Titulní list

Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	Zimní 2016/2017
Autoři	Luděk Adamec – xadal12, Dac Ha Tran – xtrah03, Erik Ziegler – xziee01
Téma	So How Do You Make Agile Successful?
Datum odevzdání	18. 12. 2016

Abstrakt

Tato semestrální práce dává čtenáři ucelenou představu o tom, jaké jsou výhody a nevýhody agilních metodik. Na základě analýzy nedostatků a nevýhod následně poskytuje několik způsobů, jak učinit agilní metodiku účinnou a efektivní. Pro lepší představu, jak tyto způsoby zefektivnění agilních metodik aplikovat, jsou v závěru práce uvedeny příklady úspěšného, agilně vyvíjeného softwaru.

Klíčová slova

Agilní, Metodika, Scrum, Spotify

Obsah

1. Úvod

Úspěšnost agilních metodik v dnešní době převyšuje úspěšnost metodik tradičních, tedy rigorózních. Proč je ale třeba řešit otázku „Jak udělat agilní metodiky úspěšné?“. Taková otázka na první pohled může naznačovat, že agilní metodiky nejsou správnou volbou pro vývoj softwaru a nejsou ani příliš efektivní. Skutečnost je ale jiná. Agilní metodiky lze považovat za funkční a vhodné pro vývoj softwaru a tuto otázku je nutné řešit z jiného důvodu. Boom agilních metodik láká stále nové vývojáře k jejich používání, přibývá zákazníků, kteří jsou schopni tyto metodiky respektovat, ale na druhé straně se množí případy, kdy mnoho agilních projektů končí fiaskem, i když byl dodržován agilní manifest. Tato semestrální práce ukazuje důvody, proč se to děje a zároveň hledá řešení, jak tomu předcházet.

Prvním cílem této semestrální práce je představit čtenáři výhody a nevýhody agilního přístupu při realizaci softwarových projektů. Druhým cílem je analýza nedostatků a nevýhod agilních metodik s následným návrhem opatření, která by zvýšila úspěšnost vývoje softwarových projektů agilním přístupem.

K dosažení cíle práce jsme nashromáždili a prostudovali informace o problematice agilních metodik. Z nasbíraných informací byly vytaženy ty nejpodstatnější, které jsme zkompletovali a využili tak, aby byl vystižen výše formulovaný cíl. V práci jsme použili metody analýzy a syntézy. Analýza byla použita na konkrétní informace z jednotlivých využitých zdrojů a syntéza při sumarizaci závěrů z použitých analyzovaných zdrojů.

2. Přednosti agilních metodik

Metodiky vycházejí z Agilního manifestu, který byl sepsán na základě rozsáhlých zkušeností odborníků z vývoje i analýzy. Díky manifestu lze lépe zvládat problémy vyskytující se u řízení projektů podle tradičních metodik. Velkou výhodou je menší vytíženost manažera a zároveň větší výkonnost týmu. Pro vedoucího manažera by mělo být mnohem snadnější lépe zvládnout větší množství zaměstnanců i rozsáhlejší projekt. Tým dopředu také ví, jak moc práce bude schopen realizovat, a to díky zkušenosti z předcházejících iterací. Díky časovým odhadům i velice dílčích úkolů se dá korigovat rychlost týmu. Za velkou výhodu je považována možnost měnit zadání za běhu, což je například pro Waterfall nemyslitelné. Výsledný produkt je dodáván klientovi dříve, protože se doručuje část funkcionality po každé iteraci. Produkt má také přislíbenou určitou míru kvality díky automatizovaným testům. (1)

