Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologií

Zavedení agilní metodiky vývoje softwaru ve vybrané společnosti

Diplomová práce

Autor: Bc. Pavel Bílek

Studijní obor: Informační management

Vedoucí práce: doc. Ing. Hana Tomášková, Ph.D.

Hradec Králové srpen 2021

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury a uvedených zdrojů.

V Hradci Králové dne 16.8. 2021 Bc. Pavel Bílek

Poděkování:

Děkuji mé vedoucí práce doc. Ing. Haně Tomáškové, PhD. za způsob vedení, vhodné připomínky a značnou trpělivost při psaní této diplomové práce.

Anotace

BÍLEK, Pavel. Zavedení agilní metodiky vývoje softwaru ve vybrané společnosti. Hradec Králové, 2021. Diplomová práce. Fakulta informatiky a managementu, Univerzita Hradec Králové.

Diplomová práce pojednává o využití agilních metodik během životního cyklu vývoje softwaru ve vybrané společnosti. Cílem je navrhnout vhodné techniky a praktiky vycházející z agilních metodik pro odstranění identifikovaných problémů v jednom vývojovém týmu. V teoretické části je představen obor softwarového inženýrství, vodopádový model řízení vývoje a jednotlivé agilní metodiky. Praktická část se skládá ze tří částí. Nejprve je analyzován současný stav způsobu řízení vývoje softwaru ve vybraném týmu mezinárodní společnosti. Tato kapitola je uzavřena seznamem zásadních problémů, které jsou vnímány členy týmu. Další kapitola popisuje důležitost kritérií spolupráce z pohledu zákazníka zkoumaného týmu a aktuální trendy agilního přístupu ve světě. Obě zjištění jsou podkladem pro poslední kapitolu, ve které jsou navrhnuta doporučení pro odstranění hlavních překážek.

Klíčová slova: agilní metodiky, vývoj softwaru, scrum, kanban, scrumban

Annotation

BÍLEK, Pavel. Use of agile methodologies in project management focused on SW development. Hradec Králové, 2021. Diplomová práce. Fakulta informatiky a managementu, Univerzita Hradec Králové.

The diploma thesis deals with the use of agile methodologies during the software development life cycle in a selected company. The aim is to propose appropriate techniques and practices based on agile methodologies to eliminate identified problems in one development team. The theoretical part introduces the field of software engineering, the waterfall model of development management and the agile methodologies. The practical part consists of three parts. First, the current state of software development management in a selected team of an international company is analysed. This section concludes with a list of the major issues that are perceived by the team members. The next chapter describes the importance of collaboration criteria from the perspective of the customer of the team under study and the current trends of agile approach in the world. Both findings form the basis for the last chapter, in which recommendations are proposed to remove the main obstacles.

Keywords: agile methodology, software development, scrum, kanban, scrumban

Obsah

Úvod

Organizace mezi sebou denně bojují o přežití a vzhledem k současné globální konkurenci je nezbytné, aby byly schopné zajistit vývoj nových produktů a služeb ve velmi krátkém čase, zavádět potřebné obchodní procesy a měnit informační systémy v adekvátním čase a kvalitě. A právě z těchto důvodů je tato diplomová práce zaměřena na způsob řízení vývoje softwaru ve vybrané výrobní společnosti. Za cíl si práce pokládá identifikaci hlavních problémů, kterým jeden z vývojových týmů čelí a na základě teoretického poznání vytvořit vhodný návrh řešení, který odstraní nalezené překážky. Další motivací pro výběr tohoto tématu je, že autor práce zároveň působí v prostředí zkoumané organizace a vnímá potřebu změn, které by měl přinést agilní přístup a způsob myšlení o procesech spjatých s vývojem softwarového produktu.

Teoretická část je rozvržena do tří častí. První část popisuje základní poznání oboru softwarového inženýrství, kam životní cyklus vývoje softwaru spadá. V této kapitole se rovněž dozvíme, co stálo za vznikem tohoto oboru a jakými metodikami byl udáván směr v oblasti vývoje softwaru od jeho počátku.

Druhá kapitola popisuje vodopádový způsob řízení vývoje, podle kterého se řídila většina vývojových společností od dob jeho vzniku. Z praxe je známo, že některé firmy se stále tímto způsobem řídí, a i když je tato metodika na ústupu, tak si zaslouží určitý prostor v rámci obsahu práce, neboť některé myšlenky a praktiky mohou být využitelné i v této době.

Hlavní část teoretického úseku tvoří agilní metodiky a jejich využití během vývoje softwaru. Nejprve jsou popsány principy, na kterých je základ agilního vývoje založen. Následně jsou představeny jednotlivé metodiky, které patří mezi nejvyužívanější, a to Scrum, Kanban a Scrumban.

Praktická část je rozdělena do dalších tří kapitol. První analyzuje prostředí zkoumaného SAP týmu a společnosti, kde je tým zodpovědný za vývoj a údržbu softwarového produktu. Výstupem z této analýzy je definice zásadních překážek, kterým tým čelí v rámci životního cyklu vývoje softwaru. Pátá kapitola popisuje kritéria a oblasti, které jsou důležité pro zákazníky týmu. Výstup z tohoto šetření je následně využit pro sestavení vhodného návrhu doporučení, která jsou rozepsány v poslední části práce.

Teoretická část

1 Softwarové inženýrství

Pojem softwarové inženýrství je odvětvím, které se zaměřuje na zavedení a používání inženýrských principů během celého životního cyklu vývoje softwaru (SDLC – *software development life cycle*) tak, abychom na konci cyklu dostali spolehlivý a požadovaný produkt, který si zákazník přál (Kadlec, 2004). Vývoj softwaru je komplexní a empirický problém, jehož největším úskalím je velmi špatná možnost odhadování a plánování dokončení. To jsou základní příčiny vzniku softwarové krize, která stojí za založením oboru softwarového inženýrství (Sommervile, 2013).

1.1 Softwarová krize a historie softwarového inženýrství

Softwarové inženýrství je obor, který se řadí mezi nejmladší z inženýrských odvětví, což je jedním z důvodů, proč zažívá nepřetržitý rozvoj a silné změny (Kadlec, 2004). Zároveň je to i odpovědí na to, proč většina aspektů tohoto oboru není přesně ustanovených. Tím, jak se mění požadavky rozvíjejícího se světa, tak podobně musí reagovat obor zaměřený na vývoj softwaru, aby posouval odvětví kupředu.

První náznaky a snahy o zavedení systematického přístupu při vývoji systémů bylo možné pozorovat na přelomu 60. a 70. let, kdy byl pojem softwarové inženýrství představen na konferenci NATO (Kadlec, 2004). Organizace si uvědomovali, že individuální přístup k vývoji systémů nebude škálovatelný na sofistikovanější produkty, čemuž nasvědčovali i výsledky z realizovaných projektů – neustálé prodlužování dodání produktů, nespolehlivost softwaru, neporozumění požadavkům, a především neúnosný nárůst nákladů na vývoj. Všechny tyto a další nezmíněné faktory vedly k období nazývanému softwarová krize.

Softwarové společnosti zápasí s výše zmíněnými problémy tohoto odvětví odjakživa. Přes 70 % projektů je dodáno se zpožděním, jejich rozpočet přesahuje původní odhad a není výjimkou, že na konci zákazník obdrží něco, co vůbec nechtěl (Šochová & Kunce, 2019). Období krize započaté koncem 60. let postihlo tento inženýrský obor, kdy s ohledem na úroveň technologií a zkušeností, bylo zřejmé, že zakázky nebudou realizovány v potřebné kvalitě (Kadlec, 2004). Pro ilustraci krize lze využít data z její doby týkajících se zakázek americké vlády. Přes 40 % projektů bylo dodáno, ale nikdy nebylo úspěšně nasazeno a používáno. Dalších 25 % bylo zaplaceno, ale nikdy nedokončeno, a asi 15 % projektů bylo po různých modifikacích úplně zrušeno. Pouze kolem 5 % procent zakázek lze považovat

za svým způsobem úspěšné. Obor softwarové inženýrství vznikal tedy začátkem 70. let, ačkoli oficiálně byl uznán s certifikátem až v roce 1997 v USA (Kadlec, 2004).

