Vysoká škola ekonomická v Praze

Škálování SCRUM

Martin Světlík (xsvem00)

Ondřej Kaushik (xkauo00)

15. 5. 2015

Obsah

1 Úvod

1.1 Cíle práce

V rámci semestrální práce na téma Škálování SCRUM je hlavním cílem krátce představit metody, které se škálování věnují, a dále se blíže zaměřit na framework LeSS (Large-Scale Scrum), včetně uvedení jeho principů, implementace, řízení atp. a také představení případových studií. Součástí by mělo být i zhodnocení LeSS a jeho použitelnosti v praxi.

1.2 Důvody potřeby škálování SCRUM

SCRUM jako agilní metodiku pro iterativní a inkrementální vývoj software netřeba dlouze představovat. Než si představíme obecné důvody pro škálování, je více než vhodné si definovat pojem škálovatelnost. Škálovatelnost je schopnost systému, či procesu obsáhnout rostoucí počet objektů, zvládnout zvyšující se objem práce, rozšíření o dodatečné komponenty či mít k takovému rozšíření předpoklady [1].

Při škálování agilních metodik tak dochází k individuálním úpravám jednotlivých praktik a přidávání nových praktik s ohledem na zvětšený objem objektů v projektu [1]. Agilní metodiky obecně jsou určeny pro malé a lokální týmy, ve větší organizaci tak vzniká potřeba škálovat.

2 Představení dostupných metod škálování SCRUM

2.1 Metody škálování

Způsobů škálování existuje několik, pro potřeby této práce představíme následující:

- Scrum of Scrums
- Scaled Agile Framework (SAFe)
- Disciplined Agile Delivery (DAD)
- Large-Scale Scrum (LeSS)

Samostatnou část jsme věnovali frameworku LeSS, který jsme zvolili jako nejzajímavější, ostatní budou pouze krátce představeny. Některým z nich se pak věnují samostatné práce, nebudeme tedy zacházet do podrobností jako v případě LeSS.

Peter Stevens, certifikovaný a zkušený kouč, trenér a mentor s vášní pro pomoc organizacím k transformaci do 21. století, jak je uvedeno v jeho profilu na Scrum Alliance [2], publikoval na svém blogu *scrum breakfast* článek *Scaling Scrum: SAFe, DAD, or LeSS?* [3] Ten bude základním východiskem při představení jednotlivých metod škálování a pro jejich srovnání.

2.2 Scrum of Scrums

Scrum of Scrums je poměrně jednoduchý způsob a funguje jako důležitá technika škálování pro větší projektové týmy. [4] V praxi tato technika znamená rozdělení projektového týmu na menší agilní týmy o 5 – 10 členech. Každý daily scrum v rámci těchto menších týmů končí určením jednoho člena jako ambasadora, který se zúčastní daily meetingu v rámci ostatních týmů. Nutno podotknout, že ambasador nemusí být nutně Scrum Master.

Scrum of Scrums pak probíhá jako standardní daily meeting, kdy ambasadoři reportují dokončené úkoly, další kroky a překážky v zastoupení za tým, který reprezentují. Důležitou součástí je koordinace mezi samotnými týmy a používá se samostatný backlog, kde každá položka přispívá ke zlepšení. [5]

2.3 Scaled Agile Framework (SAFe)

SAFe si klade za cíl implementovat agilní praktiky v měřítku velkých společností. Společnost pak rozděluje na tři úrovně: Team, Program a Portfolio. [6] Na úrovni týmu se objevují prvky známé ze Scrumu, tak současně zahrnuje i osvědčené postupy extrémního programování. Program a Portfolio jsou určeny pro fungování v rámci všech úrovní organizace a můžeme v nich najít i další techniky, jako například Kanban.

Jak podotýká i Peter Stevens [3], na webu SAFe lze nalézt tvrzení, že Lean principy, The Principles of Product Development Flow a výhody agilního vývoje (Agile Manifesto, Scrum praktiky, Kanban) hrají významnou roli při stanovení principů a praktik SAFe [7]. Jako hlavní hodnoty však SAFe uvádí Alignment (Postavení), které považuje za důležité pro vtahy ve velké organizaci, Code Quality (Kvalita kódu), Transparency (Transparentnost) a Program Execution. Těmto hodnotám lze také rozumět tak, že vývojové týmy budou nadále pracovat na základě rozhodnutí managementu.

Leč se objevují od odborné veřejnosti skeptické názory na SAFe, tak nutno podotknout, že stále kombinuje Scrum a Kanban pro pokrytí potřeb větších týmů v rámci velkých společností, pro které navíc definuje úrovně a principy pro všechny části organizace, včetně vrcholového managementu. Umožňuje také inkrementální implementaci.

2.4 Disciplined Agile Delivery (DAD)

Na toto téma je určena samostatná práce, proto jej nebudeme podrobně rozebírat. Disciplined Agile Delivery vyvinul Scott Ambler při jeho působení v IBM. Už jen kombinace názvu a faktu, že vznik spadá k IBM, může vyvolat určitou skepsi. Dle jejich vlastní definice je DAD framework pro zjednodušení procesního rozhodování při dodávce IT řešení, který je zaměřen na lidi, podporuje učení a je hybridní [8].

