TAJEMSTVÍ ÚSPĚŠNÝCH SOFTWAROVÝCH TÝMŮ

Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	**LS 2017/2018**
Autoři **– jméno, příjmení, xname**	**Tereza Staňková, stat02** **Bílý Radim, xbilr00** **Kuklová Andrea, xkuka02**
Téma	**Tajemství úspěšných softwarových týmů**
Datum odevzdání	**12. 05. 2018**

Abstrakt: V první kapitole této seminární práci jsou identifikovány metriky pro rozlišení úspěšných a neúspěšných softwarových týmů. Další kapitola se zabývá hlavními problémy, se kterými se vývojové softwarové týmy potýkají. Ve třetí kapitole jsou definovány principy úspěšných softwarových týmů. Čtvrtá kapitola popisuje možnosti manažera ovlivnit výkon vývojového týmu.

Klíčová slova: softwarový tým, efektivita týmu, úspěšné týmy, role manažera týmu, prioritizace, průběh projektu

Obsah

Úvod

S digitalizací světové ekonomiky se problematika softwarového vývoje dere stále více do popředí. Vývojových týmů existuje totiž stále více. Od malých garážových start-upů po velké technologické korporace vyvíjející produkty pro miliardy uživatelů. Přestože způsoby řízení těchto různě velkých týmů jsou z logiky věci různé, vidíme u nich jeden společný prvek – snahu o maximální efektivitu.

A tím se dostáváme k tématu naší seminární práce, které jsme nazvali prozaicky „Tajemství úspěšných softwarových týmů“. Vycházíme v ní hodně z článku Steva Mezaka Secret of High-Performance Software Teams publikovaném v roce 2017 v periodiku Better Software, který jsme doplnili několika dalšími prameny.

Cíle naší seminární práce jsou následující:

- identifikovat metriky, podle kterých lze rozlišit úspěšné a neúspěšné softwarové týmy;
- identifikovat hlavní problémy, se kterými se vývojové softwarové týmy potýkají;
- definovat zásady, které jsou v úspěšných softwarových týmech běžné;
- zjistit možnosti manažera ovlivnit výkon vývojového týmu, případně možnosti k tomu vhodné.

V rámci seminární práce budeme využívat nejen akademické prameny, ale budeme též vycházet z vlastních zkušeností, které se budeme snažit při zpracování seminární práce vhodně využít.

1. Jak změřit efektivitu týmu

V úvodních kapitolách seminární práce si definujeme stěžejní problémy, které primárně zapříčiňují snižující se efektivitu softwarových týmů. Ještě, než se ale k popisu samotných problémů dostaneme, je nejprve nasnadě si definovat již zmíněný pojem efektivita. Existuje hned několik metod, díky kterým je možné měřit efektivitu v týmu. Každá z metod má své specifikum a nahlíží na problematiku podle jiných kritérií – což je logické, protože ne vždy je vůbec možné objektivně posoudit týmovou efektivitu. Pak je určitě nasnadě mít více nástrojů a podle určité situace vybrat ten nejvhodnější.

V praxi se velmi často požívá metoda hodnocení na základě stanovených cílů. Hodnocení výkonu se tedy provádí na základě předem stanovených norem. V případě, že činnost týmu můžeme popsat jako výkon práce v dané kvalitě a za určitý čas, pak není žádný problém s měřitelností efektivity a celkové objektivity. Je jednoduché posoudit, zda tým vyprodukoval více či méně ve sledovaném časovém úseku. Tato metoda je tedy do určitého bodu skvělou volbou k měření efektivity týmu. Pokud se stane, že potřebujeme jít v reflexi týmu hlouběji a není dostatečné vyhodnocení pouze na bázi zpracováno či nezpracováno, pak tato metoda není úplně vhodná. Převážně v případě, že je tým různorodý a členové mají odlišnou náplň práce, není jednoduché dojít k jednoznačnému závěru. Členové týmu mají různé vzdělání, schopnosti i týmové role, v případě tvorby software či strategie pro konkrétního klienta, již není tak jednoduché posoudit, zda bylo maximální efektivity dosaženo. Přestože bude software naprogramován či strategie vytvořena, manažer či vedení nemá k dispozici dostatek informací, aby mohli posoudit, zda se daný proces dal zvládnout lépe, s menšími chybami, levněji či v kratším čase. Tato metoda je orientována spíše na posuzování celkového výsledku než na samotnou týmovou efektivitu týmu. Existují ale metodiky, které si právě s těmito typy situací umí poradit. (Stýblo, 2011)

Jednou z prvních byla metodika zaměřující se na role v pracovních skupinách. Na základě výzkumu ji v roce 1948 vytvořila dvojice Benne a Sheats. Na principech této metodiky můžeme člověka rozdělit do tří kategorií:

- Role zaměřené na výkon – Iniciátor (přichází s novými nápady na řešení), Hybatel (vybízí ostatní k jednání), Zpracovatel (rozpracovává předložené názory);
- Role zaměřené na budování a udržování skupiny (týmu) – Povzbuzovatel (vybízí ostatní k jednání), Pozorovatel (objektivně komentuje atmosféru ve skupině);
- Role zaměřené na sebe – Agresor (prosazuje se napadáním ostatních), Obhájce vlastních zájmů (stále trvá na svém), Playboy (odvádí pozornost, vtipkuje).