1.2 Metodiky vývoje softwaru

V předchozí kapitole byly popsané hlavní příčiny, které způsobily softwarovou krizi, která proběhla v podstatě na samém počátku tohoto oboru. Vzhledem k tomu, že od té doby již uběhly desítky let, mohlo by se zdát, že všechny tyto problémy byly vymizeny a že skrze raketový rozvoj odvětví byly vytvořeny propracované inženýrské postupy a metodiky, které nám zajistí vysokou úspěšnost při realizaci softwarového řešení během všech fází vývoje softwaru. Avšak toto tvrzení nelze potvrdit ani v této době. Softwarové společnosti se i nadále potýkají s občas neúspěšnými, zpožděnými či zbytečně prodraženými projekty (Kadlec, 2004).

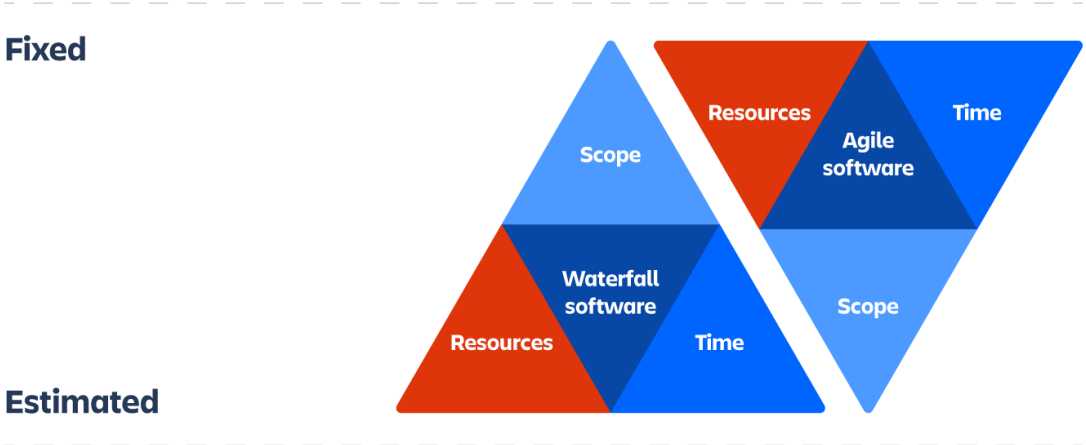
Když došlo k velkému progresu v rámci softwarového inženýrství, proč se organizace potýkají stále s neúspěšnými zakázkami? Důvod je ten, že paralelně s rozvojem vývojových metodik roste i složitost, náročnost a rozsah projektů. Zároveň s tím nároky na rychlé dodání a kvalitní produkt to ještě více komplikují. Pro minimalizaci rizika selhání musí softwarové společnosti najít vhodné metodiky a osvědčené postupy vývoje softwaru. A protože každý projekt je jiný, nelze se ani spolehnout na využití pouze jedné metodiky, ale kolikrát je vhodné až žádoucí metodiky kombinovat či modifikovat s ohledem na prostředí společnosti a další faktory. Neměli bychom tedy mít metodiky za dogmata, ale spíše průvodce procesem, který by nás měl dovést k tíženým výsledkům (Myslín, 2016).

Odvětví softwarového inženýrství je stejně jako celá oblast informačních technologií velice pokrokové a živé. Díky tomu mohou společnosti čerpat z mnoha teoretických poznatků a implementovat do svých projektů pro ně nejvhodnější kombinaci metodik. Dostáváme se k již ke konkrétním metodikám, které vznikaly a vznikají od počátku softwarového inženýrství. Jednotlivé metodiky, které byly postupně definovány, odrážely aktuální potřeby v daném čase (Kadlec, 2004).

Vzhledem k široké paletě existujících metodik vývoje softwaru budou v rámci práce představené pouze metodiky, které budou mít návaznost na praktickou část. Ostatní metodiky budou pouze zmíněné v této části bez hlubšího teoretického detailu.

Do první kategorie, kterou si představíme, patří tradiční metodiky. Jejich vznik měl být odpovědí na dříve vzniklou softwarovou krizi. U těchto metodik byl kladen velký důraz na přípravné fáze, tedy analýzy a specifikace projektů. Do těchto metodik patří jeden z nejznámějších, a i dnes používaný Vodopádový model životního cyklu, dále Spirálový model životního cyklu či metodika Rational Unified Process. Z těchto tří metodik bude dále více rozebrán pouze první zmíněný – Vodopádový model (Kadlec, 2004).

Metodiky z druhé kategorie vznikly za účelem dodávat softwarový produkt rychleji a v požadované kvalitě, tak aby byly vyhověno požadavkům dnešní doby. Jde o Agilní metodiky vývoje softwaru, které přinášejí vývojovým společnostem větší flexibilitu, efektivitu, předvídatelnost, kvalitu a zábavu v rámci životního cyklu vývoje produktu (Šochová, a další, 2019). Do této kategorie spadají metodiky jako Scrum, Kanban, Lean nebo Extreme Programming, ale protože komplexita se neustále zvyšuje a společnosti se skládají z více rozdílných týmů, je nutné zajistit škálovatelnost těchto agilních metodik. K tomu nám mohou pomoct metodiky jako například Scrum of Scrum nebo Large-Scale Scrum.



Obrázek 1 - Trojúhelníkový model projektového řízení

Převzato z: https://www.atlassian.com/agile/agile-at-scale/agile-iron-triangle

Rozdílnost mezi výše zmíněnými kategoriemi metodik lze dobře pozorovat na obr. 1, který srovnává a zobrazuje základní omezení se kterými musí softwarová společnost pracovat v rámci projektu. Problém je v tom, že není možné změnit jedno omezení, aniž by to nemělo dopad na ostatní.

Původní trojúhelník zobrazen na levé straně obr. 1, který byl navržen v roce 1969, vychází z vodopádového přístupu k vývoji produktu. Ten byl postaven na předpokladu, že všechny zákazníkovy požadavky budou detailně vyspecifikovány na začátku životního cyklu projektu, tudíž toto omezení je bráno jako fixní. Zbylá omezení, tedy čas dodání a cena jsou v tomto případě domluvené spíše na úrovni pouhého odhadu, kdy a za kolik bude projekt možná dodán. Již z popisu prvního omezení je zřetelné, že tento přístup bude velmi náročně aplikovat na velké a komplexní softwarové projekty, neboť v praxi není jednoduché ve fázi analýz a specifikací pokrýt všechny možné aspekty projektu, tak aby nedošlo následně k nějakým změnám. A to je právě slabinou tohoto přístupu, neboť projektové týmy se řídí původním návrhem a nebere zřetel na případné návaznosti, na což doplatí produkt menší kvalitou. Následně se předá dokončený produkt zákazníkovi, který měl ale úplně jiná očekávání, a tak se dostanou společně zpět do fáze specifikace a celý cyklus jede znovu, dokud klient není spokojený. Tímto způsobem se projekt prodlouží a prodraží, proto jsou tato omezení brána jako variabilní (Aljaber).