Slovem hybridní v tomto případě rozumíme kombinaci jak agilních způsobů (Scrum, XP, Kanban…), tak i těch tradičních. Ostatně DAD rozděluje projekty mezi tři fáze: Inception, Construction and Transition, které můžeme znát z metodiky RUP.

Oproti standardnímu Scrumu se objevují také nové role a používají se jiné názvy, například Scrum Master je v tomto případě Team Lead. Oproti klasickému Scrumu také klade větší důraz na architekturu a snižování technických rizik.

3 Framework LeSS

3.1 Úvod

Pro hlubší analýzu do naší práce jsme si vybrali framework LeSS. Ten byl představen tvůrci Basem Voddem a Craigem Larmanem v roce 2013, jako výsledek dlouholeté práce na škálování agilní metodiky vývoje Scrum. Při návrhu celého frameworku došli autoři ke třem zásadním rozhodnutím, které následně ovlivnily celý výsledný framework.

- LeSS musí být jednoduchý
- LeSS je vyškálovaný Scrum
- LeSS se škáluje nahoru, místo zeštíhlování

Tyto počáteční pravidla byla zvolena z velmi jednoduchých důvodů. Požadavek na jednoduchost LeSS vychází z toho, že při škálování nahoru obecně vznikají tendence přidávat role, procesy a podobně. Což je však špatná praktika, neboť takto vznikají zbytečné objekty. Naopak všechny objekty by měly být přidány až ve chvíli, kdy vznikne potřeba jejich existence. S tímto bodem je úzce spojena další premisa, že LeSS je vyškálovaný Scrum. Autoři nechtěli používat Scrum jako stavební prvek jejich nového frameworku. Jejich myšlenkou bylo skutečné škálování – postupně projít všechny prvky klasického Scrumu, zjistit jejich účel a prozkoumat jak by mohlo být jejich funkce dosáhnuto v případě, že na produktu pracuje více než jeden tým. A stejně tak je s prvním pravidlem spojeno pravidlo druhé, které říká, že LeSS se vždy musí škálovat nahoru. Problém, který se totiž při nasazování metodik do firem často objevuje, je to, že firmy mají pocit, že potřebují nasadit veškeré prvky metodiky. Což zapříčiňuje neúměrný růst procesu vývoje a zahlcení pracovníků zbytečnými formalitami.

Výsledkem práce obou pánů je tedy LeSS framework, respektive LeSS, který je určen pro až 8 týmů, každý s až 8 lidmi, čili až 64 lidí celkem a nadstavbu pojmenovanou LeSS Huge, která základní LeSS rozšiřuje o prvky, se kterými lze úspěšně řídit vývoj jednoho produktu, na kterém pracuje až několik tisíc lidí.

K oběma těmto částem byla uveřejněna zdarma rozsáhlá dokumentace na webových stránkách projektu [9]. Dokumentace obsahuje nejen základní principy,

pravidla, návody a pomůcky, ale také mnoho případových studií, které popisují nasazení LeSSu ve firmách. Ačkoliv obě části frameworku obsahují striktně definované praktiky, je u každé části frameworku autory záměrně ponechán dostatek prostoru, který uživatelům dovoluje upravit framework na míru jejich potřebám. K tomuto autoři také nabádají a tvrdí, že pro maximální výkonnost je nutné pochopit základní pravidla a principy v kontextu celého frameworku a následně tyto pravidla rozbít a přizpůsobit svým potřebám.

Struktura samotného LeSSu se skládá ze 4 částí, přičemž každou jednotlivou část rozebereme v dalších subkapitolách

- LeSS pravidla
- Objekty LeSS frameworku
- LeSS principy
- LeSS návody

Při psaní této semestrální práce jsme se potýkali s problémem, zda překládat názvy rolí a jiné termíny, které se vyskytují v rámci LeSSu a jsou většinou přejaté z metodiky Scrum. Protože ale u čtenářů této práce předpokládáme znalost standardního Scrumu, rozhodli jsme se, po zvážení, ponechat je v původním anglickém znění a doplnit vysvětlivky pouze tam, kde by mohl nastat problém s porozuměním.

3.2 LeSS pravidla

Pravidla LeSSu shrnují základní teze, na kterých je framework vybudován a stejně jako ostatní části jsou maximálně stručná. Autoři je označují jako nutnou součást každého nasazení frameworku. Pravidla se obecně dělí do dvou skupin – pravidla pro LeSS a pravidla pro LeSS Huge. Avšak pro LeSS Huge platí i pravidla LeSSu, pokud není uvedeno jinak. Obě tyto skupiny autoři dále dělí podle jejich předmětu a to na pravidla vztahující se ke struktuře, produktu a sprintu. Kompletní pravidla v aktuálním znění lze nalézt na stránce http://less.works. Pravidla z dubna 2015 byla ve stručném znění takováto *(Následuje volný překlad autora):*

3.2.1 Struktura LeSS

- Strukturujte organizaci za pomoci skutečných týmů jakožto základní stavební jednotky