Tyto role se můžou překrývat např. jako Hybatel a Povzbuzovatel, mají stejný cíl, ale liší se v orientaci. Metodika se primárně zaměřuje na interakci v týmu, neklade důraz na osobností rysy týmových členů či jejich potenciál. (Hayes, 2001)

Další metodikou, kterou si uvedeme, je metodika týmových rolí dle M. Belbin. Doktor Belbin založil svou teorii na výzkumu, kdy sledoval manažery z různých koutů světa a jejich práci s týmy. Z toho vyvodil tři druhy týmových pracovníků a to:

- Role zaměřené na výkon;
- Role zaměřené na lidi;
- Role zaměřené na kontrolu a analýzu.

Metodika vychází z hodnocení sebe sama v různých situacích. Člen týmu většinou zastává více rolí, někdy je to i nutné vzhledem k měnící se zodpovědnosti. O negativní zpětnou vazbu k této metodice se postaral fakt, že je založena na sebehodnocení, které může být u některých jedinců velmi zkreslené. (Belbin, 1993)

Poslední metodikou, která bude představena, je Balesovo schéma interakční analýzy. Jedná se o kvantitativní metodiku, která je založena na systému pozorovacích kategorií. Pozorují se předem zvolené kategorie, které si předem určité. Není možné rozklíčovat celou komunikaci, jelikož je příliš obsáhlá. V momentě, kdy je jasné, které aspekty jsou pro nás zajímavé, musíme zajistit objektivitu pozorování. Touto metodikou můžeme sledovat povrchové chování jedince, ale nejsme schopni odhalit jedincovi motivy. Zkoumáme tedy pouze část efektivity. (Pavlica, 2000)

2. Hlavní problémy snižující efektivitu týmu

Negativních vlivů či přímo problémů, které negativně ovlivňují efektivitu týmu, je celá řada. V této podkapitole se budeme snažit vyzdvihnout ty nejvýznamnější, které mají vážné důsledky na týmové fungování a následné plnění firemních cílů. Zpravidla pracovní týmy nedosahují výsledků, protože nestíhají důležité termíny nebo značně překračují náklady. To ale spíše můžeme definovat jako samotný nepříznivý důsledek trvajících problémů. Každý softwarový tým by měl mít na paměti, že musí být především profitabilní. Značný vliv zde zastává projektový manažer či produktový manažer, který svým přístupem může svůj tým zásadně ovlivnit. Manažeři mají nejblíže k názorům vedení na vývoj software, a tak většina týmové energie pochází z manažerova myšlení a chování.

Níže se zaměříme na tři hlavní problémy, které jsou zdrojem poklesu efektivity a profitability v týmu. Mohou tedy negativně poškodit samotné fungování softwarového týmu. (Mezak, 2017)

2.1. Soustředěnost na náklady namísto kvality

Důraz je kladen na nadměrné zdůrazňování všech nákladů. Může jít o přehnanou snahu v souvislosti s dodržováním nákladů, která může vyústit až v nákladové omezení. Tento problém ale v důsledku způsobí závažné škody jako např. omezené nebo pozdní spuštění. Zde nastává kolize, protože investovaný lidský kapitál a úsilí není plně využito a dochází k neefektivitě. V případě, že v rámci přílišného šetření společnost představí klientovi neuspokojivý software, tak dochází ke ztrátě zdrojů i času. Je pochopitelné, že firemní zdroje nejsou neomezené, není ale dobré se omezeností zdrojů nechat limitovat. Některé společnosti

se ve snaze ušetřit uchylují k outsourcingu týmu na vývoj softwaru. Outsourcing je vhodné brát jako zdroj k oslovení zkušených vývojářů ne jako způsob, jak ušetřit. Úkolem manažera je nastavit rozpočet tak, aby stál na základě reálných předpokladů.