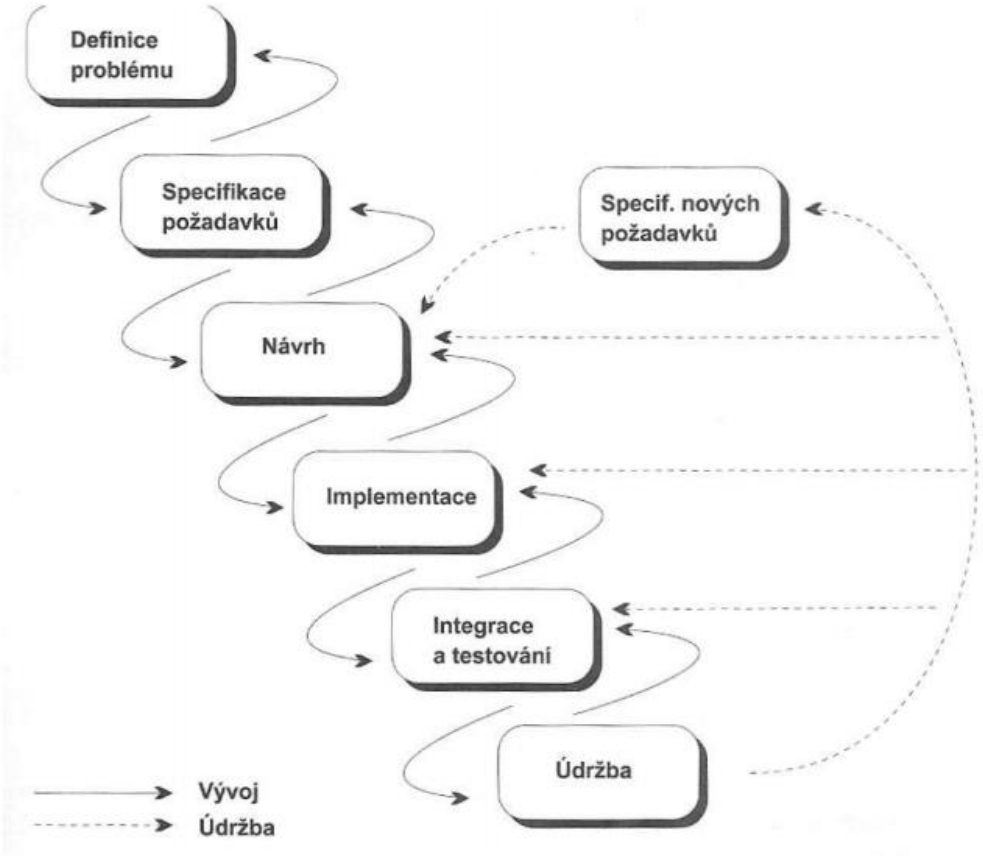
Agilní metodologie přistupují k těmto omezením přesně naopak. Termín dodání a zdroje jsou fixní, pouze rozsah dodávaných funkcionalit zůstává variabilní. Znamená to tedy, že softwarová firma se zavazuje dodat produkt ve smluveném čase a za schválený rozpočet, ale výsledné funkcionality se mohou lišit od toho, co se na začátku projektu odhadovalo či požadovalo. Vychází to z toho, že v agilním světě je kladen důraz na spolupráci a komunikaci se zákazníkem v krátkých časových iteracích, tak aby byl zákazník vtažen do celého životního cyklu vývoje jeho softwaru. Díky tomu dokáže vývojový tým reagovat včas na požadované změny a upevňovat se v tom, že pracují pouze na tom, co má smysl. Agile sice neznamená žádný striktní proces, ale není to ani chaos. Metodiky určují jasná pravidla a definují hranice, v jejichž rámci si vývojové týmy mohou určovat svá vlastní pravidla tak, aby byly co nejvíce produktivní, efektivní a dodali kvalitní produkt v co nejkratším čase (Šochová, a další, 2019).

2 Vodopádový model řízení vývoje

Vodopádový model patří do první skupiny metodik vývoje, tedy do tradičních. Název modelu je odvozen z posloupnosti jednotlivých fází, které jsou v modelu vyobrazeny tak, že na sebe postupně navazují směrem dolů. Všechny fáze jsou za sebou logicky seřazeny z toho důvodu, že jednotlivé fáze jsou závislé na výstupech fáze předchozí. Model byl vytvořen již v roce 1970 a v době svého vzniku představoval zásadní převrat, jak přistupovat k vývoji softwaru (Kadlec, 2004).

Základním principem pro využití vodopádového modelu pro vývoj softwaru je následování lineárního, sekvenčního postupu skrz jednotlivé fáze procesu, přičemž podmínkou pro přechod do fáze další je dokončení fáze předchozí. Není tedy možné mít rozpracováno více fází projektu najednou. Jednou z výhod tohoto modelu je to, že vždy všechny osoby zainteresované v projektu vědí, v jaké fázi se právě nachází. V případě, že je počátečním fázím věnován dostatek času a kapacit, lze předpokládat, že výsledný produkt nemusí být katastrofou, neboť bychom tím mohli eliminovat potřebu opravovat následné chyby a nedostatky. Samozřejmě do toho musí vstupovat i další faktory jako je třeba velmi dobře nadefinovaný seznam požadavků od zákazníka. Bohužel tento model v původním znění má nedostatky v chybějících průběžných iteracích, kde lze dodávat funkcionality postupně na což se váže komunikace se zákazníkem. Protože model byl představen již v roce 1970, tak je zřejmé, že během uběhlé doby už existuje spousta modifikací tohoto modelu. Tyto modifikace se budou lišit především v konkrétní množině fází se kterými budou v daném projektu pracovat (Sommervile, 2013). V práci se ale zabýváme pouze původní definicí.

Na obr. 2 lze vidět všechny fáze životního cyklu vývoje softwaru, které definuje vodopádový model. Z pohledu na diagram je jasné, proč se modelu říká vodopádový. Zároveň si můžeme všimnout, že v modelu jsou vyobrazeny zpětné šipky. Jejich význam je, že v případě, že v nějaké fázi narazíme na zjevný a nerealizovatelný problém, lze aktuální fázi pozastavit a vrátit se o jeden krok zpět, aby byla předchozí fáze zrevidována a modifikována. Situaci si lze představit na příkladě, kdy vývojový tým již realizuje zadání z fáze návrhu, ale zadrhnou se na nějaké části, která je nerealizovatelná. Tým se tedy vrátí o fázi před, dojde k redefinování návrhu, který následně musí být znovu zdokumentován a schválen, než bude možné zase přejít k implementaci (Kadlec, 2004).



Obrázek 2 - Základní struktura vodopádového modelu

Zdroj: (Kadlec, 2004, str. 57)

Základní fáze modelu:

1) **Definice problému, poznání zákazníka –** jde o první styk dodavatele se zákazníkem, kdy se softwarová firma snaží co nejlépe poznat prostředí zákazníka, aby byla schopna pochopit všechny požadavky a přání správným způsobem. V průběhu této fáze může ideálně dodavatel navštívit poptávající firmu, projít si společně jejich interní procesy, tak aby mohli pochopit, jakým způsobem jim má nový systém nejlépe pomoci a co vlastně zákazník potřebuje. Výstupem z této fáze je dokument – Úvodní studie, který shlukuje všechny zjištěné informace o zákazníkovi, o jeho potřebách a motivaci, proč by měl být systém realizován (Sommervile, 2013).
2) **Analýza a specifikace požadavků –** v této fázi se již analytik zaměřuje na detailní sepsání všech požadovaných funkcionalit a jeho výstupem by měl být dokument Specifikace požadavků, podle kterého se následně bude tým řídit v další fázi designu systému. Popis požadavků by měl být exaktní, vydefinován pomocí přesných termínů, ale psán v jazyce, kterému zákazník porozumí. Měl by se tedy vyhnout

programátorskému žargonu a různým IT zkratkám. Důvodem je to, že takovýto dokumentu musí být na konci fáze přezkoumán a schválen zákazníkem. Pokud by zákazník nerozuměl obsahu dokumentu, dojde velmi snadno k nespokojenosti při předání hotového produktu. Formu takového dokumentu definuje spousta připravených standardů, podle kterých se softwarové společnosti mohou řídit. Dokument by neměl obsahovat žádné informace o tom, jak mají být požadavky implementovány z technického hlediska. To vše je až součástí následující fáze (Kadlec, 2004).