- Každý tým řídí sám sebe, je multifunkční [1], umístěn společně s ostatními a dlouho trvající
- Většina týmů je zaměřena na doručování funkcí pro zákazníka
- Scrum masteři jsou zodpovědní za dobře funkční organizaci
- Scrum master je role za plný úvazek
- Scrum master může pracovat s jedním až třemi týmy
- Manažeři jsou volitelný prvek LeSS, pokud existují, tak se zaměřují na rozvoj možností vývojového systému než na produkt samotný
- Pro vývoj produktu ustanovte kompletní strukturu LeSS od samého počátku
- Pro větší organizaci mimo vývoj produktu přijměte "Go and See [2]" jako normu pro zlepšování produktu

3.2.2 LeSS Produkt

- Existuje pouze jeden Product owner a jeden Product backlog pro celý produkt
- Product owner by neměl pracovat sám na Product backlogu
- Veškeré změny priorit prochází přes Product ownera
- Existuje pouze jedna, sdílená, definice hotového produktu
- Každý tým může mít svojí rozšířenou definici hotového produktu
- Definice produktu by měla být zpočátku širší a tvořena z pohledu zákazníka či uživatele
- Cílem je vylepšit definici hotového produktu tak, že každý Sprint je dodáno dodatelné zlepšení

3.2.3 LeSS Sprint

- Existuje pouze jeden Sprint na úrovni produktu. Sprint každého týmu začíná a končí společně s ostatními
- Plánování sprintů je dvoufázové. První fáze je společná pro všechny, druhá zpravidla pro každý tým zvlášť
- První fáze se účastní Product Owner a zástupci týmů
- Každý tým má svůj vlastní Sprint Backlog

[1] Jeho členové nemají jednu společnou specializaci
[2] Vysvětleno dále v kapitole o LeSS Návodech

- Během fáze dvě si každý tým rozhoduje, jak budou realizovat vybrané položky.
- Každý tým má svůj vlastní Daily Scrum
- Spolupráce mezi týmy je organizována týmy
- Rafinaci Product Backlogu[3] provádí každý tým
- Existuje pouze jeden Sprint Review, společný pro všechny týmy
- Každý tým má svojí, individuální, retrospektivu Sprintu
- Celková retrospektiva se uskutečňuje až po týmových retrospektivách

3.2.4 LeSS Huge – Struktura

- Požadavky zákazníka, které jsou z pohledu zákazníka spojené, jsou rozděleny do Requirement Areas[4]
- Každý tým se specializuje na jednu RA
- Na každou RA se specializuje 4-8 týmů
- Nasazení LeSS Huge je provedeno evoluční cestou

3.2.5 LeSS Huge – Produkt

- Každá RA má svého Product ownera
- Existuje jeden celkový Product owner, který je zodpovědný za celkové priority a rozdělování týmů do RA.
- Product owneři jednotlivých RA se chovají jako Product owneři ke svým týmům
- Existuje pouze jeden Product backlog, každá položka v něm patří přesně do jedné RA
- Pro každou RA existuje jeden Area Product backlog, který je podrobnější než celkový Product backlog

3.2.6 LeSS Huge – Sprint

- Existuje pouze jeden Sprint, neexistují rozdílné sprinty pro každou RA
- Celkový Product owner a jednotliví RA Product owneři úzce spolupracují

[3] V originále Product Backlog Refinement

[4] Dále RA. Česky Oblasti požadavků

- Mimo standardních Review a Retrospektivy se organizují také na RA úrovni

Tyto základní jednoduchá pravidla jsou tedy základním kamenem celého frameworku, jehož strukturu detailněji rozebírá následující kapitola.

3.3 Objekty LeSS frameworku

V této kapitole se budeme věnovat samotné skladbě LeSSu, především tomu, jak se jednotlivé role a procesy změnily v porovnání s klasickým Scrumem. Vzhledem k rozsahu této semestrální práce není možné podrobně popsat všechny objekty, budeme se proto věnovat pouze těm hlavním.

První rolí, která doznala zásadní změny je Product owner. Stále se jedná o jednoho člověka, což umožňuje neustále udržovat zaměření na ucelený produkt v průběhu celého vývoje. Hlavním úkolem Product ownera je určování priorit jednotlivým položkám z Product backlogu, druhou rolí je pomoc při vyjasňování jednotlivých položek, zde však vystupuje v roli jakéhosi mediátora, který zprostředkovává kontakt mezi uživateli produktu a týmy. Rozhodně by však jeho rolí neměl být přenos samotných požadavků, pouze zprostředkování kontaktu a podporování celého dialogu. Během celého vývoje musí Product owner udržovat 5 základních kontaktů, aby mohl být výsledný produkt zdárně dokončen. Jsou to kontakty mezi ním a týmy; mezi ním a zákazníky; mezi týmy a zákazníky; mezi ním a vyšším managementem a v neposlední řadě také mezi ním a Scrum mastery. To vše by mu však během standardního dvoutýdenního sprintu nemělo zabrat více než 8 hodin týdně. Neboť první fáze plánování sprintu by neměla přesáhnout 1 hodinu, rafinace Product backlogu běžně trvá 4 hodiny, Review a Retrospektivy většinou zaberou dohromady 3,5 hodiny.