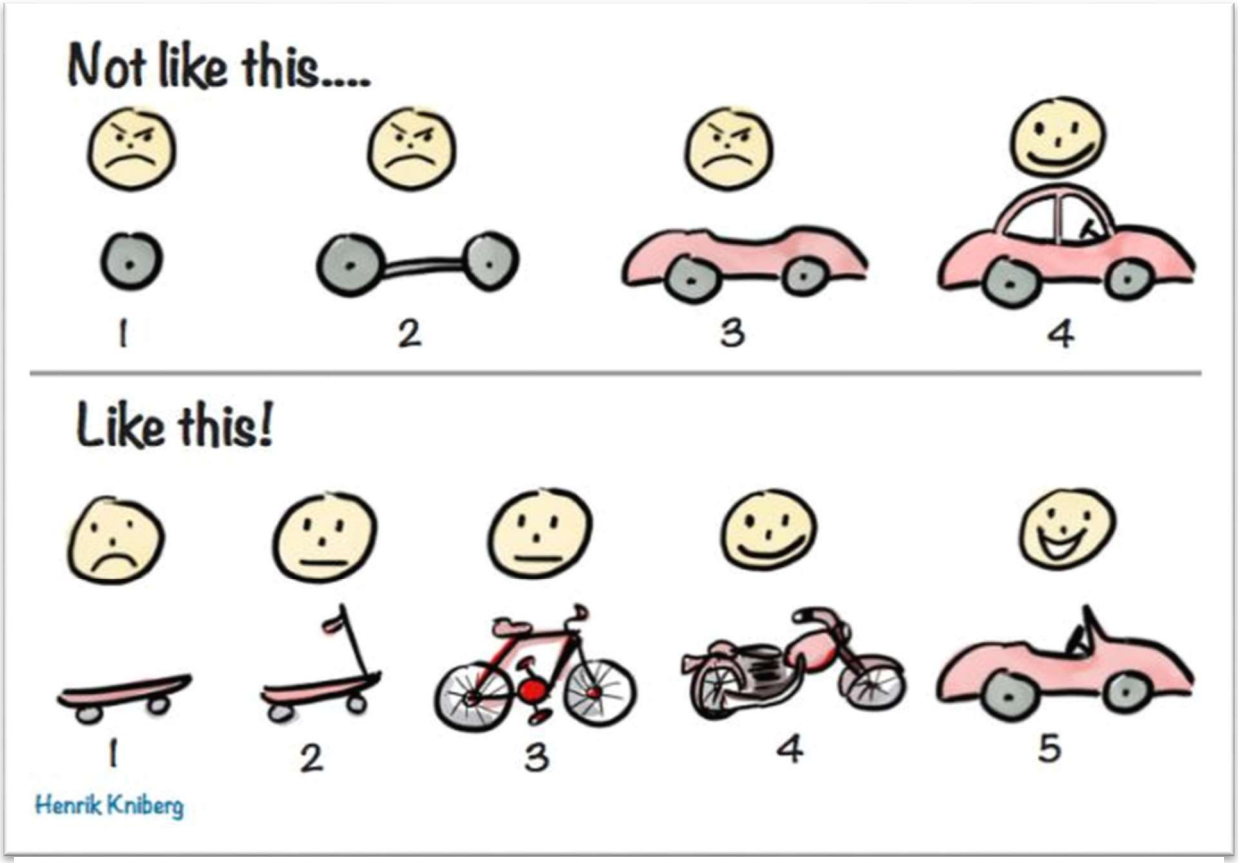
3. Nevýhody a nedostatky agilních metodik

Zábranou pro použití agilních metodik jsou především velice rozsáhlé projekty, u kterých je nedostatek komunikace s klientem. Slabinou může být i fakt, že agilní metodiky se snaží vynechávat přípravu údajně nepotřebné dokumentace, modelování a snaha je rovnou přejít k samotnému programování. To ale metaforicky znamená, že začínáme stavět dům bez potřebných nákresů. Toto jsou ale spíše technické problémy. Zásadnější problémy přinášejí lidé. V oblasti vývoje software totiž existuje mylná představa, že ho může řídit kdokoliv, ať už programátor, který se chce někam posunout či někdo z úplně jiného oboru. Člověk neznalý problematice tak může přinést do vývoje mnoho chaosu. Toto je přehled nejčastěji se vyskytujících problémů. V další části práce se podíváme na podrobnější ukázku špatného a správného použití agilního vývoje se zaměřením na lidský faktor. (2)