3) **Návrh systému –** tvorba softwarové architektury vzniká v této fázi na základě dokumentu z předchozí fáze. Architekti procházejí a zkoumají jednotlivé požadavky podle kterých vytvářejí vhodnou architekturu systému. Tím je myšleno zvolení vhodných technologií, rozdělení aplikace do funkčních modulů a vrstev, definování, jak budou jednotlivé moduly mezi sebou komunikovat, jakým způsobem budou získávána a uchovávána data, a další aktivity spojené s touto fází. Konkrétní postupy pro realizaci této fáze je možné převzít z existujících konkrétních metod, které jsou pro tento účel vytvořeny (Sommervile, 2013).
4) **Implementace systému –** ve fázi implementace dochází k samotnému kódování, tj. tvorbě zdrojového kódu, které realizují programátoři na základě předchozích dokumentů. Z výstupů, které vývojový tým obdrží z předchozích fází, by měl být tým schopen implementovat systém bez jakýchkoliv pochybností. Důležité je, že se tým musí řídit přesné specifikace a neměl by si vymýšlet jiné funkcionality než to, co je po něm požadováno ve schváleném dokumentu. V případě, že by narazil na potřebu něco změnit, je nutné se vrátit zpět do předchozí fáze, nechat změnit návrh a ten následně schválit (Kadlec, 2004).
5) **Integrace a testování systému –** jak název fáze říká, jde o testování veškerých modulů a funkcionalit, které se v ideálním scénáři nejdříve testují separátně a následně se otestují z integrované funkcionality do jednoho kompletního systému. Cílem fáze je ověření, že implementovaný systém dělá přesně to, co zákazník požadoval. Postupů na testování existuje spousta, zmínit můžeme dva – *white-box* a *black-box.* První metoda navrhuje testovat funkce podle vnitřní struktury kódu, tj. otestovat všechny cesty v programu). Druhá metoda říká, ať testujeme funkce z pohledu logiky, tj. ověřování očekávané chování funkcionality vůči výsledkům. Z praxe víme, že tato

část bývá kolikrát ošizena, proto se neobejdeme bez další fáze, kde lze neodhalené chyby opravit (Kadlec, 2004).

6) **Provoz a údržba –** v případě, že zákazník akceptoval vyvinutý produkt a chce ho nechat nasadit na produkčním prostředí, tak tým transportuje vývoj a produkt předává. Tímto ale práce nekončí. Jak již bylo zmíněno výše, nejen, že je nutné být zákazníkovi po ruce v případě odhalení chyby, která nebyla objevena během fáze testování, ale je i vhodné nadále systém udržovat a vylepšovat, neboť technologie jdou neustále kupředu a nikdo nemůže dobrovolně chtít, aby veškerá snaha při vývoji systému přišla vniveč kvůli tomu, že nebude systém nadále udržován (Sommervile, 2013).

2.1 Výhody vodopádového modelu

Hlavní výhodou tohoto modelu je jeho jednoduchost, což přináší další benefity především pro projektové manažery, kterým se snáze řídí projekty. Vždy totiž všichni vědí, v jaké fázi se zrovna nachází a jaké aktivity v dané fázi mají být provedené. Tím, že jde o první model vývoje softwaru, tak je tento model nejvíce rozšířený a měl by ho znát opravdu skoro každý, kdo se v IT světě pohybuje. Využitelnost tohoto modelu je u rozsahově malých, specifických a specifikačně jednoduchých projektů (Kadlec, 2004).

2.2 Nevýhody vodopádového modelu

Nevýhodou aplikace vodopádového modelu je, že je neflexibilní, což je zásadní nedostatek u rozsáhlejších a komplexnějších projektů, kde není výjimkou, že se tým potřebuje vracet zpětně do předchozích fází. S ohledem na to, že v praxi je zcela běžné pozorovat příchozí změny na funkcionality od zákazníka, je opět tento model spíše nevyhovující. Dalším vážným problémem je nedostatek komunikace a získávání zpětně vazby od zákazníka. V podstatě jediná spojení zadavatele s dodavatelem jsou na začátku projektu a ke konci, nic není v průběhu vývoje. To vyústí k možným nepochopením požadavků analytikem, který dává dohromady úvodní dokumentaci, podle které se bude dále ubírat směr vývoje produktu. Výsledkem pak bude nechtěný produkt a nespokojený zákazník.

3 Agilní metodiky řízení vývoje

Agilní metodiky přinášejí jiný náhled a přístup k vývoji softwaru. Agilní znamená flexibilní, přizpůsobivý, dynamický, interaktivní, iterativní, hravý, rychle reagující na změnu a spousta dalších synonym. Být agilní neznamená využívat pouze předepsané metodiky a frameworky, ale ideálně celá firma by měla žít agilní filozofií a mít správně nastavenou firemní kulturu. Firmám, které se mají v plánu transformovat na agilní, tedy nepomůže žádný přesně předepsaný seznam kroků, které proto musí udělat (Holbeche, 2018). Zároveň to, že implementují některou z agilních metod do svých interních procesů, z nich nedělá agilní firmu. Důležité je pochopit principy agility. Musíte tak myslet, žít a chovat se. Takové týmy jsou zaměřené na dodávání největší hodnoty pro zákazníka, aby dodávaly produkt rychle, ověřovaly si funkčnost vyvinutých inkrementů a priority v krátkých iteracích. To vše je postavené na vysoké míře spolupráce a komunikace mezi týmy. Díky tomu je umožněno vytvářet změny a rychle na ně reagovat (Šochová, a další, 2019) (Shore, a další, 2008).

Pojem agilní metodiky se začal objevovat už v 90. letech minulého století ačkoli první oficiální podobu dostal v roce 2001, kdy se sešlo sedmnáct předních softwarových vývojářů v Utahu, aby diskutovali o způsobech, které využívají. Probírali jejich pohledy a zkušenosti s vývojem softwaru v časech, kdy selhávající vodopádový přístup začínal být nahrazován metodikou RUP (Rational Unified Process). A protože ani tato metodika nepřinášela výrazně lepší výsledky bylo podle vývojářů nutné přijít se změnou. Zmínění vývojáři měli rozdílné zkušenosti s agilními metodami jako je Scrum, eXtreme Programming, Crystal, Feature Driven Development, Adaptive Software Development a spoustu dalších. Výstupem z jejich diskuzí vzešel na svět Agilní manifest (Verheyen, 2013).

3.1 Agilní manifest

Jak již bylo zmíněno výše v práci, agilní manifest vznikl v roce 2001 a jeho cílem je pomáhat pochopit myšlenky, které jsou základem agilního přístupu. Manifest obsahuje čtyři výroky, které tvoří základ pro agilní vývoj softwaru. Následně byl vytvořen doplňující seznam dvanácti principů, kterých by se agilní společnosti měli držet v případě vyznávání agilní filozofie a řízení vývoje softwaru.

Manifest (The Agile Manifesto, 2001):

„Objevujeme lepší způsoby vývoje software tím,
že jej tvoříme a pomáháme při jeho tvorbě ostatním.
Při této práci jsme dospěli k těmto hodnotám:

- **Jednotlivci a interakce** před procesy a nástroji
- **Fungující software** před vyčerpávající dokumentací
- **Spolupráce se zákazníkem** před vyjednáváním o smlouvě
- **Reagování na změny** před dodržováním plánu

Jakkoliv jsou body napravo hodnotné,
bodů nalevo si ceníme více.“

Jednotlivci a interakce před procesy a nástroji:

Manifest poukazuje na to, že je důležitější a přínosnější, aby týmy složené z kompetentních lidí dokázaly co nejefektivněji spolupracovat dle svého uvážení, než že by musely dodržovat striktně procesy a nástroje, které by jím říkali, jak to dělat. Na druhou stranu to neříká ani to, že by žádné procesy a nástroje neměli existovat. Jen by měly týmy mít možnost si vybrat takové nástroje, které jim budou pomáhat v dosahování správných výsledků (Šochová, a další, 2019).