Dalším prvkem, jímž se budeme zabývat je Product backlog. Autoři LeSS Frameworku tvrdí, že pro úspěšné dokončení produktu je zapotřebí mít pouze jeden Product backlog, který obsahuje veškeré úkoly, které se musí v souvislosti s produktem udělat, žádný tým nemá svůj vlastní backlog a stejně tak nejsou žádné úkoly přiřazované týmům dopředu. Současně však u každého úkolu musí existovat odhad pracnosti, úkoly s vysokou prioritou musí být přesně definovány a vůbec musí existovat priority.

Jednou z největších změn, které je možno oproti standardnímu Scrumu pozorovat, jsou změny Sprintu. Samozřejmě, že je stále cílem každého sprintu dodat potencionálně dodatelné zlepšení produktu. Jak již bylo řečeno v Pravidlech, plánování Sprintu se odehrává ve dvou fázích. V první fázi se setkají zástupci týmů s Product ownerem a vytvoří plán na další Sprint. Autoři frameworku důrazně upozorňují, že v případě většího počtu týmů je vhodné omezit počet účastníků, ideálně dva zástupce za každý tým. V první části plánování nadcházejícího Sprintu Product owner přednese požadavky, které se mají během Sprintu zpracovat a následně si je týmy rozdělí mezi sebe. Pak následuje diskuze nejasností a plánuje se případná spolupráce týmů. Po té přichází na řadu fáze dvě, která probíhá individuálně pro každý tým, pouze v případě potřeby těsnější spolupráce mohou týmy provádět tuto fázi společně. Během fáze dvě si každý tým vytvoří svůj vlastní Sprint backlog, se kterým pracují v průběhu nadcházejícího Sprintu. Dále je možné pořádat individuální schůzky mezi týmy, pomocí kterých se organizuje spolupráce. Tyto schůzky jsou plánovány podle potřeby i několikrát týdně.

Oproti standardnímu Scrumu nebyl takřka změněn princip Daily Scrumů, neboť jsou pro každý tým individuální a každý člen odpovídá na stejnou sadu tří otázek[5]. V případě, že práce na produktu vyžaduje užší spolupráci dvou týmů, mohou se zástupci jednoho týmu účastnit Daily Scrumu druhého týmu.

Podobným principem byl upraven Sprint Review, ten autoři vidí jako příležitost pro setkání zákazníků, Product ownera a týmů, aby prozkoumali výsledky práce týmů a prodiskutovali nové nápady či změny. Pro celý Sprint review je podle autorů dobře použitelná metoda Tržiště, při které zástupci týmů prezentují svou práci.

Po uskutečnění Sprint review následují týmové retrospektivy, které jsou shodné s retrospektivou standardního Scrumu, pouze s přídavkem toho, že týmy diskutují i návrhy změn v organizaci procesu vývoje. Celý Sprint je uzavřen společnou retrospektivou, které se účastní Product owner, Scrum masteři, zástupci týmů či přímo jejich členové.

Zhruba vprostřed Sprintu autoři doporučují dělat proces rafinace backlogu. Ten je opět rozdělen na dvě hlavní části – společnou rafinaci a týmovou rafinaci. Společná

[5] Na čem jsem pracoval včera?; Na čem budu pracovat dnes?; Jaké jsou překážky mojí práce?

fáze má za cíl rozdělit velké položky z backlogu, provést jednoduchou analýzu nutnou k základnímu pochopení problému, odhadnout pracnost a identifikovat spojené položky, které budou vyžadovat kooperaci mezi týmy. Na týmové úrovni, která opět může proběhnout i za účasti členů z jiného týmu, vyžadují-li položky backlogu spolupráci, proces probíhá stejně jak v případě standardního Scrumu.

A poslední rolí, kterou v této kapitole zmíníme je role Scrum mastera. V pravidlech LeSS frameworku existuje požadavek, že Scrum master je práce na plný úvazek. To z jednoho prostého důvodu - při škálování vzniká problém, jehož hlavním příznakem je ztráta soustředění na ucelený produkt ze strany vývojářů. Hlavním úkolem Scrum mastera je toto soustředění udržovat, i nadále však musí být prostředníkem k transparentnosti procesu vývoje celého produktu, protože toto autoři označují jako jeden z klíčových principů úspěšného vývoje. Další povinnosti Scrum mastera jsou shodné s jeho rolí tak, jak ji známe z klasického Scrumu.

3.4 LeSS principy

Minimalistická pravidla LeSSu definují samotný v framework, ale neposkytují žádné rady, jak framework nasadit do nějaké konkrétní organizace. To je úkolem takzvaných principů, kterých je rovných deset a které jsou vyjmenovány níže. U každého principu se pokusíme shrnout jeho základní myšlenku.

