Scaling Agile Across Project Teams and Departments

Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	ZS 2017/2018
Autoři	Vítězslav Štrajt, xstrv30 Roman Fausek, xfaur00 Anastasi Korjenevskaya, xkora20
Téma	Scaling Agile Across Project Teams and Departments
Datum odevzdání	17. 12. 2017

Obsah

Abstrakt

Tato práce je určena především pro čtenáře, kteří se účastní kurzu 4IT421 Zlepšování procesů budování IS na Vysoké škole ekonomické, kterým by měla pomoci s orientací v problematice podílení se více týmů na agilně řízených projektech. Zároveň je však práce použitelná i pro ostatní čtenáře zajímající se o agilní metodiky a témata s nimi spojená.

Klíčová slova

Agile, Scaling, Testování, Automatizované testování, Test driven development, Programování řízené testy, Agilní vývoj, Agilní testování, Agilní metriky, SAFe

Téma v kontextu budování IS

Vzhledem ke zvyšující se oblíbenosti agilních metodik je aktuálním tématem i způsob, jak rozšířit týmy pracující na větších projektech. Již z definice agilních přístupů vyplývá, že by počet členů v týmu neměl přesáhnout určitou hodnotu. Proto se tento problém řeší tím, že se na jednotlivých projektech podílí více menších týmů. Právě o tom pojednává i tato práce, a to včetně tzv. best practises i konkrétních metodik pro řízení distribuovaných týmů.

Cíle

Cílem této práce je rozdělení a zprostředkování přehledu o možnostech škálovaných projektů používajících agilní metodiky. Zprostředkovává hlavní myšlenky práce Scaling Agile Across Project Teams and Departments (Scaling Agile Across Project Teams, 2017) a rozšiřuje je o odkazy k souvisejícím tématům. To souvisí s druhým cílem, vytvořením rozcestníku odkazujícího na navazující témata probíraná v kurzu 4IT421.

Přístupy a postupy k dosáhnutí cíle

Pro splnění výše definovaných cílů je v této práci použit v první řadě dokument Scaling Agile Across Project Teams and Departments (Scaling Agile Across Project Teams, 2017). Pomocí tohoto dokumentu je tvořena kostra této práce, kdy části odkazující na tento dokument pojednávají o krocích, které je vhodné následovat a dosáhnout tak fungující agilní týmy. Co však v dokumentu chybí, jsou konkrétní metodiky, které se pro agilní přístup k projektům používají. Tento nedostatek kompenzuje práce tím, že stručně shrnuje a dále odkazuje na články vytvořené jinými skupinami studentů.

1. Úvod

Obecně se metodiky vývoje dělí na tradiční a agilní metodiky. V případě této práce budou popisovány principy týkající se jen agilních metodik. Budou tedy v úvodu této kapitoly popsány, aby čtenář měl přehled o specifikách těchto metodik a dokázal si lépe představit souvislosti s popisovanými doporučeními.

Agilní metodiky vývoje se snaží jen o definování principů vývoje bez toho, aniž by definovaly a popisovaly procesy, které se mají při vývoji dodržovat. Zároveň vývoj programů pomocí těchto metodik je dynamičtější, jelikož nedílnou součástí vývoje je komunikace se zákazníkem, kdy často validujeme vzniklé přírůstky a v případě potřeby zákazníka mohou být dříve definované funkcionality programu měněny. Tím je zaručeno, že má zákazník přehled o tom, co přesně je pro něj vytvářeno a svým pohledem dokáže pomoci dodavateli softwaru k tomu, aby výsledný produkt přinesl co nejvíce užitku. (Buchalcevová, 2009)

Dalším specifikem agilního vývoje je, že produkt je dodávaný zákazníkovi v kratších intervalech, než je běžné v případě tradičních metodik, například než u vodopádového modelu. Tím se pomalu dostáváme k obsahu hlavního článku, kde je zmíněna právě doba mezi jednotlivými uvolněními nových verzí k užívání zákazníkem.

Druhým bodem zmíněným v úvodu článku je dostupnost těchto produktových verzí z pohledu zákazníka, kdy je nutné sledovat kvalitu doručovaného softwaru a jeho dostupnost. Tato pasáž je dále rozepsána v druhé kapitole.