2.2. Chybí definice “hotového úkolu“ a špatná prioritizace

Stěžejní je si v rámci týmu potažmo celé společnosti nastavit, co znamená hotový úkol a jaké jsou jeho charakteristiky. Bez tohoto kroku může dojít k tomu, že úkoly budou zůstávat nedokončené nebo se naopak do nekonečna budou přehazovat mezi členy týmu, aby se řešily nepodstatné detaily. Často se neklade důraz na vyřízení problémů podle jejich důležitosti. Je důležité, aby samotná definice “hotového úkolu“ přišla od pracovního týmu. Může se stát, že samotné vedení nebude brát důležité úkoly pro softwarový tým jako prioritu. V této situaci je dobré si uvědomit, co vytvářený software do firmy přináší. A definovat si plán s akčními kroky k dokončení celého projektu. Dle provedené studie jsou týmy, které se rychle vypořádávají s důležitými nedokončenými úkoly, v drtivé většině v kategorii úspěšných. Pokud je tým schopný plynule přecházet z úkolu na úkol, vyvaruje se problémům s výsledky a financemi.

2.3. Chybí podpora vize a inovací

Především na začátku na místo diskuzí o nákladech a úsporách, by si tým měl vytvořit vizi své práce a vytvářeného software. Na začátku je ještě dost času, aby se tým mohl zkontaktovat s potenciálními uživateli a zákazníky a zmapoval všechny příležitosti. Mít od začátku jasnou vizi pomůže později k udržení soustředění. Povědomí o oboru a ostatních projektech také pomůže vytvořit realistický rozpočet. K nalezení vize je třeba přemýšlet v rámci dlouhého období ve vyšším měřítku. Často na to týmový kolegové zapomínají, ale při tvorbě vize či jakékoli týmové práci je stěžejní naslouchat partnerovi či kolegům. Netrvat tedy jen na svém názoru, ale nechat prostor i ostatním, aby vyjádřili své myšlenky. Podporováním sdílení vlastních nápadů a myšlenek vývojářů, je daleko snadněji možné vytvořit opravdu kvalitní software. Vize musí být taktéž podpořena plánem kroků, aby tým mohl začít pracovat na jejím naplnění. Je chyba při plánování spuštění brát v úvahu pouze rozpočet, je třeba nahlížet také na zdroje, čas, zaměstnance a možnost outsourcingu. Rozsah a potenciál projektu by neměl být v zásadní části limitován rozpočtem. Pokud manažer týmu sám ve vizi nevěří, nemůže plnohodnotně vést ani softwarový tým a ovlivňuje tak myšlenkové nastavení celé skupiny. (Mezak, 2017)

3. Tajemství úspěchu týmů

V této kapitole budou uvedeny zásady doporučované odborníky na vedení softwarových týmů, kterých by se týmy a jejich vedení mělo držet. Budou také krátce představeny principy pro

práci týmů tří známých a úspěšných firem – Netflixu, Facebooku a GitLabu. Dále bude uveden doporučený postup manažera v průběhu projektu.

3.1. Zásady úspěšných softwarových týmů

Vytvoření skvělého softwaru vyžaduje, aby všechny zúčastněné strany souhlasily s dodržováním určitých postupů. Formální pravidla a metody mohou být přínosné. Ne vždy ale každý chce a může dodržovat zavedené podrobné a mnohdy zdlouhavé či zastaralé procesy. Potřeba je tedy více neformální přístup ke společné práci. To zajistí zavedení všemi uznávaných principů pro společnou práci. (Miller, 2017)

Miller (2017), manažer s 20 lety zkušeností specializující se na webový vývoj, představil PACT, jakožto soubor čtyř zásad – přípravy, adaptace, komunikace a důvěry.

Miranda (2016), viceprezident pro inženýrství (VPE) v Doximity, shrnul deset principů úspěšných týmů softwarového vývoje. Patří k nim podle něj sdílené vlastnictví kódu, otevřená diskuze, vyvážený pracovní a rodinný život, psaní znovupoužitelného kódu, dokumentace. Každý člen týmu by měl rozlišovat mezi tvořením prototypu a tím, co bude předáno zákazníkovi, zanechávat věci v lepším stavu, než ve kterém byly nalezeny, učit se nové nástroje, ale používat je s rozvahou, učit, co se naučil, poučit se z vlastních chyb a přizpůsobit se.

Sdílené vlastnictví kódu znamená, že nezáleží na tom, kdo psal, jakou část kódu. Všichni jsou zodpovědní za špatná i dobrá rozhodnutí. Miranda (2016) doporučuje také limitovat rozsah práce na 40 hodin týdně, aby nedošlo k vyhoření zaměstnanců. Pro poučení se z chyb je důležitá retrospektiva – sezení, kde je předána pozitivní i negativní zpětná vazba mezi členy týmu.

Pronschinske (2018), výzkumník v oblasti vývoje softwaru se sedmiletou praxí, doporučuje manažerům zavedení prostředí, které inspiruje k zvažování nových možností a najímání pouze zodpovědných lidí, kteří si poskytnutou svobodu zaslouží. Říká, že týmy by neměly být omezovány příliš podrobnými procesy a mělo by jim být umožněno dělat chyby, z kterých se mohou poučit. Zásadní je dle něj vytvoření inovativního prostředí. Vedení by mělo podporovat vstřícnost, pokoru a komunikaci. Adaptibilita a dynamika týmu může být zajištěna menšími týmy s kratšími (menšími) projekty.