- **Scrum ve velkém měřítku (LeSS) je Scrum** *(Large-scale Scrum is Scrum)* – Dle autorů frameworku není LeSS nový a vylepšený Scrum, ani o framework, jehož základem je Scrum na týmové úrovni a následné škálování. Jde skutečně o Scrum, který má všechny objekty vyškálované.
- **Transparentnost** *(Transparency)* – Pro úspěšné dokončení produktu je nutné, aby každý tým viděl, jak dobře na produktu pracuje a napříč celým procesem vývoje panovala naprostá transparentnost, prosta jakéhokoliv zatajování informací či politikaření.
- **S LeSSem více** *(More with LeSS)* – Tento princip se skládá z pěti komplementárně spojených dílčích principů, prvním z nich je empirické řízení procesu (viz dále v této kapitole) – více učení a adaptace s méně definovaným procesem. Dále větší škálovatelnost s méně objekty (role, procesy) v porovnání s jinými přístupy ke škálování. Pak ještě více agility s

menší komplexitou, s tím je propojeno více vytvořené hodnoty s méně odpadem. A nakonec: Více výstupů s méně vstupy.

- **Zaměření na celý produkt** *(Whole Product Focus)* – Zákazníci si nechtějí koupit dokonalou část produktu, ale celý produkt. Což znamená, že části, které nejsou ještě součástí produktu, prozatím nemají žádnou hodnotu. Tým, který dokončil určitou část, není hotov, dokud ona část není integrována do výsledného produktu.
- **Orientace na zákazníka** *(Customer Centric)* – S ohledem na to, že zákazník je ten kdo výsledný produkt bude používat a ve velké části případů ho i zaplatí, musí mít vývojáři stále na paměti, že pracují na produktu, který nebudou používat oni, ale zákazníci.
- **Neustálé zlepšování na cestě k dokonalosti** *(Continuous Improvement Towards Perfection)* – Neustálé zlepšování není jen pilířem Lean přístupu, ale také principem LeSSu. Autoři doporučují neustále organizaci a produkt sledovat, měřit a snažit se odhalit místa, která mohou být zlepšena.
- **Štíhlé (Lean) myšlení** *(Lean thinking)* – Ačkoliv byl Lean přístup vymyšlen v Toyotě, jeho uplatnění je velmi široké a nalézá své místo i v procesu škálování Scrumu. Bohužel není možné se v rámci této práce věnovat podrobnému popisu tohoto přístupu. Avšak s aplikací tohoto přístupu se dá dosáhnout vyšší kvality výstupů za současné redukce vstupů.
- **Systémové myšlení** *(Systems Thinking)* – Použití systémového myšlení musí prostupovat celým procesem vývoje produktu, je žádoucí aby i řadoví vývojáři měli zájem naučit se více o tématech spojených s celým systémem, který obklopuje výsledný produkt.
- **Empirické řízení procesu** *(Empirical Process Control)* – Tohoto principu jsme se již v naší semestrální práci několikrát dotkli. V podstatě se jedná o to, aby skupina, která na produktu pracuje, měla definováno přesně akorát procesů. To je možné dosáhnout tím, že procesy budou definovány až ve chvíli, kdy skutečně vznikne potřeba jejich existence a ne dříve.
- **Teorie front** *(Queueing Theory)* – Hlubší popis tohoto principu by sám o sobě opět překročil rámec celé této práce, ale ve své podstatě se jedná o to,

aby fronty u sdílených zdrojů byly buď úplně eliminovány či alespoň správně řízeny.

Tímto krátkým shrnutím základních principů jsme myslím dostatečně postihli jejich základní smysl. Ačkoliv některé názvy mohou znít až marketingově, věříme, že jejích plné nastudování a pochopení je nezbytné k dokonalému nasazení LeSS frameworku.

3.5 LeSS návody

Protože i výše popsané principy nejsou konkrétními návody na zavedení LeSSu, připravili autoři frameworku několik návodů či tipů, které se osvědčily při předchozím nasazení a které mohou nové nasazení frameworku zjednodušit. Všechny návody jsou rozděleny do tří skupin a týkají se:

- Organizační struktury
- Managementu
- Technické dokonalosti

V následujících třech kapitolách uvedeme základní myšlenky tipů a návodů autorů frameworku, neklademe si však za cíl podrobně rozebrat veškeré návaznosti a celý kontext, poněvadž i to by zdaleka přesáhlo rámec celé práce.

3.5.1 Organizační struktura

Tipů, které se vztahují k organizační struktuře, je celkem 6 a mají za cíl být protipólem ke klasickému přístupu tvorby organizační struktury, kdy je cílem optimalizovat jednotlivé výstupy. Protipólem mají být z toho důvodu, aby byl, v souladu s jednou hlavních myšlenek LeSSu, neustále středem pozornosti výsledný produkt.

- **Týmy zaměřené na funkce** *(Feature teams)* – V klasickém pojetí organizace jsou pracovníci členění do specializovaných týmů, které se soustředí na jednu konkrétní oblast, ať je to analýza, vývoj či testování. Při použití týmů zaměřených na funkce se toto nahrazuje vytvořením plně soběstačného týmu, který je schopen všechny činnosti nutné k vývoji nové funkce zvládnout sám.
- **Organizační struktura** *(Organizational structure)* – V tomto tipu autoři shrnují, jak vypadá ideální organizační struktura. Jejím znakem je opět

jednoduchost, kdy týmy přímo podléhají vedení produktové skupiny, stejně jako Product owner a jeho případný tým.