4. Špatný versus správný přístup k agilní metodice

Pro lepší pochopení chyb, kterých se vývojové týmy dopouští je pro účely této semestrální práce použita metafora s výrobou auta. Nejedná se o skutečnou výrobu auta, ale lze za něj dosadit libovolný produkt. Na následujícím obrázku horní řádek ilustruje nepochopení iterativního vývoje a dodávání produktu po použitelných částech. V dnešní době mnoho projektů selže kvůli tzv. „Big Bang delivery“. Produkt je vyvíjen, dokud není úplně hotový a pak je celý dodán. Při tomto způsobu dodání se často stane, že finální software je naprosto odlišný od očekávání zadavatele. Pokud je ale agilní vývoj navrhován jako alternativa, aby se

předcházelo výše zmíněnému, tak mnoho zákazníků má obavy, že jim bude dodán nedokončený poloviční produkt.



Obrázek 1 zdroj: http://blog.crisp.se/, autor: Henrik Kniberg

Máme předpoklad, že si zákazník objednává auto. My mu po první iteraci dodáme přední kolo. Zákazník samozřejmě není nadšený a ptá se proč mu dáváme kolo, které je mu k ničemu, jelikož si objednal auto. S každou další iterací zákazník dostane produkt bližší výslednému, ale stále je rozčílený, protože ho nemůže používat. Nakonec je produkt dokončen a zákazník jásá, ale ptá se, proč mu auto nedodali hned po první iteraci a nepřeskočili těch několik zbytečných iterací. Realita je ovšem trochu odlišná. Zákazník sice po dlouhé době dostal funkční produkt, ale v průběhu vývoje neprobíhalo žádné testování, takže auto bylo designováno pouze na ne zcela správných domněnkách, co skutečně zákazník potřebuje. Prví řádek na obrázku je tedy jasný příklad zkažené agilní metodiky. Technicky se možná jedná o iterativní a inkrementální vývoj, ale kvůli absenci zpětné vazby se jedná o velmi riskantní a rozhodně ne agilní přístup.

Druhý řádek na obrázku reprezentuje správný přístup. Začíná stejným předpokladem, že si zákazník objedná auto. Tentokrát se ale nezačne rovnou bezhlavě vyrábět auto, ale nejdřív se vývojáři a konzultanti pokusí o pochopení, co vlastně zákazník potřebuje. Po položení správných otázek se ukáže, že se zákazník potřebuje dostat z bodu A do bodu B. Auto je jen jedna z možností, jak toho docílit. Tým začne zákazníkovi postupně dodávat nejmenší možné věci, které by měly naplnit zákazníkovu potřebu. Po prvním dodání minimálního funkčního produktu je požadována zpětná vazba. Na obrázku tento bod představuje skateboard. Zákazník jen stěží bude se skateboardem spokojen, ale to v tuto chvíli není důležité. Tato chvíle má za cíl hlavně, aby se tým učil. Zákazník si to nevyloží špatně, pokud mu to tým správně vysvětlí. I když se nedá mluvit o spokojenosti, tak klient má v tuto chvíli způsob, jak se dostat z bodu A do bodu B. Projekt není u konce, cílem je stále auto, ale vývoj nabírá správný směr. Samozřejmě už v tomto okamžiku se mohlo zjistit, že se klientovi nelíbí akorát barva skateboardu a po změně by mohl být projekt ukončen a ušetřilo by se velké množství peněz.