Správnou komunikaci považují za klíčovou všichni tři experti. V rámci komunikace je dle Millera (2017) třeba sdílet co nejvíce. Pokud není možný osobní kontakt, může k ní být využit Slack, Google Hangouts nebo jiné systémy umožňující přenos videa a zpráv. Komunikační bariéry brání v úspěchu. Každý člen týmu by se měl cítit zmocněn ke sdělení zásadních informací jakémukoliv členu týmu bez ohledu na hierarchii. Zásadní je tedy komunikace beze strachu. Vedení se nemůže rozhodovat bez informací. Během otevřené diskuze není žádná otázka nevhodná a každý by měl být vyslyšen. (Miranda, 2016)

S komunikací úzce souvisí důvěra, která dle Millera (2017) také patří k principům úspěšného týmu. Důvěra je základním kamenem úspěchu týmu. Manažer by měl na začátku projektu důvěřovat všem členům týmu a tuto důvěru jim projevit. Tým nemá většinou žádný důvod snažit se o neúspěch projektu. Manažer by tedy neměl hned na začátku projektu očekávat, že bude někým podveden. To se totiž nepříznivě projevuje v komunikaci a výsledcích členů týmu. Adaptace je také společným tématem všech tří expertů. Na konkurenčním trhu je pro firmu důležité růst a adaptovat se. Pokud tým nevytváří velmi malý produkt pravděpodobně bude nucen v průběhu vývoje ke změně. Dodržovat zásadu adaptace znamená být ochoten zohýbat, změnit nebo porušit některá nebo všechna pravidla, když je to potřeba. (Miller, 2017)

Aby byl tým výkonný, je důležité, aby mezi jeho členy docházelo k efektivní souhře. Týmy pro vývoj softwaru jsou ale často formovány znovu pro každý nový projekt na základě požadavků a dostupnosti jednotlivých členů. Tým tedy nemá zkušenost se společnou prací na projektu, což může efektivní společnou práci zkomplikovat. (Faraj & Sproull, 2000)

Faraj a Sproull (2000) rozlišují souhru na administrativní a souhru odborných znalostí. Administrativní koordinace je podstatná pro jednoduché rutinní úkoly. Jde o přidělování úkolů, alokování zdrojů a integraci výstupů. Pro komplexní úlohy je pak zásadní expertní souhra.

Členové týmu by měli být schopni určit, kde a u koho se nachází potřebné znalosti a dovednosti. Znalost oblasti expertízy mezi členy týmu zlepšuje jeho výkon. Členové by tedy měly mít vyvinut společný jazyk pro přidělování úkolů, rolí a oblast expertních znalostí. Poznat, kdy a kde jsou specifické znalosti potřebné je stejně tak důležité. (Faraj & Sproull, 2000)
Výše uvedené zásady aplikují některé celosvětově známé firmy. Pronschinske (2018) uvádí mimo jiné Netflix, GitLab a Facebook. Ve firmě Netflix, americkém poskytovateli online filmů, panuje kladný vztah k inovacím a snaha o minimální byrokracii. Zásadní je svoboda, ale také zodpovědnost. V Netflixu je každý vývojář zodpovědný za to, co vytvoří. Musí opravit svojí práci, pokud dojde k poruše, a psát dokumentaci.

V GitLabu jsou týmy vývojářů tvořeny většinou čtyřmi lidmi, kteří mají na starosti kromě vývoje i produktový management. Personál společnosti GitLab má tedy také velkou dávku svobody. Facebook podporuje hlavně rychlost vydání nového software, dělání chyb a učení se z nich. Vzhledem k rychle měnícímu se odvětví firmy management nemá plán na příštích pět let, ale pouhých šest měsíců. (Pronschinske, 2018)

3.2. Průběh projektu pro manažera

V roce 1995 utratily organizace v Spojených Státech Amerických za software 152 miliard dolarů. Většina velkých softwarových projektů není dokončena načas či v rámci rozpočtu a nefungují, jak bylo zamýšleno. (Faraj & Sproull, 2000)

Úspěch projektu záleží z velké části na produktovém či projektovém manažerovi. Jeho zodpovědností je zajistit, aby se tým soustředil na hlavní cíle projektu a jeho užitečnost pro zákazníky a uživatele. (Mezak, 2017)

Rajan (2017), IT konzultant a bývalý projektový manažer IBM, považuje za nejdůležitější chování manažera neustálé učení se a každodenní improvizaci. Po mnoha letech zkušeností doporučuje vytvářet opakovatelné, spolehlivé a odolné procesy pro vypouštění softwaru, které jsou v souladu se všemi členy týmu a které jsou mnohokrát testovány. Manažer by neměl být ve funkci zuřivého hlídacího psa, ale být spíše pastýřem, který zajišťuje, že jsou všechny zdroje (software, hardware a lidi) zařazeni do procesu tak, aby byla zajištěna maximální přidaná hodnota pro zákazníka.

