat i na jiné časti, jen mu tato práce trvá trošku déle, než si zjistí veškeré již fungující souvislosti a pochopí skladbu kódu dané části systému.

Vývojáři pracuji na pro ně připraveném vývojovém prostředí, kde podle dokumentace od analytika zpracovávají zadané požadavky. Práce je to velmi složitá a náročná, jelikož požadavky jsou předávány jako velké celky. Práce na těchto celcích může trvat i více než dvacet dní a během této doby nikdo nekontroluje, zda se jde správnou cestou a je pochopeno zadání. Tato kontrola ani není možná, protože když probíhá vývoj podle dokumentace, takže viditelné a testovatelné výsledky se dostaví, až když je vše hotové.

Vývojáři pracuji na vývojovém prostředí. To je uměle vytvořené prostředí pro jejich potřeby, obsahuje veškerý software, který je využíván v ostrém provozu, jen kvůli zátěži obsahuje méně dat a tato data jsou z důvodů bezpečnosti anonymizována, čímž se snižuje riziko na vynášení informací.

Průběh vývoje se neliší od jiných softwarových firem. Každý vývojář si vyvíjí u sebe na lokále, a jakmile má vše hotové a obsahuje to nějakou celou funkční část, tak vše přenese na vývojové prostředí. To je první místo, kde může nastat první velký konflikt s jiným buď novým, nebo již existujícím zdrojovým kódem. Zde je to pak na šikovnosti vývojářů, aby určili důvod konfliktu, jestli je chyba na jejich straně a mají špatně napsaný kód nebo je chyba už na architektuře, kterou navrhl analytik a podle které vývojář pracoval.

Dalším velkým problémem je spolupráce více vývojářů na jednom společném úkolu, ale zároveň každý pracuje ještě na dalších jiných úkolech buď sám, nebo případně s jinými kolegy. Zde dochází k velkým problémům při komunikaci a prioritizaci, není lehké vše koordinovat ke spokojenosti všech.

Jestliže během vývoje přijde vývojář na nesrovnalosti v dokumentaci, musí ihned informovat analytika, aby danou část prošetřil a co nejdříve navrhnul nové funkční řešení. Tyto časové prodlevy jsou velmi nepříjemné a celý projekt zpomalují.

Během vývoje nezahálejí ani software testeři, kteří se také věnují studování návrhové dokumentace a na základě informací z této dokumentace sepisují prvotní testovací scénáře. Tyto scénáře jsou využity, když se nově vyvinutý software dostane do fáze testování. Tyto scénáře ukazují princip fungování nového softwaru.

3.2.4 Testování

Testovat se začíná, jakmile jsou všechny úkoly vyvinuty a nasazeny na testovací prostředí. V tuto chvíli software testeři využijí své dříve napsané scénáře jak k nové funkčnosti, tak k stávající funkčnosti.

Testování začíná základními testy, které obsahují nejdůležitější funkčnost a procesy jejího běžného použití. Tyto scénáře jsou otestovány co nejdříve, aby byly nalezeny ty

nejzávažnější chyby, které mohou blokovat velký kus systému, ten pak není možné testovat a musí se čekat na opravu.

Ve chvíli kdy jsou otestovány nejdůležitější a nejzásadnější části systému přechází se k regresnímu testování. Účelem regresního testování je protestovat celou aplikaci, a to nejen na to zda fungují hlavní procesy, ale že fungují všechny procesy a jednotlivé části systému spolu spolupracují tak jak mají a z jejich práce vznikají kvalitní validní data. Vybraná společnost má momentálně na sledovaném oddělení přibližně 4 500 – 5 000 těchto scénářů. Důležité je opravdu protestovat každý jeden tento scénář, jelikož jen tak je zajištěno, že zákazník dostane kvalitní a správně fungující produkt.

Z důvodů, že každý release je velmi rozsáhly a obsahuje mnoho nového softwaru a tím pádem i mnoho chyb, nestačí regresní test udělat jen jednou. Proto se regresní testování provádí na čtyři nebo dokonce pět kol. Velký problém vidím ve fázi, kdy jsou nasazeny opravy reportovaných chyb a velká část softwaru už je otestována, jelikož může dojít k tomu, že oprava chyby zanese chybu do již otestované a funkční části. To je taky důvod proč se děla tolik kol regresního testování. Přesto není možné docílit, aby byla aplikace nasazena zcela bez chyb. Stavu kdy aplikace bude zcela bez chyb lze jistě docílit, ale z finančního hlediska by to bylo natolik náročné, že je lepší počítat s tím, že v aplikaci asi nějaké chyby budou a ty operativně řešit.

Když tester nalezne při testování chybu v softwaru, musí ji zapsat do aplikace na řízení projektů, která navíc podporuje správu chyb a jejich sledování. Tester musí podrobně popsat, jaké kroky vedly k nasimulování chyby, dále je vhodné dodat obrázky na kterých je daná chyba přesně vidět. Dalšími důležitými parametry jsou prostředí, kde byla chyba simulována a u webových aplikací taky čas, díky kterému si vývojář může z logu přečíst informace k chybě. Ve vybrané společnosti někteří zkušení software testeři jsou schopni informace k chybě v logu najít a přiložit k chybě. Tím nejenže ulehčují práci vývojářů, kteří to dokážou velmi ocenit, ale navíc tím zkracují dobu, za níž bude chyba opravena, protože vývojář se může plně soustředit na opravu a nemusí se ničím jiným zdržovat.

Nástroj na vedení chyb umožňuje navíc zadávat i závažnost chyby. Díky tomu mají všichni přehled, kde v systému jsou jak závažné chyby. Tato prioritizace umožňuje filtrování závažných chyb a jejich předběžné řešení. Chyby s vysokou prioritou většinou blokují větší část systému a neumožňují testováni mnoha dalších scénářů, naopak chyby s nízkou prioritou jsou často jen zobrazovací chyby. Tyto chyby neblokují systém a není nutné je odstranit okamžitě, ale i tyto chyby musí být opraveny, ovšem třeba v pozdější fázi testů.