Pokud dostaneme další zpětnou vazbu, zjistíme, že se zákazník může dostat rychle k automatu na kávu, ale nelíbí se mu nestabilnost skateboardu. V další iteraci se tedy tým pokusí vyřešit, jak zvýšit stabilitu a dodá řídítka. Zatímco zákazník používá produkt, poskytuje zpětnou vazbu, tak vývoj pokračuje. Přichází další požadavek, ve kterém si zákazník přeje, aby mohl překonávat větší vzdálenosti a po další iteraci je mu dodán bicykl. Do této chvíle jsme tedy zjistili mnoho nových věcí o zákazníkových potřebách, například, že se pohybuje mezi univerzitními budovami a má rád čerstvý vítr v obličeji. Nakonec by se bicykl mohl ukázat jako mnohem lepší výsledek, než původně plánované auto. Vývoj by se tedy mohl klidně ukončit. Ukáže se, že i přes spokojenost chce zákazník více. V další iteraci se přidá motor a kolo se nahradí motorkou. Opět by zde mohl projekt skončit, ale pokud chce zákazník stále víc, dodáme mu kabriolet. Kabriolet se lehce liší od původního požadavku na auto, ale ve výsledku je díky odhalení zákazníkových potřeb vhodnější než klasické auto se střechou. (3)

5. Základem jsou lidi

Základním kamenem pro úspěch projektu jsou lidi. Můžeme mít sebelepší metodiky, nejpřesnější a nejdetailnější popisy procesů, ale jak nemáme ty správné lidi, projekt nemá velkou pravděpodobnost skončit úspěšně. Všechny agilní metodiky mají společný prvek – dodávat produkt (resp. jeho část) v co nejkratším čase, abychom mohli získat zpětnou vazbu (feedback). Ta je užitečná k tomu, aby se z nich mohli všichni členové týmu učit.

Většinou na začátku projektu nikdo ze zainteresovaných lidí neví na 100 % přesně finální požadavky od klientů (občas ani sami klienti neví, co by chtěli), tak proto je rychlá zpětná vazba důležitá k ucelení a vyjasnění požadavků či identifikací rizik. V každém stadiu vývoje softwaru je potřeba rychlá zpětná vazba na co nejvčasnější odhalení a oprav chyb a tím pádem i v konečném výsledku lepší finální produkt.

Vývoj SW je neustálý proces vzdělávání, učení se novým věcem a znalostem, prohlubování již získaných znalostí. Někteří manažeři mohou namítnout, že lidi jsou v práci kvůli tomu, aby pracovali, ne se vzdělávali. Avšak softwarové inženýrství je ve své podstatě práce se získanými znalostmi a jejich aplikací v praxi. Neustálé vzdělávání a rozšiřování znalostí je klíč k lepší a efektivnější práci lidí v týmu, která se také odrazí v kvalitě dodávaného produktu. Učit se a vzdělávat se je mnohem složitější, než by se mohlo na první pohled zdát. K tomu, aby se člověk učil novým věcem, musí sám chtít a mít k tomu dále uvedené vlastnosti. Musí být pokorný a mít otevřenou mysl, být schopen přiznat, že něco neví či něčemu nerozumí. Měl by mít schopnost odhalit chyby a najít příležitosti pro zlepšení. Také schopnost systematického a analytického myšlení a schopnost přicházet na řešení. V neposlední řadě také odvahu vyzkoušet nová a jiná řešení.

6. Manažer není odborník

Manažeři by si měli uvědomit, že oni odborníci nejsou. Ve finále jádro produktu tvoří programátoři, vývojáři – lidi, kteří jsou experty ve svém oboru a kteří dané problematice rozumí. Manažer by měl umět aktivně naslouchat těmto klíčovým lidem a umět zohlednit a respektovat jejich názory. V opačném případě by ho také programátoři nerespektovali,

pracovní morálka a atmosféra by klesla, a to by se bezpochyby negativně odrazilo ve finálním projektu (horší kvalita produktu, nedodržení plánu – jak z finanční, tak i časového hlediska atd.).

Primárním cílem manažera není vytvořit produkt, ale umožnit svému týmu, svým lidem, aby mohli pracovat. Vytvořit pro ně takové podmínky, ve které panuje dobrá atmosféra. Jak je dokázáno, lidé jsou nejefektivnější, když se dostanou do tzv. flow. V tu chvíli je člověk koncentrovaný pouze na danou činnost, připadá mu, jako by šla práce jednoduše, automaticky, myšlenky a nápady „samovolně“ proudily.