Třetí, poslední kapitola se zabývá měřením úspěšnosti ve firmě, které předchozí dva kroky aplikovala. Jsou zde popisovány metriky, pomocí kterých lze měřit úspěchy týkající se samotného vývoje aplikací a které by následně měli používat zaměstnanci na manažerských pozicích k tomu, aby se týmy pracující na agilně řízených projektech dále zlepšovaly.

2. Krok 1: Změna kultury

Za první a zásadní krok vedoucí k dosažení optimálních výsledků spojených s aplikací agilního přístupu k vývoji softwaru považuje autor článku Scaling Agile Across Project Teams and Departments změnu kultury v dané organizaci. Zmiňuje především chování a způsob práce týkající se vývojářů, testerů a analytiků. Naopak se článek nezaobírá ostatními rolemi, které se na projektu podílejí. Proto je v první části této kapitoly uveden názor autora na optimální fungování týmu s ohledem na tyto 3 jmenované role a v následující části budeme odkazovat na jiná díla, která doplňují pohled na distribuované týmy i o další nezbytné role.

2.1. Doporučení podle hlavního článku

Jak je již zmíněno, článek pojednává především o délce doby mezi vydávanými verzemi a složení týmů. V obou případech pojednává o faktorech přímo se podílejících na tvorbě softwaru.

2.1.1. Doba mezi vydáváním verzí

Podle článku, by se měla společnost vyvíjející software snažit o co nejkratší dobu mezi vydávanými verzemi vyvíjeného produktu. Je zde zmínka, že velké společnosti vydávají verze například po roce, nebo půl roce. V takovém případě by se ani nejednalo o agilní vývoj, doporučuje se tedy snížit frekvenci pod jeden měsíc, případně snížit frekvenci uvolnění nové verze na každé dva týdny. K tomu by měla dopomoci i struktura týmů, které na projektu pracují.

2.1.2. Three amigos

Změnou kultury je myšleno mnoho změn, týkajících se například vztahů mezi zaměstnanci vykonávajícími různé role na projektu. Důležitým bodem, který se týká vztahů zaměstnanců a organizační struktury je urovnání případných neshod mezi odděleními, která bývají obvykle samostatně lokalizovaná. Aby bylo snadnější vzájemné pochopení vývojářů, testerů a analytiků, je potřeba, aby se vzájemně respektovali a pomáhali si. Musejí pracovat jako tým, místo toho, aby spolu bojovali. Ke splnění tohoto cíle by mělo napomoci přesunutí členů týmu k sobě tak, aby spolu mohli snadno komunikovat a pracovat ve stejném prostředí. V textu je používán termín Three amigos, který lze volně přeložit jako „Tři kamarádi“, což je pro tento princip poměrně výstižné.

2.1.3. Spolupráce při odhadování pracnosti

Třetí odstavec se věnuje odhadování pracnosti a oklikou se vrací k předchozím dvěma odstavcům tím, že pojednává o výhodách, které plynou ze spolupráce vývojářů, testerů a analytiků. Díky tomu, že budou při odhadování pracnosti přítomni všechny z těchto rolí, mohou se ihned identifikovat možné problémy, které by se jinak objevily až při práci konkrétního člena týmu. Lze tedy předem navrhnout řešení, které bude přijatelné pro všechny zainteresované strany a tím se může zkrátit doba vývoje, snížit náklady na vývoj a přesně dosáhnout požadavku zákazníka, což je právě to, o co se agilní metodiky snaží.

2.2. Doplňující články

Bohužel se v článku Scaling Agile Across Project Teams and Departments neobjevuje žádný odkaz na metodiku, pomocí které by se daly tyto změny hodnot aplikovat na větší množství týmů, které pracují na jednom projektu. Aby bylo možné výsledky této práce použít v praxi při spolupráci více týmů, odkazuje tato kapitola na díla jiných skupin, které mají za úkol napsat o uznávaných metodikách používaných pro škálování vývojářských týmů.