Společnost používá při reportování chyb čtyři úrovně závažnosti chyb. Nástroji na řízení projektů má projektový manažer možnost si chyby podle závažnosti filtrovat a má díky tomu poměrně jasný přehled, kolik chyb projekt nyní brzdí a kolik chyb už bylo vyřešeno. Úrovně závažnosti jsou tyto:

- low (nízká – jedná se většinou o chybu, kde je překlep v nějakém textu, chyba nemá vliv na další testování),
- medium (střední – jedná se chybu, která například zamezí jen zobrazení jedné finální části testovaného procesu),
- high (vysoká – chyba která neumožňuje pokračovat v mnoha dalších scénářích či procesech, většinou se jedná o chyby, které znemožňují vyhledávání či vrací zcela nesmyslný a nesprávný výsledek),
- critical (kritická – velmi závažná chyba, která neumožňuje například možnost přihlásit se do aplikace).

Chyby úrovně high a critical musí být řešeny dříve, než chyby úrovně medium a low, jelikož tyto chyby blokují velkou část systému a znemožňují plynulé testování.

K tomu, aby software testeři dobře věděli o tom, které scénáře již byly v aktuálním kole regresního testování otestovány, slouží katalog všech scénářů v Excelu. Tento katalog obsahuje hlavičky všech scénářů a je rozdělen podle oblastí. Katalog umožňuje poměrně přesný přehled o tom, kolik už toho je otestováno a kolik toho ještě chybí. Díky správnému nastavení filtru je možné v jedné záložce také sledovat množství

scénářů, které blokují chyby. K tomuto informovaní se využívají informace, které lze ke každému scénáři lze z combo boxu zadat:

- not started (primární přiřazení, které značí, že test ještě nebyl ani začat),
- in progress (tento stav se zadává u scénářů, u kterých dochází k dlouhému čekaní – například je-li nutno čekat na proběhnutí nočního jobu a scénář je vyhodnocen až druhý den),
- not stopping bug (zadává se u scénáře, kde byla nalezena a reportována chyba, ale lze přesto testovací scénář dokončit),
- showstopper (příznak chyby, která blokuje dokončení scénáře),
- blocked (stav, kdy scénář nelze otestovat z důvodu, že potřebuje splnění podmínky jiným scénářem a ten nelze splnit kvůli chybě),
- tested correctly (ideální situace, scénář otestován a vše funguje jak má).

Veškeré tyto stavy v katalogu zadává tester, který navíc ndo dalšího sloupce doplní číslo chyby, z důvodů jednoduššího dohledávání a přetestovaní. Jakmile tester provede jakoukoliv změnu stavu, musí také do příslušného sloupce doplnit své jméno, aby bylo jasné, kdo test prováděl a je za něj zodpovědný.

3.2.5 Oprava chyb

Vývojáři opravují chyby na základě podnětu od testerů, tento podnět přichází mailem ze systému na vedení chyb, ve chvíli kdy tester přepne chybu k řešení a přiřadí k ní daného vývojáře. Jak již bylo zmíněno, chyby musí být řešeny dle priorit, čím vyšší priorita, tím dřív musí být chyba opravena. Ve chvíli, kdy dojde k opravě, musí vývojář kontaktovat jiného kolegu, aby mu udělal kontrolu správnosti kódu ve chvíli, kdy je nový kód schválen, dochází k jeho nasazení na testovací databázi. Vývojáři mohou po celý den skládat své požadavky do fronty, následně každou noc probíhá automatické nasazení nových věcí na testovací databázi.

3.2.6 Opakované testování chyb

Každé ráno si vývojáři, kteří den předtím opravili nějaké chyby, zkontrolují, zda opravdu došlo k nasazení jejich opravy na testovací databázi a pokud vše proběhlo jak mělo, tak přepnou chyby v systému zpět k testům na testera, který chybu zadal. Úkolem toho testera je daný problém otestovat podle postupu, který v chybě uvedl. Jestliže tester zjistí, že chyba stále přetrvává, vrací chybu zpět na vývojáře, aby ji opravil, a proces opravy se opakuje. Jestliže chyba byla odstraněna, tak tester chybu v aplikaci uzavře a přetestuje testovací scénář, který k chybě vedl. Po úspěšném dokončení testů je opět nutné výsledek zapsat do katalogu, kde dojde k nahrazení sloupce showstopper nebo not stopping bug za hodnotu tested correctly.

3.2.7 Nasazení

Jedná se poslední fázi vývoje a dodání systému, dochází k uvolnění veškeré funkčnosti, jak nové, tak stávající na produkční servery zákazníka. Tyto servery sice spravuje vybraná společnost, ale data patří zákazníkovi. Nasazení probíhá zpravidla v noci, protože dochází k odpojení veškerých systémů. Noční hodiny snižují počet zasažených lidí na minimum, jelikož po dobu nasazení nefungují ani věci, které zákazník nabízí svým koncovým uživatelům. Nasazení probíhá následovně:

- zálohováni a uložení koncového bodu databáze,
- odpojení stávajícího systému,
- nasazení systému i s nově dodanými věcmi,
- základní testování nově nasazeného systému,
- odstranění dat způsobených testováním.

Nasazení je velkým krokem v rámci celého projektu, ovšem není to poslední krok projektu. I po nasazení je nutné vše důkladně sledovat a případně rychle řešit chyby, které se mohou vyskytnout. Aplikační podporu vybraná společnost poskytuje po celý rok, každý den, ovšem po nasazení je vždy do podpory zapojeno více lidí.