Během přípravy projektu by dle Millera (2017) mělo dojít k ujasnění cílů a jejich komunikaci týmu. Zajištění všeho potřebného, co tým potřebuje k úspěchu – zdroje, nástroje, licence a software. Týmu je dána pravomoc rozhodovat o věcech, které přímo ovlivňují cíle jejich projektu. Dle Mezaka (2017) se manažer na začátku projektu ideálně zaměří na průzkum trhu k ujasnění vizi pro budoucí software. Vizi konzultuje s potenciálními zákazníky a uživateli. V průběhu projektu na průzkumy už nebude čas.

Vizi je také nutné projednat s týmem. Členové týmu se k projektu mohou svobodně vyjádřit, jsou povzbuzováni v nových nápadech a řešeních. Během diskuze je podporováno inovativní myšlení. Vyhrazen je čas i na retrospekci minulých projektů, z kterých může tým čerpat zkušenosti a doporučení pro nový projekt. (Mezak, 2017)

Na mnoha projektech vývojáři hází svoji práci na testery, kteří ji přehazují na provozní oddělení. Manažer by měl zajistit, aby každý byl zapojen do projektu hned od jeho počátku. Měl by vytvořit komunikační kanály a podporovat diskuzi v týmu. (Rajan, 2017)

Po přípravné fázi je stanoven plán projektu, který udává směr, kam se software bude ubírat. Dobrý manažer stanoví očekávaný rozpočet s realistickými předpoklady dopředu. Někdy je ale nutné rozpočet upravit a nemusí se jednat o nic špatného. Jedním z dobrých důvodů je uživatelská zpětná vazba, která může výrazně změnit rozpočet i časový harmonogram projektu. (Mezak, 2017)

Naplánovány jsou potřebné dílčí úkoly včetně časového harmonogramu, personální a jiné zdroje. Manažer naplánuje nejtěžší část projektu frekventovaně a nenechává jí na poslední chvíli. Pokud tým například nerad testuje projekt, testování by mělo být nasazeno hned od prvního dne a děláno po menších částech. Rajan (2017) také doporučuje zaměřit se na kvalitu. Zachycovat nedostatky včas šetří mnoho peněz. Jakmile je nalezena chyba, je potřeba ji opravit co nejdříve.

Důležitá je také jasná definice toho, kdy je úkol hotov. Manažer ujasňuje priority pro zbývající úkoly dle jejich hodnoty. Některé softwarové funkce se mohou jevit jako příliš komplexní a jejich zajištění je velice nákladné. Správný manažer se nebojí zajistit ani tyto funkce v plné kvalitě, produkt je potřebuje ke svému úspěchu. Pokud se bude snažit ušetřit na klíčových vlastnostech softwaru, může se stát, že ho uživatelé nebudou používat a celý projekt bude proděločný. Větší spokojenost uživatelů s produktem vede v mnoha případech k větší ziskovosti projektu a investice ve formě větších nákladů se tedy vyplatí. (Mezak, 2017)

Manažer by neměl při stanovování priorit nadhodnocovat význam nákladů vývoje. Jednou z častých chyb vedoucích týmů je přílišná snaha o minimalizaci nákladů vývoje za cenu snížení kvality výsledného produktu. Důsledkem může být například opožděné spuštění, či nesplnění některých požadavků. (Mezak, 2017)

Vše, co lze, by se mělo automatizovat. Automatizujte pouze efektivní procesy, což jejich efektivitu umocní. Verze všeho, co je potřeba vytvořit, otestovat a nasadit je třeba ukládat. To se týká dokumentace požadavků na aplikaci, testovacích skriptů a případů i knihoven. Nespoléhejte se na lidskou paměť. Díky ukládání verzí se můžete vrátit do místa před tím, než se něco pokazilo. Rajan (2017) opět upozorňuje na důležitost adaptace. Říká, že pokud manažer nezvládne udržet tempo s neustále se měnícím prostředím, neuspěje.

Pro shrnutí, manažerům je doporučeno provádět výzkum trhu na začátku projektu, podporovat otevřenou komunikace a inovace, nenadhodnocovat náklady, stanovovat jasné priority úkolům, automatizovat a ukládat verze.