- **Komunity** *(Communities)* – Ne vše je však při použití týmů zaměřených na funkce pozitivní. Velkým negativem je to, že lidé s podobným zaměřením mohou ztratit mezi sebou kontakt a přicházejí tak o možnost si sdělovat své zkušenosti, diskutovat nové trendy a metody. Tento nedostatek odstraňují komunity, což jsou skupiny lidí, které spojuje zaměření. Tyto skupiny jsou neformální, členství v nich není povinné a členové si schůzky plánují sami.
- **Scrum master** *(Scrummaster)* – V rámci tohoto tipu jsou uvedeny souvislosti práce Scrum mastera v širším kontextu, jsou zde rozvedeny oblasti, na které by se měl v čase soustředit (Tým, Product owner, Organizace, Vývojářské metody) a také seznam doporučené četby pro Scrum mastery.
- **Týmy** *(Teams)* – Tento tip prakticky shrnuje a rozšiřuje veškeré zákonitosti, které LeSS definuje a které se týkají týmů, neboť týmy jsou základní stavební jednotkou organizace. Je zde pohlíženo na správné vlastnosti týmů, vnitřní organizaci a procesy týmů, ale také řešení konfliktů v rámci týmů i mezi nimi.
- **Organizace dle hodnoty pro zákazníka** *(Organizing by customer value)* – Při použití standardního Scrumu je běžné, že všichni pracují na všem, autoři LeSSu si však uvědomují, že toto nemusí vždy dobře fungovat při zvětšení počtu pracovníků. Doporučují proto zvážit možnost rozdělení týmů do skupin podle toho, jakou přidanou hodnotu přináší zákazníkům. Například se může jednat o týmy, které vyvíjejí hlasovou, respektive textovou komunikaci u komunikační aplikace.

3.5.2 Management

Další sada tipů je zaměřena na efektivní management. Jak jsme již zmínili, pro LeSS není pozice manažera klíčová, v zásadě nemusí vůbec existovat a celý proces vývoje, při správném nastavení, bude fungovat bez ní. Pokud ale existuje, tak by manažer

měl být spíše osobností Y dle rozdělení typů pracovníků podle McGregorovy teorie XY [6]. Což znamená, že manažer by měl chtít pracovat ze své přirozenosti, řídit sám sebe a vyhledávat zodpovědnost, místo toho, aby se jí vyhýbal.

- **Jděte se podívat** *(Go See)* – Tímto tipem se autoři snaží poukázat na to, že by manažer neměl být odtržený od reality a měl by skutečně vědět, jak jeho podřízení pracují, znát jejich pracovní problémy a obecně by jim měl rozumět. To může být dosaženo tím, že je často navštěvuje, rozmlouvá s nimi či s nimi přímo sdílí kancelář.
- **Učení řešení problémů** *(Teaching problém solving)* – Jednou z manažerových rolí je také podporovat své podřízení v samostatném řešení problému. Doslova je zde zmíněno, že prací manažera je aktivně trénovat lidi, aby uměli přemýšlet sami za sebe. Rozhodně nežádoucí praktikou je, když podřízení přejímají řešení manažera bez hlubší znalosti věci.
- **Sebemanagement** *(Self-management)* – Základní části LeSSu jsou týmy, které vedou samy sebe. Stejný vzorec se musí zavést do chování manažera. Jeho úkolem není pohánět vývojáře do práce či jim nařizovat přesčasy. Praxí manažera by mělo být osvojení techniky „Jděte se podívat“ a to by mu tým měl oplácet maximální otevřeností a transparentností.
- **Zlepšovací služba** *(Improvement service)* – Častým nešvarem manažerů, který se nevyskytuje jen v oblasti vývoje softwaru, je to, že manažer se snaží zlepšovat něco, o čem on je přesvědčen, že zlepšení potřebuje. I přesto, že pocity řadových zaměstnanců mohou být zcela odlišné. Tomu se snaží předcházet Zlepšovací služba, která zavádí pojem Zlepšovací backlog [7], do nějž položky přidávají jen a pouze členové týmů či Scrum masteři a na pravidelných schůzkách manažera a týmů se rozhoduje, jaké věci se budou skutečně realizovat. Po realizaci následují věci známé ze standardního Scrumu – review a retrospektivy provedených zlepšení.
- **Manažer jako Scrum master?** *(Manager as Scrummaster?)* – Otázkou, která logicky v průběhu zavádění LeSSu vyvstane je, zda může být manažer zároveň Scrum master. Autoři uznávají oba přístupy k této

[6] Více o této teorii lze nalézt například v DVOŘÁKOVÁ, Zuzana a kol. Řízení lidských zdrojů. 1. vydání. Praha : C. H. Beck, 2012 v kapitole 11.2.4

[7] V originále Improvement Backlog

otázce, sami se však kloní k odpovědi ne, neboť pro manažera může být složité rozlišovat tyto dvě role a celá implementace LeSSu může skončit neúspěchem. Další přípustnou alternativou je, když manažer jednoho týmu je zároveň Scrum masterem týmu druhého. Zde však autoři uvádí, že zaznamenali smíšené výsledky.

- **Role manažera** *(Role of manager)* – Několikrát jsme výše zmínili, co by manažer dělat neměl či co dělat vůbec nesmí. Jaká je tedy role manažera a k čemu je v organizaci dobrý? Autoři frameworku tvrdí, že manažer by měl svým týmům vytvářet prostředí plné možností k jejich rozvoji a hlavně prostředí, které jim umožní vyvíjet dobré produkty. Dle velikosti organizace může být také spojovacím můstkem mezi celým produktovým týmem a zbytkem organizace.