2.2.1. SAFe 4.5

Jedním z řešení pro řízení agilně vedených projektů s více týmy, je metodika SAFe. Jedná se o metodiku, která dokáže podporovat rozsáhlé projekty, na kterých se podílí až tisíce osob. Stejně tak je ale možné použít tuto metodiku pro menší projekty. Podle webových stránek je menší tým definovaný jako tým s 50 až 125 členy.

Dále se SAFe dělí podle požadavků kladených na metodiku na Essential SAFe, Portfolio SAFe, Large Solution SAFe a Full SAFe. Všechny vlastnosti projektu, které by se měly brát v potaz při výběru z těchto čtyř možností lze opět dohledat na webových stránkách.

Hlavním přínosem a také důvodem, proč je zde tato metodika uvedena, je doplnění potřebných rolí při vývoji softwaru. Definuje roli ředitele vývoje a produkt manažera. To jsou alespoň z pohledu dodavatele nezbytné role, které dále řeší rozložení práce mezi týmy, složení týmů a požadované vlastnosti vyvíjeného produktu. (What is SAFe?, 2017)

2.2.2. The Disciplined Agile (DA) Framework

Dalším řešením za účelem zavedení řádu do distribuování práce mezi více týmů je metodika Disciplined Agile. Jedná se o lehkou metodiku používanou pro zpřehlednění vztahů mezi různými odděleními ve firmách obecně. Pro účely této práce, kdy je cílem držet se především samotného vývoje produktů, lze čerpat informace z části zvané Disciplined Agile Delivery, kde jsou opět definovány role, které jsou nutné pro vývoj v reálném prostředí. Dodatečné

role jsou: týmový vůdce, vlastník architektury a vlastník produktu. Role vývojáře, testera a analytika patří do čtvrté skupiny, do členů týmu. Díky tomuto rozdělení rolí v týmech získávají někteří zaměstnanci pravomoci a možnost komunikovat s jinými zaměstnanci, díky tomu lze domlouvat rozdělení práce a zodpovědnosti za požadované funkcionality. (Introduction to Disciplined Agile Delivery (DAD), 2017)

3. Krok 2: Procesní inženýrství

Tímto krokem vzniká snaha o vytváření krátkých, opakujících se cyklů změn. Změnou je myšleno jedna či více úprav služby IT, kterou společnost sestavuje, testuje a nasazuje najednou. Může se jednat o změny hardware, software, dokumentace, procesů a dalších komponent. (Bestpractice.cz 2017)

Momentem, kdy se týmová kultura změní směrem k lepší podpoře rychlého dodávání je tedy úkolem zajistit, aby se vše co nejvíce opakovalo, s čímž souvisí i zkrácení. K dosažení opakujícího se procesu je potřeba investovat peníze do procesního inženýrství. Cílem je kromě zlepšení kultury společnosti také investice do programů a metodik.

3.1. Automatizace dodávek produktů

Vytvoření automatizovaných dodávek produktů je prvním krokem v každém projektu s dodávkou software. Vytváří se za účelem odstranění lidské chyby a zajišťuje se tím určité bezpečí a vysoká kvalita kódu, který jej převádí do vhodného prostředí. Automatizací se zároveň vyřeší otázky související s auditem a dodržováním předpisů. Společnost implementací této metodiky získává svojí nejvzácnější a nejdůležitější vlastnost.

3.1.1. Software

Existuje relativně velký seznam aplikací, se kterými je možné dosáhnout daného cíle, ale mezi nejznámější patří nástroje jako TeamCity, Jenkins nebo Microsoft Team Foundation Server. Důležitým faktorem je vybrat nástroj, který má široké uplatnění v rámci softwarové komunity, nikoliv nástroj na úrovni „Enterprise-level“ s uplatněním úzkým.