Manažer by měl také umět správně namotivovat svůj tým, aby každý člen věděl svoji roli, pozici a odpovědnosti. Dobrý manažer se také nebojí často delegovat odpovědnost na své lidi. Tím vzniká oboustranná důvěra a vede ke zpevnění a utužení pracovních vztahů.

Autor v článku (4) uvádí klíčové vlastnosti a znalosti, které by měl úspěšný manažer (resp. leader) mít a umět. Tak jak je to definováno podle Toyoty:

- Pozorovat práci v organizaci aktivně a s otevřenou myslí
- Aktivně naslouchat lidem co skutečně říkají
- Jednoznačně definovat problém a umět určit jeho skutečnou příčinu
- Efektivní plánování
- Umět plány převést do činností s jasně definovanou odpovědností
- Vyhradit si čas a energii pro detailní sebereflexi a umět nalézt vhodné příležitosti ke zlepšení
- Znát silné a slabé stránky každého člena týmu
- Umět motivovat a řídit lidi v celé organizaci k dosažení společných cílů (s ohledem na firemní kulturu a hodnoty)
- Schopnost naučit ostatní lidi všechny výše zmíněné body

V případě, že dojde k problému v týmu, manažer by měl umět najít příčinu a brát to jako možnost pro zlepšení než jako katastrofu. V případě selhání jedince by ho neměl manažer zkritizovat a shodit veřejně před ostatními, protože to často může vést k přesně opačným

cílům a místo zlepšení přijde zhoršení. Spíše by měl nalézt a pochopit skutečnou příčinu problému. Poté jedince nasměrovat správným směrem, pomoct a umožnit mu, aby se mohl zlepšit.

V případě neefektivity týmu, která se následně odrazí v burndown chartu, je příčina někde uvnitř týmu jako celku. Manažer by se neměl řídit a soudit efektivitu pouze podle grafů, neměl by například říkat, že burndown chart je špatně. V takovém případě dokáže programátor udělat takový burndown chart, který vypadá dobře, ale skutečné problémy to neřeší. Je třeba hledat příčinu uvnitř týmu, kterých může být víc.

Manažer by si měl klást obdobné otázky: Ví všichni lidé v týmu, co mají dělat a jakou mají přesně roli? Proč něco dělají? Jaký je jejich smysl a osobní přínos? Mají všichni dostatečné znalosti a nástroje pro splnění svého úkolu dobře a včas? Je potřeba dodatečná školení?

Zde jsou uvedeny klíčové vlastnosti týmu:

- Odvaha stát si za vlastním názorem
- Zaměření na jeden projekt – každý ví, co má dělat a zná svoji roli v týmu
- Odhodlání dosáhnout cíle
- Respekt vůči sobě i ostatním
- Otevřenost, vzájemná komunikace

K dosažení posledního bodu otevřenosti a vzájemné komunikace slouží v agilních metodikách (hlavně Scrumu) meetingy.

7. Meetingy

Meetingy neboli schůzky, setkání, jsou často kritizovanou součástí Scrumu. Denní meetingy, backlog meetingy, plánovací meetingy, revizní meetingy, retrospektivní meetingy atd. – někteří tvrdí, že jich je zkrátka příliš.

Na jednu stranu je důležité si vyjasnit co kdo přesně dělá, co se má udělat a co už naopak uděláno bylo, dochází tu k výměně názorů. Měla by být jasně definovaná odpovědnost, meetingy slouží také k tomu, aby všichni měli přehled o aktuálním stavu projektu.

Aby se předešlo takovému scénáři, že by na jednom problému pracovalo zbytečně dlouhodobě více lidí. To by bylo značně neefektivní (škoda lidských zdrojů), neekonomické

(odpracovaný čas se i tak musí zaplatit) a může to ohrozit i časový plán projektu. Takový případ svědčí o neexistující nebo spíše špatné komunikaci mezi členy týmu.