3.5.3 Technická dokonalost

Co je často vytýkáno klasickému Scrumu, je absence konkrétních metod, které jsou použitelné při vývoji. Tento nedostatek se snaží LeSS odstranit a poskytuje deset návodů a tipů co zavést a používat, aby proces vývoje byl hladký, na chyby se přišlo brzy a vše se zákazníkovi dodalo v nejkratší možné době.

- **Kontinuální dodávky** *(Continuous delivery)* – Ačkoliv je tento tip na oficiálním webu zatím bez popisu, jeho smyslem bezpochyby je zavést neustálé dodávání rozpracovaného produktu zákazníkům. Protože tímto způsobem lze snadno a především brzy, čili levně, odhalit pokud vývoj uhne špatným směrem a provést nápravu.
- **Akceptační testování** *(Acceptance testing)* – Akceptační testování je ve vývojářském světě dobře známý pojem, tudíž není potřeba jej podrobněji vysvětlovat. Připomeňme však, že mimo uživatelského akceptačního testování existuje také interní akceptační testování, které může být prováděno například během Sprint review na základě scénářů zadaných během plánování Sprintu.
- **Architektura a Desing** *(Architecture & Desing)* – Obsahem tohoto tipu je poměrně rozsáhlý text, jehož smyslem je upozornit a připomenout, že software na rozdíl od budov není statický, takže se nedá mluvit o architektuře v pravém slova smyslu, protože každý vývojář přispěje svým

vlastním stylem. Tím pádem je potřeba, aby designéři systému byli pořád v těsném kontaktu s vývojáři a oba dva tábory udržovali agilní přístup k vyvíjenému produktu.

- **Čistý kód** *(Clean code)* – I tento tip prozatím neobsahuje mimo doporučené četby žádný popis. Dle doporučené četby je však neustále mít na paměti, že naším cílem je čistý kód, jenž se dá snadno číst a tím pádem také udržovat a rozvíjet.
- **Jednotkové testování** *(Unit testing)* – Stejně jako akceptační testování, i jednotkové testování je velmi dobře popsané a mělo by být vývojářům známé. Autoři však upozorňují na několik běžných nedorozumění, které vznikají při implementaci jednotkového testování, je jím například to, že jednotkové testy píší testeři a ne sami vývojáři. Jednotkové testování by nikdy nemělo v testovacím procesu figurovat samo, jeho funkce je spíš zkontrolovat, zda kód dělá to, co bylo zamýšleno. Měli by jej tedy psát sami vývojáři, kteří danou část kódu napsali.
- **Vývoj řízený testy** *(Test-driven development)* – Hlavní myšlenka této metody by se dala shrnout do třech bodů – 1. Napište test; 2. Napište přesně tolik kódu potřebného k projití testu; 3. Refaktorujte kód aby byl čistý.
- **Přemýšlení o testování** *(Think about testing)* – Aby testování nebylo samoúčelné a skutečně zajišťovalo bezchybný produkt, shrnuli autoři svoje poznatky do tohoto tipu. V jejich obsáhlém popisu boří zažitá dogmata, například že by vývoj neměl být oddělený od testů, či že nezdary automatických testů se musí opravovat okamžitě.
- **Automatizace testů** *(Test automation)* – Cílem veškerých testů je zajistit bezchybný produkt, ale často se stává, že vývojář zásahem do kódu může rozbít funkcionalitu, o které neměl tušení. Tomu se snaží předcházet automatické testování. Veškeré testy se automatizují a spouštějí se pravidelně tak, aby se na takto vzniklé chyby přišlo co nejdříve po jejich vzniku a mohly být ihned opraveny.
- **Specifikace příklady** *(Specification by example)* – Mezi požadavky a testy je samozřejmě úzká vazba, ideální je, když jsou požadavky specifikovány

podle existujícího příkladu, na jehož základě se pak vytvářejí testy, které kontrolují výsledný kód.

- **Kontinuální sestavováni** *(Continuous integration)* – Aby bylo možné zajistit kontinuální dodávky, je potřebné zavést i kontinuální sestavování, které bývá často označováno jako podstatná část celého agilního vývoje. I když vyžaduje změnu v chování vývojářů, aby vůbec bylo možné nasadit systém pro kontinuální sestavování, autoři říkají, že se to rozhodně vyplatí a radí nevzdávat se, pokud si myslíte, že váš produkt je příliš komplexní pro zavedení této metody.