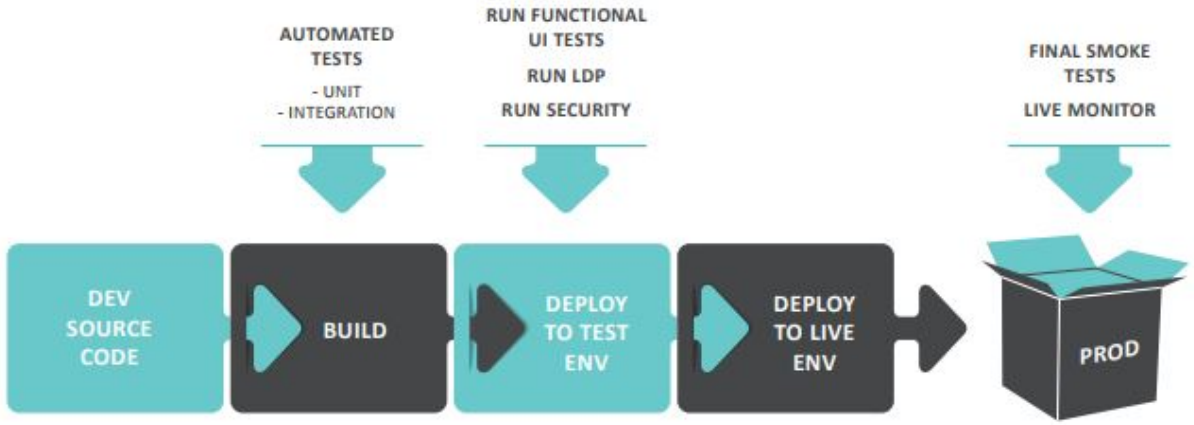
Každý tým v organizaci by měl mít určitou formu automatizovaných dodávek produktů. Využívaný software by měl zvládnout při nejmenším alespoň tyto prvky:

- Stáhnutí (pull) nejaktuálnější verze vyvíjeného software. Pro webové služby se nejčastěji využivá v kombinaci s GitHub nebo BitBucket softwarem.
- Build (sestavení) softwaru.
- Spuštění automatických testů dle potřeby
- Static code analysis nebo jiné nástroje na analýzu kvality kódu – analyzování kódu aniž by se aplikace pouštěla. Většinou se využívá k nalezení syntaktických, sémantických a chyb slovní zásoby. (The MathWorks 2017)
- Nasazení softwaru včetně dat do vývojového či ostrého prostředí.

- Spuštění automatizovaných testů ve fázi po nasazení (integrační, bezpečnostní, funkční a další). (Nikulina 2016)
- Ukládání artefaktů z nasazení (schéma databáze, binární soubory) včetně kvalitativních informací jako jsou výsledky testů.

Jednou z možností pro zlepšení kvality je spouštění zátěžových a výkonnostních testů v období, kdy je software nejméně používán. Nejčastěji tomu bývá v noci, ale to opravdu záleží na charakteru.

Rizikem těchto nástrojů je velká časová náročnost k nastudování. Stejně tak s tím souvisí, že by měl ve společnosti existovat někdo, kdo bude daný nástroj spravovat a monitorovat. Chyba může nastat i u špatné volby infrastruktury. Společnost by se měla ujistit, že je správně nastavené užívání lidí a zdrojů k správné podpoře těchto nástrojů.



Obrázek 1 - průchod dodávkou produktu

4. Automatizované testování kvality

K testování kvality je využíváno frameworků, nástrojů pro zátěžové a výkonnostní testy a nástroje, které umožňují static code analysis. Automatizovaných testů je využíváno hlavně z důvodu rychlé zpětné vazby na daný software. Díky testům je nám zprostředkována spolehlivá zpětná vazba, díky které je možno jednodušeji a bezpečněji vytvářet, upravovat či dále rozšiřovat codebase.

V případě, kdy je již agilní tým ve fázi vývoje a nevyužívá automatizovaných testů, měl by okamžitě začít plánovat. Existuje spoustu typů testů a u každého z nich je potřeba, aby bylo porozuměno tomu, jak by měly být správně implementovány k dosažení největší efektivity. Testy, které jsou vhodné k zvážení:

- Unit testy
- Testy integrace
- Testování externího systému
- Funkční/UI testování
- Testování zabezpečení
- Zátěžové a výkonnostní testování a další

Začít testovat je zprvopočátku velice náročná činnost. Vyžaduje to praktické zkušenosti, určitou znalost specifických vzorů softwarového designu a schopnost naplánovat funkční implementaci automatizace, která se zaměří na aspekty s největší hodnotou a zároveň se protne s cíli dlouhodobé udržovatelnosti. (Hlava 2011) V případě, že tým nemá zkušenost s takovouto implementací, tak stojí za zvážení sehnat někoho ze společnosti, kdo se nenachází v daném agilním týmu, či někoho externího.