Na druhou stranu však i když denní meetingy trvají v průměru jenom kolem 15 min, ve skutečnosti takové přerušení zabere daleko více času. Jak už bylo výše zmiňováno, nejlépe se pracuje v tzv. flow, avšak s vědomím, že se například za půl hodiny koná meeting, tak člověk má větší problémy dostat se do tohoto stavu, protože ví, že za chvíli stejně bude muset svoji práci přerušit, tak raději sleduje čas. To samé platí i po skončení meetingu, trvá nějakou dobu, než zase vypustí myšlenky z meetingu a zorientuje se a zabere zase do své práce.

Takové každodenní meetingy může mít negativní následky pro efektivitu práce, protože může celý Scrum obrátit v tzv. mikromanagement, protože člověk musí každý den odpovídat na otázky typu co jsi včera dělal? Co dneska plánuješ dělat? Proč jsi neudělal to, co jsi říkal že včera uděláš? Takový styl je podle (4) horší než klasický vodopádový model.

Tvrdí, že na to nejvíce doplatí zkušení seniorní vývojáři. Oni ze své zkušenosti ví, jak problém rozdělit na menší části a naplánovat podle toho svoji práci a harmonogram. Autor zmiňoval, že by se z takových myšlenkových procesů mohli hodně naučit juniorní developeři. Na druhou stranu by však nikdy z důvodu vytvoření hezkého a reprezentativního burndown chartu nenutil seniorní vývojáře malé části rozdělovat na ještě menší úkoly a vymýšlet každý den jiná „technická stories“, která by musela být schválena ještě předtím, než by na nich mohli pracovat.

8. Příklad z praxe - Spotify

Správnou aplikaci agilních metodik lze demonstrovat na příkladu firmy Spotify, ta vznikla v roce 2006 spoluprací Martina Lorentzona a Danieka Eka (5). Jejich cílem bylo vytvořit hudební přehrávač s knihovnou, kde celá služba byla na bázi streamování obsahu. V tuto chvíli je potřeba si uvědomit, jaká byla situace na poli konzumace multimédií a technologie s tím spojené. Jelikož rychlost připojení k internetu nebyla zdaleka tak vysoká jako dnes a pokrytí mobilní sítí taktéž, byli uživatelé zvyklí přehrávat si hudbu uloženou přímo ve svých přenosných či jiných zařízeních, což znamenalo okamžitou možnost přehrávání bez jakýchkoliv zásek či čekání. To stavělo Spotify do velice těžké pozice hned ze začátku, hlavními otázkami bylo, jak s takto omezenou technologií docílit komfortu hudby uložené přímo v zařízení a hlavně jak ukázat uživatelům, že by tuto službu vůbec měli chtít.

Pokud se vrátíme k modelu s autem a skateboardem, tak toto je přesný příklad, kdy se Spotify rozhodlo vytvořit velice základní verzi, soustředící se pouze na jedinou věc, na okamžité přehrávání skladeb bez čekání, tedy skateboard. Vývojáři se soustředili na dobu odezvy, rychlost načítání a minimalizaci požadavků na rychlost připojení (3). Jejich první verze tedy neobsahovala pokročilé možnosti, které služba umí dnes, ale podařilo se jí dosáhnout základního předpokladu pro jakýkoliv další postup. První verze ukázala, že je možné pomocí streamování dodat stejný uživatelský prožitek, jako u přehrávání z pevného média. Tento prototyp také pomohl přesvědčit investory a hudební společnosti o potenciálu celého projektu.