4. Jak může manažer ovlivnit efektivitu týmu

V této části práce si ukážeme možnosti manažera, jak může ovlivnit efektivitu týmu. Respektive se vůbec zamyslíme, do jaké míry vůbec může manažer ovlivňovat efektivitu týmu v době pokročilých vývojových metodik, které jsou poměrně striktní ohledně rozdělení rolí v týmu. V případě, že dospějeme k závěru, že manažer má možnost ovlivnit výkonnost týmu, ukážeme si vybrané best-practices, jak toho může dosáhnout.

Při citování externích zdrojů obvykle nerozlišujeme mezi funkcí vedoucího vývojového týmu a projektového manažera. Uvědomujeme si, že mezi těmito dvěma pozicemi existují z hlediska náplně jejich práce určité rozdíly. Pro potřeby naší práce však můžeme od těchto rozdílů abstrahovat, neboť projektový manažer je v citovaných článcích vždy vnímán jako osoba vedoucí daný vývojový tým.

4.1. Význam manažera při řízení výkonnosti vývojového týmu

Již v úvodu této kapitoly jsme vyjádřili možnou pochybnost nad rolí manažera v době rozvinutých vývojových metodik, které poměrně striktně definují jednotlivé role v týmu. Dříve to bylo jasné. V době, kdy vývojové metodiky byly poměrně nerozvinuté, byly, alespoň z mého pohledu, dovednosti manažera jediným dělítkem mezi úspěšnými a neúspěšnými týmy. Moderní vývojové metodiky (jako scrum apod.) však mají poměrně jasně definované týmové role. (Gonçalves & Lopes, 2014) To by mohlo podporovat tezi, že role manažera vývojového týmu již není tolik podstatná.

Tuto tezi však vyvrací velké množství aktuálních článků o správném vedení IT týmů, které byly vydány v posledních letech. Obsah článků poměrně hromadně ukazuje, že se mění význam role vedoucího vývojového týmu. Obsahem jeho práce není již tolik správné nastavení procesů v týmu (to řeší jednotlivé vývojové metodiky), nýbrž spíše sladění interpersonální komunikace mezi různými osobnostními typy. (Creasy, 2013) (Petter, 2009)

4.2. Role manažera ve vývojovém týmu

Ve výše zmíněném odstavci jsme tedy došli k závěru, že i v době pokročilých vývojových metodik může ovlivnit manažer vývojového týmu ovlivňovat jeho výkonnost. Pojďme si ukázat však, jaká je přesněji jeho role.

Dle výše zmíněného obsahu by se mohlo zdát, že role manažera je být takovým koučem a psychologem vývojového týmu. Dbát na efektivní komunikaci a dobré vztahy uvnitř týmu. Je to však jejich jediná role?

Poměrně komplexně se úlohou projektového manažera zabývá článek Anantatmuly v Engineering Management Journal. Tento článek vysvětluje, že kompetence vedoucího týmu začínají už v okamžiku výběru jeho členů, kteří musí mít nejen vhodné charakteristiky pro danou pozici, projekt, ale zároveň musí být jeden vůči druhému osobnostně kompatibilní. To v praxi znamená stát se takovým malým HR specialistou. Výběrem členů týmu ale práce teprve začíná. Autor v článku hodně rozebírá styl, jakým jsou jednotlivé vývojové týmy vedeny. Zmiňuje fakt, že vedoucí vývojového týmu (v tomto případě projektový manažer) obvykle zodpovídá za doručení projektu, o kterém má jen obecné znalosti a musí se na projekt vždy adaptovat, aby dosáhl dobrého výsledku. A zároveň musí být schopen své nově nabyté poznatky předat i ostatním členům vývojového týmu. Jako ideál zmiňuje autor stav tzv. leadershipu, kdy vedoucí vývojového týmu nevystupuje jako manažer udílející úkoly, ale jako leader motivující ostatní členy k nejlepšímu možnému výsledku. (Anantatmula, 2010)

Projektový manažer je často také postaven před složitá rozhodovací dilemata. Příkladem je například dodržení plánovaného rozpočtu vs. dosažení nejlepšího možného výstupu. Již zmiňovaný článek od Steva Mezaka se k tomuto dilematu staví poměrně jasně. Říká, že je nutné vyvinout co nejlepší produkt bez ohledu na překročení plánovaného rozpočtu. (Mezak, 2017)

S tím si však dovolíme polemizovat. U vlastního vývoje můžeme dle našeho názoru praktikovat tento přístup, pokud tím výrazně neoddálíme uvedení produktu na trh (domníváme se, že je lepší uvést na trh funkční produkt bez některých vybraných funkcionalit, než produkt žádný). U agenturního vývoje, kdy klient není ochoten akceptovat významnější navýšení rozpočtu, považujeme tento přístup za ještě daleko problematičtější.