3.6 Případové studie

LeSS framework samozřejmě již dávno není jen teoretická práce dvou odborníků, díky své jednoduchosti si získal příznivce i mezi velkými korporacemi a podniky. Za všechny jmenujme Nokii, John Deere, přístav v Rotterdamu, Ericsson či JP Morgan. Všechny tyto firmy mají společné, že před zavedením LeSS frameworku pracovali pomocí rigorózních metodik a jejich vývoj většinou stagnoval, dodával neotestované produkty pozdě, minoritní výjimky se vyskytovaly pouze na úrovni týmů. Hlavním problémem, který byl při nasazování LeSS identifikován byl chybějící Product owner a chybějící transparentnost procesu vývoje. Jako první bylo potřeba přesvědčit manažery, aby změnili svůj způsob myšlení a zahodili své dosavadní zkušenosti. Tím bylo dosaženo prostoru ke změně, čímž pádem mohl být vývojový proces a celá organizace produktového týmu změněna tak, aby odpovídala pravidlům a principům LeSSu. Nyní všechny zmíněné společnosti, i mnoho dalších, spojuje znatelné zlepšení vývoje a zkvalitnění produktu. Veškeré informace o reálných nasazeních LeSSu jsme však čerpali z oficiálních stránek projektu, protože se nám nepodařilo najít žádné jiné relevantní zdroje zabývající se tímto tématem.

3.7 Zhodnocení

Ačkoliv LeSS samozřejmě není univerzální řešení na všechny problémy, věříme, že může výrazně pomoci ke škálování Scrumu v organizaci. A to i přesto, že se jedná o nový nástroj v této oblasti. I nadále je však nezbytné mít na klíčových pozicích pracovníky, kteří mají již se Scrumem zkušenosti a mají dobře zažité jeho principy.

To samozřejmě vyžadují i další metodiky zmíněné v této práci, u LeSSu je však tato potřeba extrémně vysoká, neboť nechává v mnohém volnou ruku samotné organizaci, která se ho rozhodne implementovat. Ze všech představených metodik nám LeSS připadá nejlépe použitelný, neboť se skutečně jedná o komplexně vyškálovaný Scrum, ne o Scrum rozšířený o škálování. Proto bychom ho doporučili alespoň k prostudování každému, kdo se zajímá o škálování agilních metodik, poněvadž jeho principy jsou široce využitelné i mimo Scrum.

4 Závěr

V předchozích řádkách jsme se zaměřili na možnosti škálování SCRUM. S ohledem na popularitu agilního přístupu k vývoji není překvapením, že pro větší projekty a společnosti je třeba řešit otázku škálování. V menším měřítku není třeba řešit, zda je lepší Scrum nebo Kanban, ale všechny metodiky vhodně kombinovat, včetně například extrémního programování.

Scrum of Scrums je důležitá základní technika pro škálování větších projektových týmů. Scaled Agile Framework má bezpochyby výhody pro opravdu velké organizace, názory odborné veřejnosti však nešetří skeptickými názory a to zejména právě kvůli jeho směřování k vrcholovému managementu. Disciplined Agile Delivery pak umožňuje zjednodušit procesní rozhodování při dodávce IT řešení za pomoci hybridního přístupu, kdy vhodně kombinuje jak agilní přístupy, tak ty tradiční. Jak již bylo zmíněno, daleko zajímavější je v této oblasti však LeSS, který nemíří pouze na velké korporace, ani v jedné takové nevznikl, a je opravdu otevřený a přizpůsobivý potřebám konkrétní organizace.

5 Zdroje

1. ZIKMUND, J. *Škálování agilních metodik*. Praha: 2012. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky [cit. 2015-05-07]. Dostupné z: http://www.vse.cz/vskp/eid/33214

2. SCRUM ALLIANCE, I. SCRUM ALLIANCE. *Peter Stevens* [online]. 2015 [cit. 2015-05-14]. Dostupné z: https://www.scrumalliance.org/community/profile/pstevens

3. STEVENS, P. scrum breakfast. *Scaling Scrum: SAFe, DAD, or LeSS?* [online]. 2013 [cit. 2015-05-14]. Dostupné z: http://www.scrum-breakfast.com/2013/10/scaling-scrum-safe-dad-or-less.html

4. SCRUM ALLIANCE, I. SCRUM ALLIANCE. *Advice on Conducting the Scrum of Scrums Meeting* [online]. 2007 [cit. 2015-05-14]. Dostupné z: https://www.scrumalliance.org/community/articles/2007/may/advice-on-conducting-the-scrum-of-scrums-meeting

5. ALLIANCE, A. Agile Alliance. *Scrum Of Scrums* [online]. 2013 [cit. 2015-05-14]. Dostupné z: http://guide.agilealliance.org/guide/scrumofscrums.html

6. SCALED AGILE, I. Scaled Agile Framework. *Scaled Agile Framework* [online]. 2015 [cit. 2015-05-09]. Dostupné z: http://www.scaledagileframework.com

7. SCALED AGILE, I. Scaled Agile Framework. *Scaled Agile Framework Core Values* [online]. 2015 [cit. 2015-05-09]. Dostupné z: http://www.scaledagileframework.com/safe-core-values/

8. Disciplined Agile Delivery. *Introduction to DAD* [online]. [cit. 2015-05-09]. Dostupné z: https://disciplinedagiledelivery.wordpress.com/introduction-to-dad/

9. B.V. T. L. C. *LeSS* [online]. 2014 [cit. 2015-05-10]. Dostupné z: http://less.works

10. BROWN, A. Scrum Inc. [online]. Dostupné také z: http://www.scruminc.com/wp-content/uploads/2014/07/Scrum-at-Scale-A-Modular-Approach.pdf