9. Způsob organizace Spotify

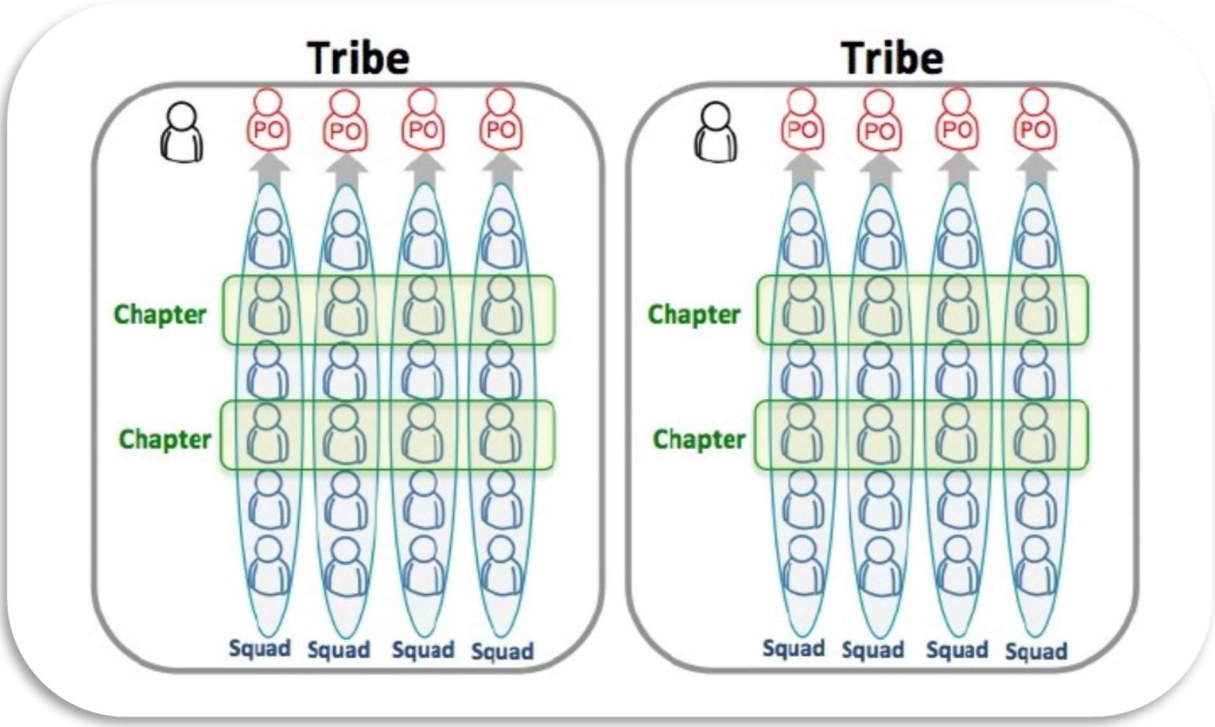
Určitá forma prototypování ovšem nebyla jediným agilním aspektem při vývoji této služby, způsob organizace a celého přístupu k vývoji je neméně zajímavý.

Spotify začalo vyvíjet po menších skupinkách držících se metodiky Scrum, avšak po určité době se některé principy této metodiky začaly jevit jako nepotřebné (6). Proto společnost upravila tuto metodiku podle svých potřeb.

Jádro tohoto systému stojí na jednotlivých „četách“ (squads), což jsou autonomní týmy skládající se z lidí různých pozic, mající na starost jednu část systému. Tyto skupiny obvykle nejsou větší jak osm lidí a mají „end to end“ zodpovědnost za určitou funkcionalitu, tedy návrh, vývoj, provoz, údržbu a další zlepšování. Autonomie znamená, že každá četa si sama určuje co bude vyvíjet, jak toho dosáhnout a jakým způsobem bude organizována práce. Samozřejmě jsou zde určité hranice v podobě mise těchto skupin, produktové strategie a krátkodobých cílů, které jsou sjednávány každé čtvrtletí. Tato struktura zaručuje pružnější reakce z hlediska rozhodování a přehlednější organizační strukturu. Celé fungování by se dalo parafrázovat jazzovou kapelou, kde každý hudebník má naprostou volnost ale musí naslouchat ostatním členům, protože výsledná muzika kterou vyprodukují je důležitější než kterýkoliv jednotlivý člen.