Shrneme-li si výše zmíněné, vyplývá nám z toho, že role manažera spočívá především v těchto 3 kompetencích:

1. pochopení smyslu projektu (přestože se může jednat o projekt jemu/jí neznámý) a jeho předání ostatním členům vývojového týmu;
2. péče o komunikaci a mezilidské vztahy uvnitř týmu, vedení příkladem;
3. činění operativních rozhodnutí v průběhu projektu, průběžná úprava jeho cílů a plánovaných výstupů.

Kromě technických znalostí by tedy měl být projektový manažer obdařen především silnými kompetencemi v oblasti soft-skills.

4.3. Praktická opatření pro zvýšení výkonnosti

Zatímco v předcházejících částech této kapitoly jsme se zaměřovali na roli manažera vývojového týmu jako takovou, nyní si ukážeme praktická opatření, jak v již fungujícím vývojovém týmu výkonnost zvyšovat. Jeden z členů našeho týmu, Radim Bílý, sám pracuje v digitální agentuře Fresh Services zabývající se vývojem webových projektů malého rozsahu. Přestože jsou projekty malého rozsahu a nelze při nich uplatňovat pokročilejší vývojové metodiky (či alespoň převládá v agentuře takový názor), setkal se Radim Bílý v průběhu své práce v agentuře a vedení webových projektů s několika úspěšnými praktikami, ze kterých zde budeme vycházet. Zmíněné praktiky se týkají především efektivity týmové komunikace, u které jsme již v předešlých částech kapitoly došli k názoru, že se jedná o jednu z klíčových kompetencí vedoucího vývojového týmu.

4.3.1. Projektová platforma

Přestože projektové meetingy, tabule a osobní komunikace mají při vedení projektů své místo, nelze na ně zdaleka spoléhat jako na jediné nástroje. Zvláště ne v týmech, kde je fungující praxí práce z domova, při které vývojář opravdu nevidí jakoukoliv vývojovou tabuli umístěnou v kanceláři. Proto by měly výkonné softwarové týmy spoléhat na některou z projektových platforem typu Base Camp, Apollo, Teamwork apod. Ty vesměs umožňují u jednotlivých projektů následující funkce:

- přiřazovat vývojáře k jednotlivým projektům;
- sdílet podklady k těmto projektům;
- přiřazovat vývojářům konkrétní úkoly na projektu a sledovat jejich splnění;
- zobrazovat, v jaké fázi se projekt nachází a jakých milníků již bylo dosaženo;
- stanovovat vývojářům deadliny (mělo by být činěno po oboustranné dohodě, pozn. autorů) a sledovat jejich splnění;
- trackovat čas strávený jednotlivým vývojářem na přiděleném úkolu. (Captera Inc., 2018)

Zároveň umožňují mít o každé aktivitě písemný záznam, čímž předchází případným dohadům mezi jednotlivými členy vývojového týmu a jeho manažerem.

4.3.2. Instant Messenger

Přestože má projektová platforma mnoho výhod, na okamžitou komunikaci někdy může být trošku těžkopádná. Je proto vhodné mít pro okamžitou komunikaci ještě instant-messagingovou platformu. Tou může být pro firmy například Slack, který umí členit konverzaci do několika vláken (například dle projektů) a do určitého rozsahu je zdarma. (Slack Ltd., 2018) V rámci vývojového týmu je však nutné přesně stanovit, jaká komunikace bude probíhat přes projektovou platformu a jaká přes instant-messenger. Například v agentuře, ve které pracuje kolega Bílý, se ujalo pravidlo, že projektová platforma je pro zadávání úkolů, kompletní plánování projektu a všechny potřebné náležitosti s projektem spojené, zatímco instant-messenger je pouze pro rychlou diskuzi, ujasnění si zadání u jednotlivých úkolů a neformální diskuzi.

4.3.3. Dlouhodobá projektová analytika

Pokud na jednotlivých projektech pracuje opakovaně podobná sestava lidí (týká se například krátkodobého zakázkového vývoje), přináší to prostor pro projektovou analytiku. Například projektová platforma Apollo (a dle našich zkušeností mají i ostatní projektové platformy podobnou funkcionalitu) umí kumulovat u jednotlivých projektů data, která se například týkají času stráveném na jednotlivých úkolech. (Apollo Team, 2018) To může při správném využití (například porovnání plánovaných a reálných časů strávených na konkrétních úkolech) vést k realističtějšímu projektovému plánování v dlouhodobém horizontu.

V této kapitole jsme dospěli k závěru, že i v době pokročilých vývojových metodik může stále manažer v rámci projektu významně ovlivňovat jeho výkon. Následně jsme si blíže definovali jeho role. Poté jsme popsali několik reálných praktik, jak výkonnost vývojového týmu zvyšovat.

Závěr

Cíle práce se podařilo plně naplnit. Byly identifikovány metriky úspěšnosti týmů, hlavní problémy softwarových týmů, principy úspěšných softwarových týmů i možnosti manažera ovlivnit výkon vývojového týmu.