Fungující software před vyčerpávající dokumentací:

Dobrá dokumentace je důležitá, ale neměla by mít převahu nad samotným produktem a jeho funkčností. Manifest říká, že je dobré dokumentací pokrýt oblasti, které nejsou intuitivní anebo stojí za to, aby byli zdokumentované například pro další generace. Popřípadě pokud si společnost určí další případy, kdy dokumentace bude přínosná, tak není proti ničemu dokumentaci vytvořit. Efektivnější ale bývá psát rovnou stručnější dokumentaci do kódu (Šochová, a další, 2019).

Spolupráce se zákazníkem před vyjednáváním o smlouvě:

Je samozřejmostí, že by měla mezi zákazníkem a dodavatelem existovat smlouva či dohoda o spolupráci, ale neměla by být prostředkem, který nahradí vzájemnou komunikaci. Při tvorbě takovéto smlouvy bychom se měli zamyslet nad tím, jak spolupráce pravděpodobně bude vypadat a přiblížit její obsah co nejvíce realitě. Cílem je, aby zákazník dostal takový produkt, který pro něj a jeho byznys bude mít přínos (Šochová, a další, 2019).

Reagování na změny před dodržováním plánu:

Technologie se neustále mění a s nimi se mění i požadavky a problémy klientů softwarových společností. Nemusí jít pouze o konkurenční boje a s nimi spojené vynucené změny na požadavky, ale zákazník si může v různých fázích životního cyklu vývoje systému rozmyslet, že původně žádaná funkcionalita mu vlastně neřeší problém, který má a místo toho by chtěl nechat vyvinout něco jiného. Toto není něco výjimečného v reálném světě, a proto manifest doporučuje, aby se týmy dokázaly přizpůsobit těmto potřebám spíše, než aby se striktně držely původního plánu, který by měl sloužit pouze jako jakési vodítko (Šochová, a další, 2019).

Rozšiřující seznam principů stojících za Agilním manifestem (The Agile Manifesto, 2001):

1) Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
2) Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
3) Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
4) Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.
5) Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
6) Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
7) Hlavním měřítkem pokroku je fungující software.
8) Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
9) Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
10) Jednoduchost – umění maximalizovat množství nevykonané práce – je klíčová.
11) Nejlepší architektury, požadavky a návrhy vzejdou ze samoorganizujících se týmů.
12) Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti.

3.2 Lean Development

Metodika Lean Development vznikla na základě již existující metody Lean Manufacturing, jenž má kořeny v automobilovém průmyslu, kdy v 80. letech minulého století byla tato metoda vytvořena s cílem eliminovat vše zbytečné v rámci výrobního procesu a dělat pouze to, co přináší další hodnotu. Za vznikem této metody stojí Taichi Ohno, který pracoval pro japonskou automobilku Toyota (Kadlec, 2004).

Lean přeložíme jako štíhlý, ale významově znamená, abychom se v rámci našich procesů zaměřovali pouze na ty věci, které jsou potřeba a mají nějakou hodnotu. Jde spíše o přístup než striktní popis procesu, jak vývoj provádět. Je nutné chápat principy lean metodiky, neboť se prolínají s principy agilu (Šochová, a další, 2019).

3.2.1 Principy Lean Developmentu

Metodika je založena na několika principech, které naznačují a popisují, jakým způsobem by měl vývoj probíhat, čím bychom se měli řídit a kam by měl vývoj směrovat. Metodika
je založena na následujících sedmi principech (Kadlec, 2004).

1. **Eliminace plýtvání –** vyvarovat se pracím na něčem, co se na konci nevyžije, v horším případě vyhodí. Raději investovat tento čas do věcí, které mají smysl a přínos.
2. **Rozvinout učení –** využívat zpětnou vazbu, soustředit se na krátké iterace, tak aby se lidé co nejvíce učili a posouvali kupředu.
3. **Pozdní rozhodování –** vykonávat rozhodnutí co nejdéle to jde, neboť o to více informací máme pro vykonání správného finálního rozhodnutí.
4. **Časté dodávky –** čím dříve dokončíme iteraci a dodáme produktu zákazníkovi, tím dříve obdržíme zpětnou vazbu, kterou lze zohlednit v následující iteraci.
5. **Pravomocní pracovníci –** nechat zodpovědnost na týmu je zásadní pro zajištění zdravé motivace.
6. **Integrita –** nejde pouze o integritu dodávaného softwaru, ale i o integritu vývojového procesu, který by měl plynout přirozeně od vzniku požadavků až k výsledném řešení. Zaměřit bychom se měli na anomálie vzniklé během životního cyklu vývoje softwaru. Dále je kladen důraz na testování a refaktorizaci.

7. **Vidění celku –** pro zajištění úspěchu výsledného projektu je vhodné přemýšlet dopředu, neřešit nepodstatné malé chyby a selhání, ale rychle se z nich poučit.

Jednou z metodik, které aplikují principy a myšlenky leanu na softwarový vývoj je Kanban. Tato metodika se ale opírá i o agilní zásady, a proto se jí bude práce zabývat v jedné z dalších kapitol (Šochová, a další, 2019) (Kadlec, 2004).

3.3 Extreme Programming

Extreme Programming (XP) je metodika agilního vývoje softwaru, jejímž cílem je vyšší kvalita softwaru a vyšší kvalita života vývojového týmu. XP je nejkonkrétnější z agilních metodik, co se týče vhodných postupů pro vývoj softwaru. Metodika byla vytvořena Kentem Backem v roce 1999 a zaměřuje se na věci, které když se osvědčily, tak proč je nepraktikovat stále dokola. XP podporuje týmovou spolupráci a staví okolo produktů samoorganizované týmy, které stojí na pěti hodnotách definovaných metodikou (Agile Alliance) (Šochová, a další, 2019).

3.3.1 Hodnoty XP

- **Jednoduchost –** říká, že by se vývojové týmy měly zaměřovat pouze na to, co je v dané situaci požadované aplikovat a co bude funkční. Účelem je se vyhnout plýtvání a dělat pouze naprosto nezbytné věci. Pod tím si lze představit například to, že při analýze nových požadavků a následně přípravě návrhu přemýšlíme co nejjednodušeji pouze o pokrytí aktuálních požadavků, a nebudeme se zabývat možnými požadavky, které by mohly přijít v budoucnu, tzn. nebudeme připravovat návrh pro požadavky, o kterých nevíme, zda budou někdy realizovány.
- **Komunikace –** XP si zakládá na komunikaci mezi všemi členy vývojového týmu a zákazníkem. I z tohoto důvodu XP dále definuje projektovou roli tzv. kouče, který se zaměřuje na hledaní výpadků v komunikaci a následné obnovení vtahů mezi členy týmu.
- **Zpětná vazba –** týmy praktikující XP by měly dodávat produkt v co nejkratších iteracích, aby se jim dostávalo co nejvíce zpětně vazby na jejich odvedenou práci a mohli následně identifikovat oblasti, které je třeba v dalších iteracích zlepšit a upravit.
- **Odvaha –** metodika vyžaduje odvahu od všech členů týmu. Znamená to, že se vývojáři či testeři ozvou v případě, kdy narazí na nějaký problém, že tyto

problémy budou nazývat pravými jmény a nebudou je zametat pod koberec. Je nutné, aby se členové týmů nebáli otevřeně mluvit o jakýchkoliv věcech, které by šli dělat jinak a lépe.