3.3 Zákaznické akceptační testy

Jakmile sestava testerů dokončuje regresní testování, je nový systém nasazen na testovací prostředí u zákazníka. Zákazník za pomocí svých testerů zkouší své procesy a kontroluje, že vše funguje jak má, a také vše funguje podle toho, jak si to začátku přál. Vzhledem k tomu, že dochází k nasazení velkého množství nových dat, předpokládá se, že i zákazníkovi testeři najdou nějakou chybu, není to nic zvláštního, je to logický jev. Zákazníkovi je následně umožněno reportovat chyby podobným způsobem, jako u stálých zaměstnanců. Oprava je prováděna stejně, jako na vlastních testovacích prostředích, jen s větším důrazem na rychlost – přece jen je to pro zákazníka. Podstatné je, aby byl zákazník spokojen s dodaným softwarem a vše fungovalo jak má. Ve chvíli, kdy zákazník přijde na to, že nějaká část funguje jinak, než si na začátku myslel, dochází k velkým problémům. Tyto problémy sebou často nesou přepracovávání velkého množství zdrojového kódu, na které následně navazuje opět mnoho testování, z tohoto důvodu poté často dochází k oddálení nasazení a prodloužení termínu.

Vzhledem k tomu, že zákazníkovi testeři neznají pozadí aplikací natolik dopodrobna, často dochází k tomu, že zadávají chyby, které chybami nejsou. Velmi často jsou tyto „chyby“ způsoby špatným nastavením používaných metod a procesů. Tyto metody mají většinou svou správu ve zcela jiné části aplikace a vzhledem k tomu, že v rámci testování se tyto metody mění, může dojít ke změnám v jiné části, kde to jiný tester nečeká a vyhodnotí výsledek jako chybu.

3.4 Chyby po nasazení

Kdyby mělo dojít k nasazení bez žádných chyb, muselo by se testování a tím pádem i celý projekt protáhnout o několik měsíců. Proto se počítá, že i po nasazení se na produkci nějaké chyby vyskytnou. Důležité je, aby tyto chyby měly co nejmenší dopad na zákazníka a už vůbec neměly dopad na zákazníka našeho zákazníka.

Jakmile je nějaká chyba objevena na produkci, je ihned co nejrychleji reportována zpět a ihned řešena. Důležité je řešit chybu okamžitě a bez prodlení. K reportování produkčních chyb je uzpůsobena jedna aplikace, ve které se požadavky hromadí. Zde má možnost zákazník také reportovat své špatné kroky, které provedl s reálnými daty a potřebuje jejich nápravu. Těchto požadavků se tam postupně objevuje poměrně dost, jelikož zákazník má mnoho zaměstnanců a kliknout myší jinam než člověk chtěl, se prostě stává.

3.4.1 Analýza chyb

Měsíc po každém nasazení se analyzují chyby, které byly v tomto období reportovány. Zjišťuje se množství zadaných chyb, jejich dopad na procesy zákazníka a důvod zavlečení chyby. Aby byl zákazník spokojený, je důležité, aby se nevyskytovala žádná vysoká nebo kritická úroveň chyby z produkce. Pokud se to podaří, je většinou ze strany zákazníka release hodnocen jako úspěšný. Tyto analýzy slouží také testerům, aby zjistili, proč nebyly chyby odhaleny v testech. Jestliže test, který by chybu dokázal odhalit, mezi testovacími scénáři chybí, je nutné tento scénář doplnit, aby se chybě do příště předešlo.

3.5 Úprava regresních scénářů

V době kdy analytici společně se zákazníkem provádějí analýzu a návrh, tak testeři společnosti pracují na údržbě stávajících regresních scénářů a na tvorbě nových. Údržba a tvorba probíhá na podkladech předchozího vývojového cyklu, kdy je nutné nové změny zanést do testů. Tyto změny se buď zanesou do již existujících testů, jestliže mění jejich procesní postupy, nebo jsou vytvořeny nové scénáře, které zohledňují zcela novou funkčnost, která přibyla.

Tvorba scénářů má jasný řád. Je nutné, aby všichni dodržovali stejné postupy, což umožňuje, že následné úpravy může provádět kdokoliv a také, že při otevření scénáře všem bude jasné, co mají dělat. Scénář se skládá z několika kroků, a vedle každého

kroku musí být uvedena informace, jaký výstup je očekáván. Scénáře většinou obsahují pět až dvacet těchto kroků. Je velmi vhodné jej doplnit případnými dotazy do databáze nebo obrázky, jak má vypadat výsledek, ulehčuje to správnou vizuální kontrolu. Vzhledem k tomu, že nových věci se během jednoho, přibližně půl roku trvajícímu vývoje, objevuje hodně a scénářů je skoro pět tisíc, je to velmi zdlouhavá a mravenčí práce.

Když tester vyhodnotí, že je potřeba nový scénář napsat a učiní tak, je důležité tento poznatek také zapsat do katalogu scénářů, aby byl tento scénář při dalším testování projit.

3.6 Sepisování dokumentace

Úkolem testerů vybrané společnosti je také udržování manuálu k veškerému vyvinutému systému. V těchto manuálech jsou veškeré informace jak s danou oblastí aplikace pracovat. Popisuje veškeré možnosti užití a jak docílit požadovaných výsledků. Stejně jako úprava scénářů i úprava a tvorba nových manuálů probíhá v době po nasazení a při přípravách nového cyklu. Dokumentace musí obsahovat veškeré informace jak pracovat s daným systémem a také mnoho obrázků důležitých situací, které zjednodušují pochopení. Když se změní vizuální zobrazení, ale funkce jsou zachovány, i přesto je nutné obrázky v katalogu změnit aby byly aktuální.

Manuály jsou následně odesílány zákazníkovi, který je využívá k zaškolování nových zaměstnanců, případně ke školení stávajících zaměstnanců z důvodů změny aplikace. Ovšem manuály využívá také společnost, i zde jsou využity k zaučení nových testerů, aby rychle pochopili principy fungování softwaru. Vybraná společnost využívá studenty jako brigádníky k regresnímu testování, kde je potřeba mnoho testerů a právě při nabírání nových brigádníků hrají manuály důležitou roli.