Jednotlivé čety jsou dále seskupovány do „kmenů“ (tribes) což je lehká struktura seskupující čety s podobným zaměřením, například infrastruktura, backend nebo samotný hudební přehrávač (6). Uvnitř těchto struktur existují ještě „kapitoly“ (chapters), které seskupují pracovníky na stejných pozicích napříč kmeny. Takovéto sdružování podporuje efektivnější

sdílení informací a rozvoj v rámci daného zaměření, například front-end vývojáři mohou lépe konzultovat možnosti zapojení nové technologie, či projektoví manažeři diskutovat jakým způsobem zlepšit fungování jednotlivých čet.



Obrázek 2 – Grafické znázornění organizace, zdroj: Scaling Agile @ Spotify autor: Henrik Kniberg, Anders Ivarsson

10. Závěr

Cílem této práce bylo analyzovat a shrnout výhody a nevýhody agilního přístupu k vývoji softwaru, touto problematikou se zabývá kapitola 2 a 3, kde jsou popsány a vysvětleny jednotlivé aspekty. V další kapitole je na fiktivním příkladu ukázáno špatné použití agilní metodiky s následným vysvětlením proč tomu tak je a ukázka toho, jakým způsobem by se mělo ve stejné situaci postupovat, pro zajištění spokojenosti zákazníka.

Kapitoly 5 až 7 jsou věnovány hlavním aspektům agilního vývoje. Tedy lidským zdrojům a vlivu dobře poskládaného týmu na úspěšnost projektu, dále roli manažera, jakožto osoby, která udává směr a má velký vliv na konečný výsledek a v neposlední řadě funkci meetingů, které se v mnoha případech nepoužívají správným způsobem a tím pádem nemají požadovaný efekt.

Poslední dvě části této práce se zaměřují na službu a firmu Spotify, jako velice dobrý příklad, jak mohou výše zmíněné poznatky pozitivně ovlivnit ambiciózní a náročný projekt. Nejprve jsou představeny nelehké podmínky, ve kterých Spotify vznikalo společně se správným použitím prototypování z kapitoly 3, a ve druhé části je rozebrána unikátní agilní organizační struktura, kterou si společnost přizpůsobila na míru svým potřebám.

Bibliografie

1. **Knesl, Jiří.** Zdroják. *Agilní vývoj: Úvod.* [Online] 11. 12 2009. [Citace: 17. 12 2016.] https://www.zdrojak.cz/clanky/agilni-vyvoj-uvod/.

2. **Pavlas, Martin.** Think Forth. *Znásilněný Scrum.* [Online] 11. 12 2012. [Citace: 17. 12 2016.] https://blog.think-forth.com/2015/12/11/znasilneny-scrum/.

3. **Kniberg, Henrik.** Crisp. *Crisp's Blog.* [Online] 25. 1 2016. [Citace: 17. 12 2016.] http://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp.

4. **Ping, Chen.** InfoQ. *So, How Do You Make Agile Successful?* [Online] 21. 8 2015. [Citace: 17. 12 2016.] https://www.infoq.com/articles/how-make-agile-successful?utm_source=infoqWeeklyNewsletter&utm_medium=WeeklyNL_EditorialContent_culture-methods&utm_campaign=08232016news.

5. **Jordan Crook, Fitz Tepper.** A Brief History Of Spotify. *techcrunch.com.* [Online] 29. July 2015. https://techcrunch.com/gallery/a-brief-history-of-spotify/.

6. **Ingrid, Lunden.** Here's How Spotify Scales Up And Stays Agile: It Runs 'Squads' Like Lean Startups. *techcrunch.com.* [Online] 17. November 2012. https://techcrunch.com/2012/11/17/heres-how-spotify-scales-up-and-stays-agile-it-runs-squads-like-lean-startups/.

7. **Kniberg Henrik, Ivarsson Anders.** *Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds.* [Document] 2012.