- **Respekt –** členové týmu se musí vzájemně respektovat, aby spolu mohli dobře komunikovat, spolupracovat, poskytovat a přijímat zpětnou vazbu. Každý člen řeší problémy ostatních. Management nechává zodpovědnost za organizaci týmu v rámci týmu.

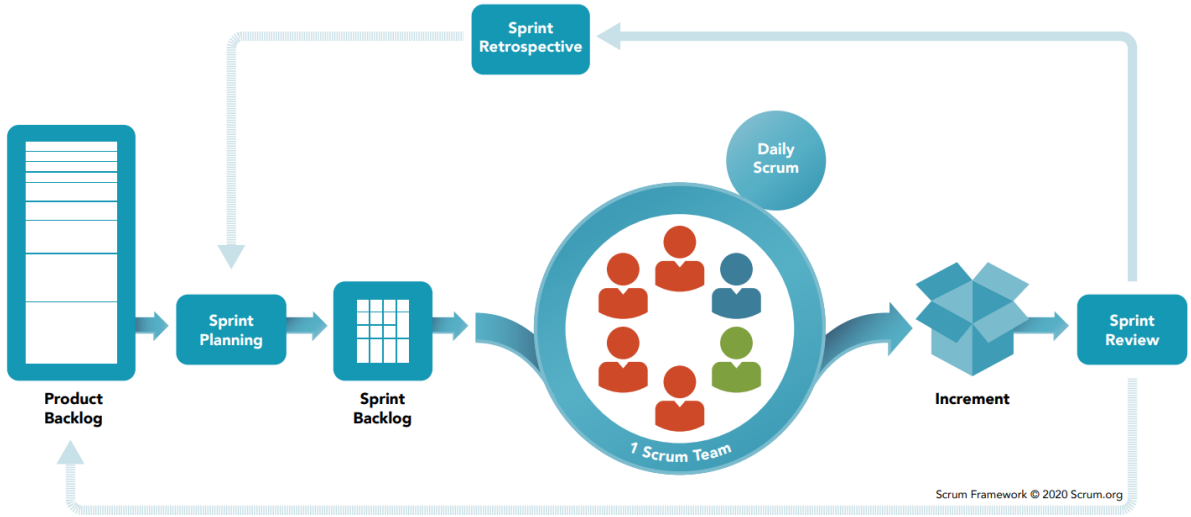
Těchto pět základních hodnot udává ideový rámec XP metodiky, nicméně v takto obecné rovině by to bylo nedostačující pro definování použitelné metodiky. Z tohoto důvodu Kent Beck definoval dvanáct konkrétních postupů, které by měly vést vývojové týmy k vytváření kvalitnějších softwarových produktů. Jde o tyto praktiky: Plánovací hra, Malé verze, Metafora, Jednoduchý návrh, Testování, Refaktorizace, Párové programování, Společné vlastnictví, Nepřetržitá integrace, Čtyřicetihodinový pracovní týden, Zákazník na pracovišti a Standardy pro psaní zdrojové textu. Výše uvedené postupy mohou fungovat pouze v případě, když jsou dodržovány všechny (Beck, 2002) (Kadlec, 2004).

Vhodnost implementace XP nemá jasné hranice, avšak metodika má obecná doporučení, která říkají, že XP lze dobře praktikovat v týmu s dvěma až deseti členy, kteří se často potýkají s měnícími a nejasnými požadavky. Pro širší týmy bude pravděpodobně náročné dodržovat hodnoty vycházející z metodiky. Obecně se dá říct, že se XP nehodí pro týmy, kde není možné dodržování výše zmíněných hodnot, například pokud manažeři nemají důvěru v tým nebo pokud vývojáři nemají dostatek odvahy zmínit se o své či někoho jiného chybě.

3.4 Scrum

Scrum je jedním z nejvyužívanějších frameworků ve společnostech, které se chtějí stát agilními (Šochová, a další, 2019). Framework pomáhá organizacím vytvářet a udržovat komplexní softwarové produkty v turbulentních podmínkách prostřednictvím samoorganizovaných týmů (Verheyen, 2013). Zároveň díky svým procesům umožňuje týmům vytvářet prostředí, kde se mají členové týmu organizovat sami, a to především díky důrazu na komunikaci v týmu, a i směrem k zákazníkovi (Verheyen, 2013).

Na obr. 3 lze vidět schéma, které popisuje, jakým způsobem probíhá cyklus vývoje softwaru v rámci jednoho týmu. Scrum se neskládá pouze z komponent vyznačených ve schématu, ale jde i o další pravidla, praktiky, artefakty, ceremoniály a role, které budou blíže rozebrány v dalších podkapitolách.



Obrázek 3 - Scrum Framework

Převzato z: https://www.scrum.org/resources/scrum-framework-poster

3.4.1 Scrum role

Scrum má vydefinované jednotlivé role pro jasné rozdělení povinností a odpovědností, díky kterým je následně využívání frameworku zprostředkováno. Role a jejich popis budou dále rozebrány.

Scrum Master

Pro zajištění funkčnosti Scrumu je vydefinována role Scrum Mastera, jehož cílem je podpořit samoorganizaci týmu tak, aby se stal vysoce výkonný a efektivní. Scrum Master by měl být pro členy týmu koučem, facilitátorem či servant leaderem, což znamená, že nejde o direktivní přístup manažera, ale o snahu pochopit a upřednostnit potřeby členů týmu. Role Scrum Mastera by se v ideálním případě neměla vůbec kombinovat s žádnou jinou rolí, aby nedocházelo ke střetu zájmů, neboť tím by mohlo dojít k poškození fungování samotného týmu a jeho efektivity (Šochová, 2018).

Product Owner

Product Owner je definován jako vlastník produktu, který vývojový tým dodává. Zaštiťuje pohled obchodní perspektivy týkajících se potřeb a požadavků na výsledný software, které konsoliduje a předává týmu k realizaci. Podstatné pro tuto roli je, že je součástí Scrum týmu a je po něm chtěno, aby se aktivně zapojoval do interakcí s ostatními členy týmu na pravidelné bázi. Hlavním úkolem Product Ownera je zajistit existenci Produktového Backlogu skrz definování produktové vize a její transparentní komunikaci skrz všechny zainteresované strany. Mezi jeho další činnosti patří určování priorit, rozhodování o tom, které funkcionality budou realizované a které nikoli. Na rozdíl od Scrum Mastera není vyžadováno, aby Product Owner trávil většinu svého času s vývojovým týmem, ale naopak, aby naslouchal zákazníkovi, jaké jsou jeho potřeby a díky tomu mohl objektivně rozhodovat, co má nejvyšší hodnotu. Mezi žádoucí dovednosti patří dobrá komunikace a silné znalosti produktu (Šochová, a další, 2019).

Vývojový Tým

Samořídící se vývojový tým, skládající se obvykle z tří až devíti členů, má za cíl podávat maximální výkon během všech aktivit spojených s vývojem na položkách vycházejících z Produktového Backlogu tak, aby dodával použitelný softwarový inkrement každý sprint. Měřícím kritériem pro tvrzení, že je dodaná část použitelná, vychází z tzv. ‚definition of done', jenž říká, za jakého stavu je položka z backlogu hotová. Mezi důležité předpoklady pro samořídící se tým patří mít společný cíl, porozumění zákazníkovi a jeho potřebám, mít stejnou vizi a důvěru v ostatní členy týmu ale i k zákazníkovi. Tým by měl táhnout za společně za jeden provaz a v případě, že někdo selže, neměli by se pídit po tom, čí to byla chyba ale převzít kolektivní zodpovědnost. Mezi hlavní zodpovědnosti členů týmu patří tvorba plánu pro nadcházející Sprint tzv. Sprint Backlog, dodržování kvality podle definition of done a přizpůsobení plánu podle cíle Sprintu (Scrum).