3.7 Demo představení

Jedná se o setkání se zákazníkem za účasti analytiku a testerů. Na této schůzce jsou zástupcům zákazníka představeny vyvinuté novinky na základě jejich požadavků. Ukazuje se, jak se s daným softwarem pracuje, kde je možné případně upravovat parametry, jestli je lze upravit přímo v systému a jak, případně jestli je úpravu možné provést jen přes databázi.

Zákazníkovi jsou ukázány veškeré procesy a průchody systémem, kdykoliv má zákazník možnost se na cokoliv zeptat a je mu zodpovězen jeho dotaz. Také jsou poznamenány jeho výtky, na základě kterých se systém ještě upraven, tak aby odpovídal jeho představám a požadavkům.

3.8 Pravidelné schůzky

Části týmů se pravidelně schází na schůzkách, kde řeší problémy, které je aktuálně zdržují nebo se snaží těmto problémům předcházet. Schůzka se provádí většinou na dvoutýdenní bázi, ovšem jestliže nejsou žádné podněty k řešení, schůzka se ruší, aby se neplýtvalo časem.

Ve vybrané společnosti mají své pravidelné schůzky vývojáři, a to buď všichni dohromady, nebo podle náplně jejich práce. Většinou se zde řeší problémy, jakými pravidly se bude řídit postup při vývoji a jaké postupy se musí dodržovat, aby všichni spolu mohli spolupracovat a zdrojový kód byl srozumitelný a přehledný.

Svoji pravidelnou schůzku mají také testeři. Nejčastěji se řeší aktuální stav testů, jejich postup vůči vytyčenému plánu, a problémy které testovaní brzdí a bylo by potřeba je odstranit. Dále se zde rozhoduje, kdo bude zodpovědný za jakou oblast a vypracuje k ní následně úpravu a nové scénáře, případně doplní informace do manuálů.

Velmi důležité jsou taky schůzky vedení týmu, na těchto pravidelných schůzkách se vyššímu managementu společnosti představují postupy a změny většinou za posledních čtrnáct dní. Řeší se zde palčivé problémy ohledně možného navýšení zaměstnanců v jednotlivých týmech, problémy s nedodržením termínu, požadavky na vybavení pro zaměstnance. Vedoucím týmů jsou zde představovány změny a novinky ohledně fungování a chodu společnosti a je po vedoucích požadováno, aby tyto informace předali všem svým podřízeným ve svých týmech. Jestliže má dojít k zásadním změnám, tak si vedoucí svolá svůj tým a na této schůzce předá informace, které obdržel od svých nadřízených.

3.9 Složení týmu

V týmu ve kterém mám možnost působit je se mnou přibližně dalších šedesát až sedmdesát lidí, kteří se snaží o úspěšné fungování projektu. Atmosféra v celém týmu je klidná a poměrná vstřícná.

3.9.1 Vedení

O vedení se stará trojice lidí v čele s vedoucím manažerem, senior projekt manažerem a junior projekt manažer. Vedoucí manažer jedná s vrchním vedením jak vybrané společnosti, tak s vrchním vedením zákazníka. Jeho úkolem je zajišťování fungováni týmu a spolupráce s nejvyšším vedením společnosti.

Projektoví manažeři jednají s vedoucími pracovníky zákazníka a stanovují termíny jednotlivých vývojových cyklů, zastřešují jednotlivé projekty a je-li to nutné, tak zajišťují implementaci našeho produktu u třetích stran. Kontrolují postup v rámci projektů, sestavují reporty odvedené práce a snaží se odstraňovat problémy, které nastaly v průběhu projektu.

3.9.2 Release leader

V týmu se nacházejí dva release leadeři, jejich úkolem je zajištění veškerých prostředí jak pro vývoj, tak pro testy. Zodpovídají za to, že prostředí budou mít požadovanou kvalitu a budou obsahovat vše, co je potřeba pro úspěšný vývoj a testování.

Zajišťují a nastavují spuštění pravidelných nasazovacích oken na vývojová a testovací prostředí, případně když jsou o to požádání, provádějí mimořádná nasazení. V případě nedostatku paměti zajistí její navýšení nebo smazání nepotřebných dat. Účastní se veškerých nočních uvolnění na produkci, protože jsou to právě oni, kdo provádí nasazení.

3.9.3 Vývojáři

Společně s testery jsou nejvíce zastoupeni mezi zaměstnanci společnosti. V rámci vybrané společnosti působí mnoho vývojářů zaměřených na různé programovací jazyky (VB6, .NET, Java, databáze, JavaScript,…). V rámci jednotlivých odvětví jsou rozděleni do juniorských a seniorských pozic, k tomuto rozdělení dochází na základě zkušenosti a znalosti jednotlivých zaměstnanců. Je tedy logické, že složitější úkoly zpracovává senior vývojář, nebo na problém minimálně dohlíží.

V poslední době se společnost musí vyrovnávat s nedostatkem vývojářů v neoblíbených či starších jazycích, mezi tyto posty bychom mohli zařadit kvalitní programátory v oblasti VB6 a .NET. V poslední době se společnosti alespoň částečně daří přepisovat zastaralé části z VB6 do .NETu.

3.9.4 Software testeři

Software testery bychom mohli v týmu rozdělit do tří kategorií. První dvě kategorie by byli stálí zaměstnanci rozdělení podle zkušeností a znalosti opět na seniorské a juniorské pozice, třetí kategorií jsou studenti, kteří vypomáhají jako brigádníci.