Největším rizikem automatizovaného testování jsou špatně napsané testy. Platí zde pravidlo, že špatné testy jsou mnohem horší jak testy žádné a to z jednoduchého důvodu – vznikají v nich bezdůvodné chyby nebo naopak chyby nevznikají i přesto, že by měly. Takové testy zabírají velký čas na opravu a snižují důvěru v týmech. Zmírnit se to dá právě tím, že jednotlivé agilní týmy budou mít členy či vedoucí s praktickými zkušenostmi.

Hodnocení výkonnosti zvýhodňuje jak zaměstnance, tak zaměstnavatele. Jak již bylo zmíněno tak poskytuje rychlou zpětnou vazbu, rozpoznává kvality výkonnosti a pomáhá stanovit očekávání pro budoucí pracovní výkon. Diskuze o výkonu může týmům a společnostem pomoci vyhnout se vážným problémům v budoucnosti. Více o těchto skutečnostech je možné se dočíst v semestrální práci na téma „Are Unit Tests Part of Your Team's Performance Reviews?“

5. Zvoleni a prijeti vhodne formy Test-First Development

Test-first development neboli programování řízené testy má mnoho podob, například:

- TDD: Test Driven Development
- BDD: Behavioural Driven Development
- ATDD: Acceptance Test Driven Development

Bez ohledu na to, jaká forma by byla zvolena, podstatou tohoto přístupu je definice funkcionality a následné napsání testů, které tuto funkcionalitu ověřují. Poté přichází na řadu psaní kódu a nakonec úprava tohoto kódu.

Bez ohledu na konkrétní formu nabízí tento přístup řadu konkrétních a měřitelných výhod:

- Téměř kompletní pokrytí automatizovanými testy.
- Kompletní sadu automatizovaných regresních testů, které vytvářejí obrovskou důvěru ve schopnost týmu pracovat rychle při vývoji a aktualizaci systému.
- Mnohem jasnější a čistší design systému.

Dosáhnout vysoké úroveň kompetencí v programování řízené testy může být mimořádně obtížné pro tým, který se tu metodiku snaží naučit sám. Úspěšné přijetí tohoto přístupu vyžaduje extrémní kázeň a odpovědnost napříč celým týmem. Dále pak značné množství zkušeností, neustálou zpětnou vazbu a schopnost se učit s předchozích. Dle (Scaling Agile Across Project Teams, 2017) existují tyto osvědčené způsoby, jak zvýšit dovednost týmů na nezbytnou úroveň:

Začínající tým (žádné dovednosti v přístupu Agile nebo automatizovanými testy):

- Najít externího trenéra, aby se zapojil do týmu a provozoval nezbytná školení.
- Přivést zkušeného odborníka z jiného týmů, ideálně z interních zdrojů.
- Kombinovat trenéra s vhodnou formu e-Learningu.

Mírně zkušený tým (předchozí zkušenosti s Agile a automatizovanými testy):

- Doplnit tým zkušenými odborníky, ať už z jiných týmů uvnitř organizace nebo pozvat externí.
- Přivést trenéra, který bude pomáhat týmu v praxi.
- e-Learning

Pokročilý tým (zkušený v Agile a automatizovaných testech)

- Posílit tým získáním správných nástrojů.
- Razit mentalitu, že použiti programování řízené testy je nutnost.
- Poskytnout další podporu prostřednictvím trenérů, externích odborníků a dalších.
- Ujistit se, že tým získává prostor pro učení a neustálé zlepšování.

Pokud organizace nikdy předtím nepoužívala programování řízené testy, je potřeba nejdříve začít osvojením metodiky vzděláváním členů týmu. Okamžitě sledovaní metrik výkonu poskytne rychlou zpětnou vazbu o tom, jak rychle tým TFD přijímá a je tak možné zjistit, jak moc si tým metodiku osvojil.

Časový tlak a lidská přirozenost mohou způsobit, že místo dopředu se projekt posouvá spíše zpět. Zachování disciplíny nezbytné pro programování řízené testy sice není vždy snadné, ale pokud se podaří dosáhnout neměnné a jasné řady procesů, budou správné inženýrské postupy fungovat ještě dlouho.