V rámci naší seminární práce jsme dospěli hned k několika zajímavým závěrům, které se pokusíme na tomto místě shrnout. Každý (nejen vývojový) tým je směsicí lidí různých osobnostních charakteristik. To velmi dobře popisuje dělení osobností v týmu dle Bene a Sheats popsané v 1. kapitole naší práce. O úspěšnosti vývojového týmu a jeho schopnosti doručovat výsledky včas a v adekvátní kvalitě rozhoduje právě schopnost manažera vybrat k sobě kompatibilní osobnostní typy a řídit jejich bezproblémové fungování.

Důležité je také nastavení komunikace v týmu. Ta by měla být pochopitelně otevřená. Úlohou vedoucího vývojového týmu je jí řídit způsobem, aby byli spolu všichni členové schopni komunikovat. Zároveň by však neměl usilovat o nekonfliktnost za každou cenu, která může způsobit snižování výkonnosti týmu a frustraci jeho členů. Každý člen týmu by však měl právo chybovat a možnost učit se ze svých chyb. K tvorbě kolaborativního prostředí také pomohou praktiky jako společné sdílené kódu, díky kterému se jednotliví členové od sebe mohou učit.

Tyto zásady může manažer podpořit také zavedením nástrojů pro podporu týmového managementu. Těmi můžou být projektové platformy jako Basecamp, Apollo, Teamwork apod. Vhodné je tyto platformy doplnit o instant-messenger typu Slack. Tyto podpůrné nástroje mohou být ještě víc přínosné v týmech, kde je rozšířený home office.

Při úspěšném zapracování těchto praktik se dá předpokládat, že vývojový tým se vyvaruje velké části problémů, se kterými se potýkají softwarové vývojové týmy. Jestli však jeho výkon bude pouze uspokojivý, či vynikající, rozhodne však i mnoho dalších obtížně predikovatelných faktorů (osobní motivace jednotlivých členů, konkurence apod.).

Zdroje

Anantatmula, V. S. (2010). Project Manager Leadership Role in Improving Project Performance. *Engineering Management Journal*.

Apollo Team. (2018). *FAQ: Timer*. Získáno 26. 04 2018, z Apollo Help: https://help.apollohq.com/en/faq/timers/

Belbin, M. (1993). Team Roles at Work. *Butterworth Heinemann*.

Captera Inc. (2018). *Project Management Software*. Získáno 19. dubna 2018, z Web Captera Inc.: https://www.capterra.com/project-management-software/

Creasy, T. (2013). From every direction--how personality traits and dimensions of project managers can conceptually affect project success. *Project management journal*.

Faraj, S., & Sproull, L. (2000). Coordinating Expertise in Software. *Management Science,* prosinec 2000, *46*(12), stránky 1554-1568. Načteno z http://www.jstor.org/stable/2661533

Gonçalves, E., & Lopes, E. (2014). Implementing Scrum as an IT Project Management Agile Methodology in a Large Scale Institution.

Hayes, N. (2001). Managing teams. Strategy for success. *Cengage Learning EMEA Higher Education*.

Mezak, S. (2017). The Secrets of High-Performance Software Teams. *BETTER SOFTWARE.* Říjen 2017. Stránky 14 - 16.

Miller, J. (2017). *Principles of Successful Software Teams*. 04. leden 2017. Získáno 11. březen 2018, z SKOOKUM: https://skookum.com/blog/principles_of_successful_software_teams

Miranda, B. (2016). *Principles of Successful Software Engineering Teams*. 16. květen 2016. Získáno 15. duben 2018, z Medium: https://blog.brunomiranda.com/principles-of-successful-software-engineering-teams-41a65bfd56b3

Pavlica, K. (2000). *Sociální výzkum, podnik a management.* Ekopress.

Petter, S. (2009). Developing soft skills to manage user expectations in IT projects: Knowledge reuse among IT project managers. *Project management journal*.

Pronschinske, M. (2018). *Handbooks for change*. 12. duben 2018. Získáno 15. duben 2018, z TechBeacon: https://techbeacon.com/lessons-7-highly-successful-software-engineering-cultures

Rajan, R. S. (2017). *How to be an Insanely SUCCESSFUL Software Manager*. 14. prosinec 2017. Získáno 15. duben 2018, z Medium: https://hackernoon.com/how-to-be-an-insanely-successful-software-manager-13efe08fd890

Slack Ltd. (2018). *Web Slack Ltd.* Získáno 15. duben 2018, z slack: https://slack.com/

Stýblo, J. (2011). Hodnocení výkonnosti zaměstnanců a jeho souvislosti v praxi. Načteno z http://www.mzdovapraxe.cz/archiv/dokument/doc-d34068v43545-hodnoceni-vykonnosti-zamestnancu-a-jeho-souvislosti-v-praxi/