Každý senior tester odpovídá za jemu přidělené moduly a v případě, že někdo něco neví, měl by to být právě on, kdo dokáže dotyčnému poradit v rámci fungování daného modulu. Společně s junior testerem jsou zodpovědní za testováni nové funkčnosti a psaní veškerých novým scénářů a manuálů.

Tester brigádník je v týmu od toho, aby testoval v rámci regresního testování pomocí již existujících scénářů a reportoval nalezené chyby a nesrovnalosti. V případě, že si není jistý zda se jedná o chybu, měl by se obrátit na senior nebo junior testera o radu.

3.9.5 Datový sklad

Vývojáři v oblasti datového skladu jsou zaměstnanci specializující se databáze. Náplní jejich práce je tvorba databázových dotazů, které sbírají data podle zadání zákazníka, tyto data jsou následně analyzována a je možno z nich vydávat reálné závěry podložené velkým množstvím kvalitních dat. Na základě těchto dat může zákazník přesně hodnotit své působení na trhu a má data pro své analýzy a plánování.

4 Návrh řešení

Vzhledem k tomu, že vybraná společnost pracuje pro zákazníka, který často mění své požadavky i v závislosti na legislativních změnách a základní funkčnosti, která mu generuje zisk. Potřebuje dostávat tyto změny do ostrého provozu co nejdříve, a proto by mělo dojít k zavedení agilního přístupu vývoje softwaru. Nejvhodnější metodikou pro vybranou společnost je scrum. Aby zavedení scrumu bylo účinné, musí své postupy změnit nejen vybraná společnost, ale také její zákazník. Bez společného úsilí se transformace do scrumu nikdy nemůže povést.

4.1 Potřebné změny ve společnosti

Vybraná společnost musí zcela změnit svůj přístup k vývoji softwaru. Všichni zaměstnanci, kterých by se změna dotkla, musí projít školením, aby věděli jak se ve scrumu pracuje a co bude nyní nově po nich vyžadováno. Základní změnou pro vedení česko-slovenského týmu je opuštění autoritativního řízení a přenést část svých kompetencí na zaměstnance.

Ve společnosti musí dojít k přerozdělení jednotlivců je projektovým scrumových týmu. Tento zásah je pouze o tom přesadit své zaměstnance na jiná místa v rámci budovy.

4.2 Potřebná změna na straně zákazníka

Vzhledem k tomu, že vybraná společnost spolupracuje pouze s jedním zákazníkem je potřeba, aby částečnou transformací prošel i zákazník. Přesvědčit zákazníka nebude složité, jelikož jemu z transformace plyne mnoho výhod.

Na straně zákazníka je nutné definovat pozici vlastníka projektu. Tato osoba bude úzce spolupracovat s jednotlivými projektovými týmy. Na tuto osobu je kladen důraz, aby udržovala a správně tvarovala produktový backlog.

4.3 Rozdělení týmu

Vzhledem k tomu, že scrum pracuje s malými soběstačnými týmy, musí dojít k přerozdělení zaměstnanců. Nyní zaměstnanci sedí a spolupracují navzájem víceméně pouze v rámci své specializace podle znalostí. Transformací dojde k vytvoření soběstačných týmu, které jsou jako celek schopné zařídit celý průběh jednoho úkolu. Od prvotní definice úkolů, kterou zpracuje analytik týmu společně se zákazníkem, následně analytik namodeluje řešení. Vývojáři za jednotlivé technologie úkol vyvinou a následně tester úkol otestuje a představí zpět vlastníkovi produktu, který se přesvědčí, že je vše, jak bylo požadováno. Takto každý jeden projektový tým zastřeší své úkoly.

4.3.1 Velikost týmu

Vzhledem k tomu, že tým musí být soběstačný a být schopen pravidelně dodávat je nutné, aby se skládal z vývojářů zastupujících jednotlivé technologie, analytika, software testera a také scrum mastera. Vzhledem k tomu, že společnost využívá technologicky osoby, které zpracovávají databázi, javu a .NET je nutné mít všechny tyto jedince v každém týmu. Výjimku může tvořit zástupce .NET, jelikož existuje dost požadavků na úpravu systému, které jeho činnost nevyžadují.

Minimální složení týmu dle technologií tedy je: databáze, java, .NET, software tester 2x, analytik a scrum master. Takto složený tým je schopný samostatně fungovat a dodávat zákazníkovi výsledný produkt tak jak si bude přát. Maximální velikost týmu by neměla přesáhnout 12 lidí. V tom případě by ideální složení týmu vypadalo takto: databáze 2x, java 3x, .NET, analytiky 2x, software tester 3x a scrum master.

4.3.2 Průběh rozdělení

Rozdělení musí proběhnout systematicky. Hlavním cílem je, aby vzniklé týmy jako garanti pokrývali co největší část stávajícího systému. Proto je nutné, aby specialisté za různé technologie pro jednu oblast systému byli v jednom týmu. Vzhledem k tomu, že společnost na trhu existuje už 20 let, je jisté, že bude náročné rozdělit všechny oblasti přesně do týmu. Z toho důvodů bude nutné na začátku zapracovat na doplnění znalostí a

zajistit garantování vždy v rámci jednoho týmu. To ovšem neznamená, že na jednu oblast musí být garantem právě jeden tým.

Samotné rozdělení tedy proběhne na základě znalostí jednotlivých členu česko-slovenského týmu. V případě, že několik lidí má stejné znalosti, může zařazení proběhnout dle dohody mezi nimi, podle jejich preferencí a to buď podle garantování části systému, nebo podle preferencí v rámci personálního složení týmu.

4.4 Průběh jednoho release

Společnost zvýší počet svých uvolnění na produkci pro zákazníka ze současných dvou nebo tři na přibližně dvanáct ročně. To znamená, že každý měsíc budou zákazníkovi na produkci nové věci, které mu mohou ihned generovat zisk. Každý release se bude skládat ze sprintu zajišťujících vývoj nových požadavků, času na regresní otestování a doby na přípravu uvolnění.