5.1. Kvalifikovaní testeři

plňkem k dobrým vývojářům a automatizovaným testům. Kvalifikovaní testeři přinášejí do projektu zcela odlišný pohled na projekt, než jakým disponují vývojáři. Skvělí testeři pomáhají týmu zaměřit se na nejdůležitější části softwaru, ať už z hlediska obchodní hodnoty nebo z hlediska uživatelské přívětivosti. Dokáží vytvářejí realistické zkušební scénáře, chápou, jak zabránit překrývání požadavků a kladou otázky, které by byly jinak přehlíženy.

Hledání profesionálních testerů je stejně obtížné, jako hledání vývojářů. Je dobré vychovávat a investovat do mladších, méně kvalifikovaných testerů nebo do těch, kteří mají určitou úroveň schopností a zájmů. Testeři potřebují pro výcvik dovedností a neustálé učení stejný prostor, jako vývojáři.

Získání špatného testera v týmu je stejně velký problém, jako získání špatného vývojáře. Špatní testeři se zaměřuji na případy, které nejsou realistické a výrazně neovlivňují celkovou kvalitu produktu. Soustřeďuji se na hledání chyb, místo toho, aby pomohli týmu předcházet chybám.

6. Krok 3: Zlepšovaní procesu dodaní SW

Posledním krokem je snaha o nepřetržité dodávání nových verzí produktu. Časté dodání projektu vytváří několik výhodných situací. Lze díky nim identifikovat překážky, které musí být vyřešeny, aby se mohlo hladce pokračovat. Častější a menší změny pomáhají také v jednodušším řešení problémů, které se mohou objevit při celkovém nasazení. Přístupem krok-po-kroku se totiž vždy začne s tím největším problémem.

Inkrementální nasazování softwaru nutí členy týmu se soustředit na malý rozsah. Díky tomu mohou dosahovat vysoké kvality práce a výsledků. Díky tomu je tým bude reagovat agilně na potřeby zúčastněných stran a zákazníků. Výhodné je to i z hlediska motivace, kdy jsou členové organizace šťastni, jak rychle byly schopni opravit chyby, přenést nové funkce a reagovat na měnící se trhy.

Na druhou stranu ale mohou být týmy posedlé rychlým vydávaným produkce za cenu stability a reprodukovatelnosti. Hlavním cílem je zaměřit na vytvoření hladkého procesu a až potom pracovat na jeho urychlení. Častá dodávka softwaru může být složitá i pro prodejní, podpůrné a obchodní oddělení nebo dokonce i samotné zákazníky. Když všechny tyto skupiny zůstanou těsně integrovány do procesu uvolňování produkce, riziko problému bude sníženo.

Jako u všech agilních přístupů je i zde kladen velký důraz na komunikaci. Je potřeba zajistit vedení příslušné dokumentace, organizovat školení a podporovat aktivity související s vylepšením komunikace.

6.1. Posun k nepřetržitému dodaní

Konstantní zpětná vazba je základním aspektem agilního přístupu, proto je měření úspěchu při pokročilé agilní transformaci zásadní aktivitou. Následná analýza výsledků měření a její zápis do nástrojů, kultury a procesu znamená, že nejen tým, ale i celá organizace je schopna se posunout k nepřetržitému doručení a tím zajistit nepřetržitý tok přidané hodnoty všem zúčastněným stranám.

7. Sledovaní základních metrik

Pečlivý výběr metriky je základním požadavkem pro poskytnutí spolehlivých informací, která pak společnost může uplatnit při dalším rozhodování. Výběr nesprávné metriky může mít za následek špatnou interpretaci informací a tím i nesprávné rozhodování, což může vyústit v další problémy uvnitř organizace. Zralé organizace chápou, že metriky jsou skvělým ukazatelem trendů, které však neslouží jako diskrétní měření.

Níže je uvedeno několik metrik, které jsou dle (Scaling Agile Across Project Teams, 2017) pro týmy velmi nápomocné pro měření zralosti agilních procesů:

Work Item Size

Týmy obvykle vytvářejí pracovní položky, které jsou příliš velké - trvají týdny nebo dokonce měsíce. Zralé týmy rozděluji pracovní balíky na mnohem menší části - kratší než týden, nejlépe na jeden nebo dva dny.