Při takovém nastavení mentality v týmu může dojít k překročení pravomocí a z toho důvodu je vhodné si pojmenovat to, o čem členové týmu nerozhodují. Členům týmu nepřísluší rozhodovat o tom, zda by měli využívat frameworku Scrum či nikoli, natož že by určovali, kdo v týmu smí či nesmí pracovat. Nesmí rušit ani ceremoniály, které Scrum definuje a na kterých stojí. V neposlední řadě týmu nepřísluší rozhodovat o tom, co se bude nebo nebude realizovat, aniž by to nejdříve neprošlo skrz Product Ownera (Šochová, a další, 2019).

V opravdu ideálně složeném Scrum týmu by nemělo jít pouze o již zmíněnou samoorganizaci, ale jednotliví členové by si měli být i vzájemně zastupitelní. To znamená, že například expert na databáze se nemusí starat pouze o aktivity spojené s databázemi, ale může vykonávat například i práce vývojáře. Samozřejmě jistě není reálné, aby každý rozuměl stoprocentně všem disciplínám, ale je vhodné, když všichni členové mají jistý přesah, neboť se tím značně zvýší flexibilita a efektivita spolupráce v rámci týmu (Šochová, a další, 2019).

3.4.2 Scrum události

Scrum framework má vydefinované události, které jsou časově omezené, určují pravidelnost a minimalizují potřebu svolávat další schůzky mimo již vydefinované v rámci Scrumu. Říkají, co by v danou chvíli měl tým dělat a jaký by měl být výsledek daného jednání. Dále dávají jistá doporučení, jak by tyto události měli probíhat.

Sprint Planning

Každý sprint začíná Sprint Planningem, kde vývojový tým společně s Product Ownerem definují plán na nadcházející Sprint, tedy vybírají se User Stories z Produktového Backlogu tak, aby byla implementace vybraných User Stories proveditelná a tým mohl dodat produktový inkrement. Maximální délka Sprint Planningu je osm hodin pro Sprinty, které fungují na měsíční bázi. Výstupem z této události by mělo být vydefinování cíle sprintu, seznam členů týmů a jim přiřazených User Stories, ke kterým se zavázali na daný Sprint, Sprint Backlog a naplánovaný termín pro Daily Scrum meetingy, které jsou součástí Sprintu (Kniberg, 2007).

Daily Scrum Meeting

Daily Scrum neboli Standup meeting je určen ke kontrole progresu v rámci Sprintu, zda se tým blíží ke Sprint Goalu ke kterému se zavázal při plánování Sprintu. Jde o pravidelný denní schůzky týmu, kde si členové sdílejí informace o aktivitách, na kterých strávili čas předchozí den, na čem budou pracovat dnes a zda čelí nějakým problémům. V případě, že se objeví nějaké problémy, tak tento meeting není určen k jejich řešení, pouze jde o to, aby všichni byli informování o tom, co se děje, ale návazná diskuze na hledání řešení problému bude probíhat mimo tuto schůzku. Meeting je časově omezený na patnáct minut a neměl by přesahovat tuto délku. U týmů, které čerstvě začínají praktikovat Scrum je vhodné,

aby byly i tyto krátké schůzky moderovány Scrum Masterem, ačkoli jeho dlouhodobým cílem je do těchto schůzek příliš nezasahovat, pouze pozorovat a naslouchat ostatním členům týmu.

Retrospektiva

Jde o schůzku, kde si členové týmu navzájem dávají zpětnou vazbu s cílem zvýšení kvality a efektivity. Retrospektiva by měla být vedena, avšak její struktura se může různě lišit. Nejčastější struktura ale vypadá tak, že jsou všichni členi posazeni u jednoho stolu a jeden po druhém odpovídají na následující otázky s ohledem na poslední Sprint:

- Co se mi líbilo a šlo fungovalo dobře?
- Co se mi nelíbilo a mohlo by být zlepšeno?
- K čemu se zavážeme v rámci příštího Sprintu pro zlepšení?

Cílem je, aby si tým sám postupně přicházel na nefungují a neefektivní procesy, zvyklosti a praktiky, a aby je následně řešil a zlepšoval. Z praxe je ale často tento meeting přeskakován a týmy se raději vrhají na následující Sprint Planning, což není dobře a měl by být kladen důraz na dodržování této schůzky (Kniberg, 2007) (Scrum).

Sprint Review

Účelem Sprint Review je předvést výstup ze sprintu zákazníkovi a dostat od něj zpětnou vazbu na předvedené produktový inkrement, který tým realizoval. Není třeba uživateli předvádět technické řešení či jednotlivé User Stories, ale ideálně by měl být představen produkt a jeho fungující funkcionality, aby mohl zákazník vidět reálný produkt a ohodnotit právě přidanou business hodnotu. Z počátku to může být náročné, ale je vhodné nechat prezentovat dodaný inkrement členy týmu, neboť právě vývojáři jsou ti, kteří vědí, jak daná funkcionalita přesně funguje a zároveň to zvyšuje motivaci jednotlivců. Během této schůzky je Product Owner v roli posluchače a sbírá veškerou zpětnou vazbu, kterou následně využije pro další sprinty (Šochová, a další, 2019).

3.4.3 Scrum artefakty

Dalšími komponenty Scrum frameworku jsou artefakty, které jsou navržené tak, aby maximalizovaly transparentnost klíčových informací a aby všichni těmto informacím rozuměli stejně.

Sprint

Scrum framework je postaven na iteracích, a tou nejzákladnější je Sprint. Jde o pravidelný cyklus časově omezený s jasně definovanými pravidly a postupy, které jsou v průběhu dané iterace dodržovány. Během jednoho Sprintu vývojový tým pracuje na implementaci User Stories, ke kterým se jako tým zavázali dostát. Doporučená délka Sprintu se liší tým od týmu a bude záležet na tom, v jakém časovém úseku je tým schopen dodávat takový inkrement, který je možné v rámci jednoho Sprintu realizovat a nasadit. Teorie v základu vychází z měsíčních cyklů, ale jak již bylo zmíněno, mělo by jít o délku, která bude pro tým optimální. Zbytečně prodlužování časového rozmezí pro Sprint není příliš doporučeno, neboť se pak tým okrádá o získávání časté zpětné vazby, která je u agilních metodik klíčová (Myslín, 2016).

Cílem Sprintu je tedy dosáhnout na začátku vydefinovaného Sprint Goalu, nikoli dokončení celého Product Backlogu. Vývojáři pracují na vybraných User Stories ze Sprint Backlogu, který by měl v ideálním případě zůstat prázdný. Během Sprintu by nemělo docházet k ad hoc meetingům potřebujících pro vydefinování či pochopení zadání, neboť toto by mělo být vyjasněno již před začátkem Sprintu a všichni by měli přesně vědět, na čem a proč pracují.

Sprint Goal

Jde o cíl či vizi Sprintu. Sprint Goal týmu říká, co je tou byznys hodnotou, kterou zákazník chce a potřebuje dodat v rámci konkrétního sprintu. Na definici cíle se podílí Product Owner společně s vývojovým týmem v rámci Sprint Planningu. Je doporučeno, aby se týmy zaměřovaly na položky, které přinášejí nejvyšší byznysovou hodnotu za nejnižší vydanou snahu. Definovaný Sprint Goal by nám měl odpovídat na otázku – Proč provádíme tento Sprint (Kniberg, 2007).

Product Backlog

Jde o uspořádaný produktový seznam položek, které říkají, co je třeba implementovat pro zlepšení produktu. Je to souhrn konsolidovaných požadavků, které kontinuálně prochází stakeholdeři a vývojový tým s Product Ownerem, který je hlavní osobou, která má Product Backlog nastarost. Product Owner je ten, kdo určuje priority jednotlivým User Stories. Seznam by měl být správně seřazen nejen podle priority, ale také podle toho, zda má konkrétní User Story jasně vydefinované zadání. Protože u agilně dodávaného produktu na začátku nevíme, co vše budeme dodávat a v jak to přesně bude vypadat, tak z tohoto

důvodu nemusí být všechny User Stories rozebrány do nejmenšího detailu, ale lze položky udržovat i na úrovni Epiců, což je úroveň nad User Stories. Epics nám sdružují více User Stories, díky kterým se následně požadována funkcionalita rozpadá do konkrétních požadavků a potřebných kroků (Šochová, a další, 2019).