Zvýšené množství releasu za jeden rok přináší velkou výhodu v tom, že může rychle reagovat na změny na trhu. Vzhledem k tomu, že zákazník působí na finančním trhu, na kterém probíhá mnoho legislativních změn, může na ně rychle a pružně reagovat. Tato skutečnost bude pro zákazníka velmi cenná hlavně na slovenském trhu, kde vláda nedává mnoho času na reakci na jimi vydané změny.

4.4.1 Množství sprintu

Do každého releasu budou připadat dva vývojové sprinty (při delších dovolených – letní prázdniny, Vánoce, se množství může zvýšit na tři). Dva sprinty jsou dostatečná doba na to, aby se mohla vyvinout část produktu, která při nasazení přinese zákazníkovi přidanou hodnotu. Navíc tato doba umožňuje do prvního sprintu vždy zařadit složitější věci, u kterých není zcela znám jejich dopad a následně je v druhém sprintu může tým dodělat případně upravit, tak aby byl zákazník s výsledkem práce spokojen.

4.4.2 Doba nasazení user story

Od prioritizace user story k jejímu uvolnění na produkci dojde přibližně za dva měsíce. Tato doba v sobě skrývá čas pro vývoj jednotlivých user story, což jsou dva čtrnáctidenní sprinty. Dále čas pro regresní otestování celého systému, který trvá asi patnáct až dvacet dní. V této době probíhá také testování softwaru na straně zákazníka. Zde si sám zákazník ověří, že nově vyvinuté části fungují tak, jak si představoval. Následně je potřeba pár dní pro přípravu samotného nasazení. Samotné nasazení na produkci proběhne v noci, aby byl co nejmenší dopad na zákazníky.

4.5 Průběh jednoho sprintu

Ideální doba trvání jednoho sprintu je dva týdny, tedy deset pracovních dní. Za tuto dobu je potřeba všechny user story vyvinout, otestovat a představit vlastníkovi produktu.

4.5.1 Schůzky v rámci sprintu

Schůzky v rámci sprintu se budou vzájemně opakovat. Některé budou za cely sprint jen jednou, jiné každý den.

Každý sprint začíná plánovací schůzkou. Tato schůzka se odehrává přibližně dva nebo tři dny před koncem předcházejícího sprintu. Účelem této schůzky je tým seznámit s požadavky na další sprint. Na této schůzce jim analytik a vlastník produktu představí user story, které si pro ně na další sprint nachystali. Zde se mohou účastníci zeptat na doplňující informace a ve chvíli, kdy je všem jasné, co je zapotřebí udělat, dojde k odhadnutí náročnosti. Každý člen týmu ukáže kartičku s body, která značí náročnost a výsledná hodnota se zapíše k dané user story. Ve chvíli, kdy jsou ohodnoceny všechny user story, tak tým rozhodne které úkoly si nabere do dalšího sprintu. Tým se musí řídit zásadou, že nabírá podle priorit, které určil vlastník produktu a že nabere pouze tolik, kolik je schopný za deset pracovních dnů dodat.

Další schůzkou ve sprintu je denní stand-up, tato schůzka probíhá každý den před započetím prací. Vzhledem k tomu, že ve společnosti je povinná pracovní doba od 9:00, je nutné, aby schůzka začala v rozmezí 9:00-9:10 jinak její účel postrádá smysl. Smyslem schůzky je informovat ostatní členy týmu o tom co kdo stihl předchozí den udělat a co bude dělat dnes.

K předání odvedené práce vlastníkovi produktu slouží sprint review. Tato schůzka musí být v každém sprintu minimálně jedna. V případě, že tým má ve sprintu větší množství user story, které postupně odbavuje, je možné tuto schůzku svolávat častěji. Čím dřív se vlastník produktu dozví, že je vše v pořádku, případně že je potřeba v dalším sprintu zapracovat ještě nějaké změny, tím lépe.

Poslední pravidelnou schůzkou sprintu, která se odehrává na konci každého sprintu, je retrospektiva. Zde se všichni vyjádří, co se povedlo a co naopak ne a ze vzniklých chyb se do budoucna poučí. Tato schůzka je velmi účinná pro dlouhodobé úspěšné působení jednotlivých týmu. Zde si tým utváří svou identitu a může vyladit své postupy, tak jak jim to vyhovuje. Právě výsledky retrospektiv utvoří malé rozdíly ve fungování jednotlivých týmů, které budou zajišťovat vývoj pro stejného zákazníka.

4.5.2 Vývojová prostředí

Z důvodů práce několika týmu najednou, je nutné, aby každý tým dostal svoje vývojové prostředí. Na tomto prostředí dochází k vývoji samotné user story a jejímu testování jakožto samotné jednotky, tak i jako součást celého systému. V případě, že je vše úspěšné a vlastník produktu souhlasí s výsledkem práce, je část kódu převedena na společné integrační prostředí. Toto prostředí slouží k integraci user story s celým systémem a taky k integraci s novými úkoly jiných týmu. Ve chvíli, kdy integrační testy dopadnou úspěšně, poté lze user story označit za hotovou.

Vzhledem k tomu, že dochází k vývoji pro český a slovenský trh, je potřeba mít pro každý trh zvláštní vývojová prostředí, jelikož ne vždy bude požadavek pro obě země vypadat stejně. Tudíž každý tým má na každý sprint k dispozici dvě vývojová prostředí dle destinací.

Pro integrační testy musí být k dispozici taktéž dvě prostředí (dle zemí). Na těchto prostředích dochází k seskupování všech nových věcí, které vytvořily všechny týmy během sprintu s již stávající funkčností. Když integrační testy dopadnou úspěšně, následující den se nová funkcionalita dostává na prostředí, kde má přístup i zákazník a může si zde veškeré novinky vyzkoušet. Na těchto prostředích po skončení vývoje ve sprintech dochází k provádění regresního testování.