Throughput

Tato metrika měří kolik stories nebo pracovních položek tým udělat od začátku do konce iterace. Její zlepšení je ukazatelem, že mnoho dalších procesů funguje dobře. Pracovní balíky jsou správně dimenzovány, cyklus vývoj/test/nasazení probíhá bez problémů.

Cycle Time

Doba trvaní pracovní položky od začátku do konce. Jedná se o přímý faktor závislosti na přechozí metrice a označuje místa, kde proces neprobíhá hladce nebo jsou omezeni lidé a zdroje.

Commitment / Completion Ratios

Je ukazatel toho, jestli týmy mají více práce, než mohou zvládnout. Pokud ano, mají týmy snížit svůj závazek. Mělo by být použito jak pro story points, tak i pro stories.

7.1. Retrospektiva

Dle (Scaling Agile Across Project Teams, 2017) jsou retrospektivy jednou z nejlepších aktivit, které lze provozovat při agilní transformaci v organizaci. Konstantní zpětná vazba je rozhodující pro co nejsnadnější agilní transformaci. Dále je také důležitá pro pokračující agilní operace, jakmile transformace je považovaná za kompletní.

8. Závěr

Velké organizace mají hned několik problémů, které musí vyřešit pro zavedení agilních metodik. Pomalá komunikace, zdlouhavé standardy, procesy a kulturní setrvačnost může oslabit i ty nejlepší záměry. Řešení těchto problémů vyžaduje monitorování, zpětnou vazbu, přizpůsobení a zapojení všech členů organizace.

Bez ohledu na to, které praktiky se organizace rozhodnou implementovat, je pro úspěšnou agilní transformaci rozhodující jedna zastřešující základní teorie: disciplína a neustálé zlepšování. Organizace, které usilují o zavádění agilních metodik, musí být k sobě upřímné a učit se z vlastních chyb, které jsou v tomto procesu nevyhnutelně.

Agilní transformační cesta každé organizace je odlišná a ignorování získaných poznatků cestu nijak neulehčuje. Organizace, které začínají s jasnými očekáváními kolem podporujícího a povzbudivého vzdělávacího prostředí, budou mít rychlejší a plynulejší transformaci. Tyto organizace budou mnohem lépe překonávat problémy spojené s komunikací a setrvačností, které trápí velké organizace, snažící se o změnu své kultury a obchodních procesů.

9. Zdoje

BESTPRACTICE.CZ, 2017. *Release and deployment management | bestpractice.cz* [online]. Dostupné z: https://www.bestpractice.cz/cs/Best-practice/-ITSM-ITIL-/Klicove-procesy-ITIL-/Release-and-deployment-management.alej

BUCHALCEVOVÁ, Alena. Metodiky budování informačních systémů. Vyd. 1. Praha: Oeconomica, 2009. 205 s. Vysokoškolská učebnice.ISBN 978-80-245-1540-3.

HLAVA, Tomáš, 2011. *Integrační Testování | Testování softwaru* [online]. Dostupné z: http://testovanisoftwaru.cz/tag/integracni-testovani/

Introduction to Disciplined Agile Delivery (DAD). The Disciplined Agile (DA) Framework [online]. Dostupné z: http://www.disciplinedagiledelivery.com/introduction-to-dad/

NIKULINA, Irina, 2016. The Three Phases of the Deployment Testing Cycle. *Master of Code Global* [online]. Dostupné z: https://masterofcode.com/blog/three-phases-deployment-testing-cycle

Scaling Agile Across Project Teams and Departmen. Zephyr [online]. Dostupné z: https://www.getzephyr.com/resources/whitepapers/step-step-guide-scaling-agile-across-project-teams-and-departments

THE MATHWORKS, 2017. *Static Code Analysis - MATLAB & Simulink* [online]. Dostupné z: https://www.mathworks.com/discovery/static-code-analysis.html

What is SAFe? SAFe [online]. Dostupné z: http://www.scaledagileframework.com/what-is-safe/

10. Seznam obrázků