4.5.3 Další práce ve sprintu

Ve chvíli, kdy jakýkoliv člen týmu nemá dostatek práce na celý sprint, případně čeká, než bude provedená práce, na kterou musí navázat, nezahálí a stále se na nějaké práci podílí. Pro každého člena týmu existují menší i větší úkoly, které může během sprintu zpracovávat a tím rozšířit působnost celého týmu nebo zlepšit své schopnosti.

V případě analytiků dochází k řešení úkolu do budoucích sprintů a vytváření zásob úkolů, které je možné kdykoliv nabrat do sprintu. Dále analytik může zanášet do programu Enterprise Architect další části systému, které nejsou správně zdokumentované.

Z pohledu vývojáře je možné odstraňovat technologický dluh, který byl způsobený buď nutností rychlého vývoje nebo zastaralými postupy. Odstraňování technologického dluhu pomáhá spolehlivějšímu fungování systému a taky jeho zrychlování. Vývojář se také může věnovat sebevzdělávání, čímž zvýší svou kvalifikaci. V neposlední řadě se může věnovat tvorbě manuálu pro oblast, kterou má na starosti a tím umožní předávání znalostí, ať už v případě svého odchodu nebo tvorbě zastupitelnosti v rámci týmu nebo i mezi týmy.

Nejčastěji má chvíli času na práci mimo sprint software tester a to vzhledem k tomu, že nějaký čas trvá, než dostane první část úkolu k testům. Tento prostor lze nejčastěji vyplnit práci na regresních testech, případně úpravou testovacích scénářů. Nutnost úprav je poměrně častá, jelikož během každého sprintu vznikne mnoho nových věcí, které buď upravují stávající činnost, nebo zavádí zcela nový kus aplikace.

4.6 Život user story

Každou user story by mělo být možno samostatně vyvinout, otestovat a nasadit na produkci s přidanou hodnotou pro zákazníka. Vznik user story je dlouho před samotným vývojem. Je nutné, aby se sešel vlastník produktu či další jeho spolupracovníci s analytikem a představili mu svou vizi nového softwaru. Může se jednat o zcela novou část či úpravu stávajícího stavu. Při představení se lze bavit o velké vizi, kterou ale je nutné následně rozdělit na menší kousky. Právě tyto user story analytik zpracuje, navrhne řešení, které může spočívat v úpravě již existujícího nebo vývoji zcela nové části aplikace. Svůj návrh následně představí vlastníkovi produktu a odsouhlasí si, že dospěl k tomu, co bylo požadováno.

Ve chvíli kdy je takto user story připravená a prošla prioritizací, může si ji tým nabrat do sprintu. V samotném sprintu na ni vývojáři podle jednotlivých zaměření postupně pracují, jestliže to možnosti umožňují, mohou pracovat zároveň. Následně user story otestuje software tester, ve chvíli kdy je to možné, tak tester testuje již v průběhu vývoje, aby případné chyby odhalil co nejdřív.

4.7 Využití podpůrných programů

Společnost nyní využívá několik programu, které po malé úpravě mohou velmi dobře pomoct řízení ve scrumu. V tomto ohledu bude jednu z nejdůležitějších funkcí zastávat program JIRA, tento program má možnost být nastaven na práci ve scrumu. Díky tomuto nástroji, lze jednoduše uspořádávat produktový backlog, dále umožňuje zaznamenávat posun user story v rámci sprintu. Každé user story zařazené do sprintu lze zadat stav a přiřadit osobu, je tedy hned jasné co se s danou user story děje a kdo za tento úkon právě zodpovídá. Jednotlivá pole jsou uživatelsky nastavitelná, takže můžou být stavy: předvývoj, vývoj, testy, čekaní na schválení vlastníkem produktu či integrační testy a následně hotovo. Zde mohou získat rychle a přehledně informaci i jednotlivci, kteří nejsou součástí týmu.

Pro lepší přehled aktuálního stavu v rámci týmu je vhodné využít podobným stylem klasickou tabuli na stěně. Zde se mohou k user story hlásit jednotliví členové týmu,

přidávat poznámky a značit postup prací. Velkou výhodu má tato tabule, když probíhá pravidelný denní stand-up. Díky tomu se může každý vyjádřit ke všem user story, které jsou nabrány do sprintu. Vizuální podoba ulehčuje pochopení změn na dané user story.

Velmi důležitou roli má také program Enterprise Architect. Tento program umožňuje analytikům zpracovávat požadavky zákazníka a vše modelovat do navzájem propojených diagramů. Z těchto materiálu následně vývojáři a software testeři čerpají informace pro svou práci. Velkou výhodou je propojování jednotlivých oblastí v programu. To umožňuje poměrně rychle a přesně určit místa, na která může daná změna mít vliv. V případě, že kdokoliv z týmu znejistí třeba i u stávajících věcí, lze dohledat, jak bylo žádáno, aby se aplikace chovala.

4.8 Přínos změny pro společnost a zákazníka

Hlavní výhodou zavedení metodiky scrum, je zkrácení doby zavedení nových požadavků do ostrého provozu. To umožní zákazníkovi čerpat výhody z odvedené práce mnohem dříve. Se zmenšeným množstvím pravidelně uvolněných změn výrazně klesá chybovost. A to jak z důvodů nasazování menšího počtu změn (ovšem v součtu za celý rok je změn nasazeno více), tak i z důvodu intenzivnějšího testování. Regresní testy budou probíhat každý měsíc, navíc jednotlivé user story jsou testovány již v rámci sprintu a také probíhají integrační testy ke každé user story.

Z pohledu zákazníka je velmi výhodná možnost rychlé reakce na dění na trhu. Ve chvíli, kdy zákazník dostane nápad, jak působit na trhu, tak nemusí už čekat více než půl roku, než svůj produkt na trh dostane. Umožní mu to velkou konkurenční výhodu.

Díky nasazování menšího počtu změn jsou také kratší a úspěšnější integrační testy. Vzhledem k tomu, že každý úkol je testován při jeho vzniku, dochází k nasazování méně chybného kódu.

Vzhledem k užší spolupráci v týmu, je zajištěna vyšší sdílnost informací o aplikaci napříč všemi pozicemi. U testerů je velká výhoda v přítomnosti vývojářů. Mohou se jich

kdykoliv zeptat na technické věci a nejen rychle vyřešit problém, ale také se mohou mnoho věci přiučit a tím zvýšit svou kvalifikaci.

Nedílnou součásti je také zvýšení prestiže společnosti. Mnoho vývojářů chce pracovat moderními postupy a rádi dají přednost společnosti, která využívá scrum místo tradičního vodopádu.

Zákazník finančně hodnotí kvalitu odvedené práce. Toto hodnocení je postaveno na dvou pilířích: kvalitě a času. Přechod společnosti na řízení pomocí scrumu zvýší kvalitu dodávaného produktu a také zkrátí čas, který je potřeba k uvolnění nových věcí na produkci. To povede k lepšímu hodnocení odvedené práce od zákazníka a tím pádem k zvýšení příjmů.

4.9 Možnost rozšíření na celou společnost

Vybraná společnost je rozdělena na několik části dle zemí, pro kterou vyvíjí nový software. Změna scrum by byla provedena jen na oddělení, které dodává software pro českého a slovenského zákazníka. Ve chvíli, kdy zástupci společnosti zjistí přínosy nejen pro ně, ale i pro zákazníka, pak je možné zavést postupně scrum i na jiných odděleních pracující pro zákazníka působícího na jiném trhu. Tímto by se veškeré výhody z působení ve scrumu promítly do všech zemí, kde nyní zákazník vybrané společnosti působí.

5 ZÁVĚR

Zavedení agilní metodiky scrum je pro vybranou společnost i jejího zákazníka velmi výhodné. Díky scrumu dojde k užší spolupráci mezi zaměstnanci obou firem. Hlavní výhodou pro vybranou společnost je přechod z vývoje velkých kusů produktu na menší části. Tímto dojde k tomu, že zákazník může postupně přesně specifikovat své požadavky a dodaný produkt přesně splní jeho přání.

Metodika scrum je poměrně nová a na českém trhu zatím ne moc používána. Vybraná společnost tímto krokem posílí své postavení na trhu a umožní ji to získat mnoho nových lidí, které láká pracovat novými moderními postupy. Další výhodou je snížení pravděpodobnosti výskytu chyb v ostrém provozu, což povede ke zvýšení reputace u zákazníka a snížení sankcí za způsobené výpadky.

Zákazník získá možnost dostat své nejvíce profitující požadavky co nejdříve do ostrého provozu, což mu umožní generovat zisky mnohem dříve než při využití vodopádů. Navíc, při správné prioritizace, lze velmi brzo odstranit nedostatky nebo zcela od projektu ustoupit, čímž se ušetří náklady, které by byly vyžadovány na další vývoj. Pravidelné a rychlé uvolňování nových věcí do ostrého provozu umožní zákazníkovi lépe reagovat na změny na trhu a poskytne mu to velkou výhodu vůči konkurenci. Navíc může velmi rychle a přesně reagovat na změny vyplývající, ze změn zákonů v zemích, ve kterých působí na trhu.

6 SEZNAM POUŽITÝCH ZDROJŮ

CEJTHAMR, V. a J. DĚDINA., 2010. *Management a organizační chování*. 2. vyd. Praha: Grada, 344 s. Expert. ISBN 978-80-247-3348-7.

DOLANSKÝ, V., V. MĚKOTA a V. NĚMEC., 1996. *Projektový management*. 1. vyd. Praha: Grada, 372 s. ISBN 80-7169-287-5.

DOLEŽAL, Jan., 2016. *Projektový management: komplexně, prakticky a podle světových standardů.* 1. vyd. Praha: Grada, 424 s. ISBN 978-80-247-5620-2.

DOUCEK, Petr., 2004. *Řízení projektu informačních systémů*. 1. vyd. Praha: Professional Publishing, 163 s. ISBN 80-86419-71-1.

FIALA, Petr., 2004. *Projektové řízení: modely, metody, analýzy*. 1. vyd. Praha: Professional Publishing, 276 s. ISBN 80-86419-24-x.

JONÁK, Stanislav., 2015. *Vývojový cyklus software: Dominují agilní metodiky trhu?*. [online] 15. dubna 2015 [cit. 2017-05-18]. Dostupné z: http://www.middleware.cz

KNESL, Jiří., 2016. *Klady a zápory různých délek iterací.* [online] 31. srpna 2016 [cit. 2017-05-18]. Dostupné z: http://www.knesl.com

KNESL, Jiří., *O Scrumu.* [online] [cit. 2017-05-18]. Dostupné z: http://www.skoleniscrum.cz

KNESL, Jiří., 2012. *SprintMethod: agilní metodika vycházející ze Scrumu*. [online] 2012 [cit. 2017-05-18]. Dostupné z: http://www.sprintmethod.cz

MYSLÍN, Josef., 2016. *Scrum: průvodce agilním vývojem softwaru.* 1. vyd. Brno: Computer Press, 167 s. ISBN 978-80-251-4650-7.

ŠMÍDA, Filip., 2007. *Zavádění a rozvoj procesního řízení ve firmě*. 1. vyd. Praha: Grada, 293 s. Management v informační společnosti. ISBN 978-80-247-1679-4.

ZIKMUND, Martin., 2010. *Agilní projektové řízení – novinka stará přes 20 let.* [online] 12. dubna 2010 [cit. 2017-05-18]. Dostupné z: http://www.businessvize.cz