Sprint Backlog

Jde o užší seznam již konkrétních User Stories, které byly vydefinovány týmem v rámci Sprint Planningu a ke kterým se tým zavázal dodat je do konce naplánovaného Sprintu. Jak již bylo zmíněno výše, User Stories do Sprint Backlogu se vybírají podle priorit a reálného odhadu práce tak, aby bylo možné dokončit Sprint Goal v rámci Sprintu a mohl být výsledný inkrement představen zákazníkovi a uživatelům.

User Story

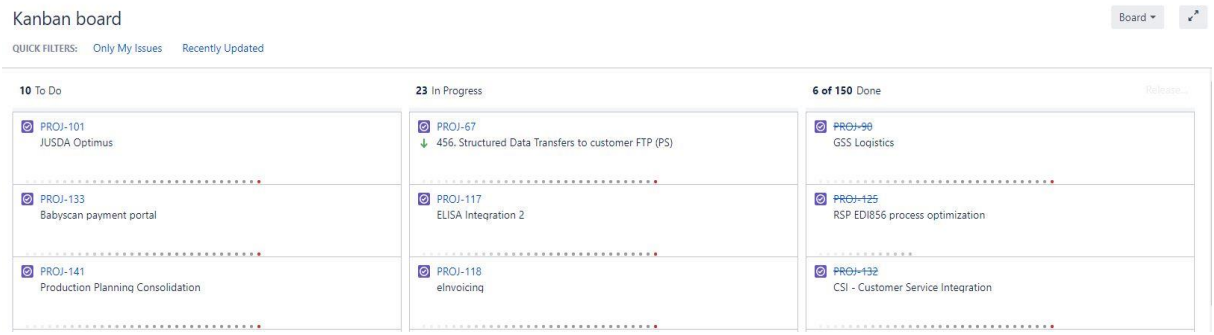
Pojem User Story vychází z metodiky Extreme Programming, kde se v rámci nich zachycují všechny požadavky na produkt. Popisují očekávanou a požadovanou funkcionalitu z uživatelské perspektivy. Formují se tímto stylem: „Jako Uživatel chci Funkcionalitu, abych dostal Byznys hodnotu“. Popis User Story by měl být jednoznačný, měl by vytvářet obrázek o požadavku. Dobře strukturovaná User Story se pozná, pokud ji vyhodnotíme, že odpovídá podle pomocného nástroje INVEST. To v originále znamená Independent, Negotiable, Valuable, Estimable, Small, Testable. Je to tedy posouzení, zda se nám do User Story vyplatí investovací čas, úsílí a peníze (Šochová, a další, 2019).

3.5 Kanban

Kanban je původem z Japonska, kde se nejvíce proslavil, když byl implementován v továrnách na řízení výroby. Kanban framework je oproti Scrumu méně zaměřený na strategické řízení vývoje softwarového produktu, ale hodí se do prostředí, kde je kladen důraz na rychlou reakci na změny a dodání produktu. Jeho hlavním nedostatkem je to, že týmům vlastně nic nenařizuje a mohou si o spoustě věcech rozhodovat sami. Jediné, čeho se týmy musí držet je dodržení třech principů (Šochová, a další, 2019):

- **Omezit rozpracovanou práci** – WIP (Work in Progress)
- **Vizualizace progresu**
- **Minimalizace času průchodu** – doby dodávky produktu

Z jedné studie vyplývá, že hlavní přínosy této metodiky jsou zkrácení doby dodání softwaru, zlepšení kvality softwaru, zlepšení komunikace a koordinace, zlepšení konzistence dodání produktu a v neposlední řadě snížení reportovaných defektů zákazníkem (Ahmad, a další, 2013).



Obrázek 4 - Kanban tabule

Zdroj: vlastní zpracování

Přehledné vizualizace požadavků a jejich stavu lze dosáhnout pomocí tzv. Kanban tabule, která zobrazuje jednotlivé úkoly a rozděluje je do sloupců podle jejich aktuálního stavu. Díky tomu všichni členové vývojového týmu vědí, jaký je aktuální stav úkolů a také na čem kdo pracuje (Šochová, a další, 2019). Na obr. 4 lze takovou tabuli vidět. Tabule má vydefinované pouze tři stavy – „To Do – In Progres – Done“. V každém z těchto sloupců lze vidět jednotlivé úkoly. Na příkladové tabuli z obr. 4 lze vidět, že je rozpracováno dvacet tři úkolů a dalších deset jich čeká až budou volné kapacity vývojového týmu a bude je možno implementovat. Důležité je zmínit, že takto nastavena Kanban tabule může být jakkoli upravena podle potřeb konkrétního týmu, neboť každému vývojovému týmu může vyhovovat jiný životní cyklus vývoje softwaru a podle toho si mohou nastavit Kanban tabuli (Šochová, a další, 2019).

Druhým zásadním nastavením v týmu využívající Kanban metodu, je správné nastavení limitu WIP, tedy kolik maximálně otevřených úkolů může tým mít v jeden okamžik, aby byl byla zajištěna co nejvyšší efektivita. V případě, že se povolí limit na vysoké číslo, dojde k tomu, že členové týmu budou přeskakovat z úkolu na úkol a v důsledku toho budou jednotlivé dodávky zpožděny (Šochová, a další, 2019). V opačném případě, kdy limit snížíme na nižší číslo, než je optimum, tak bude docházet k nízkému využití volných kapacit a neefektivní práci (Radigan).

Framework Kanban nedefinuje žádné role, které by se měli pro jeho využití aplikovat. Z praktických zkušeností je využití samotného Kanbanu pro vývoj softwaru nevhodné a doporučuje se ho kombinovat s některou z dalších agilních metodik (Šochová, a další, 2019).

3.6 ScrumBan

Jednou z dalších agilních metodik je ScrumBan, jejíž základy pocházejí podle názvu ze Scrumu a Kanbanu. Tato metodika byla Ladasem definovaná jako přechodná metodika pro týmy přecházející ze Scrumu na více vyvinuté frameworky (Ladas, 2008). ScrumBan framework umožňuje týmům využívat známe artefakty a ceremonie vycházející ze Scrumu, a zároveň užít metriky a řízení toku práce známé z Kanbanu (Reddy, 2015).

Rozdíly této metodiky oproti Scrumu jsou (Reddy, 2015) (Germanov, 2019):

- uznávání role managementu, přičemž cílem týmů zůstává sebeorganizace, ale v rámci vymezených pravidel
- uplatnění jasných zásad týkajících se způsobů práce
- uplatnění pravidel toků a teorie řazení do fronty

Oproti Kanbanu se ScrumBan liší v (Reddy, 2015):

- předepisuje základní rámec procesu vývoje softwaru (vychází ze Scrumu)
- organizace kolem týmu
- uznává přínos časově omezených iterací
- implementuje techniky neustálého zlepšování

Implementace této metodiky je doporučeno zavádět postupným objevováním stávajících procesů, do kterých budou následně postupně implementovány jednotlivé prvky ze Scrumu či Kanbanu (Reddy, 2015). Vychází to z principů Kanbanu, které nabádají k nepřetržitému zlepšování skrz kontroly a adaptaci.

3.6.1 Role a ceremonie

Teorie nedefinuje doporučený počet lidí v týmu ani žádné specifické role, tudíž je na konkrétním týmu, jaké si zvolí nastavení v jejich prostředí. Praktikované ceremonie ve ScrumBanu jsou obdobné jako ty ze Scrumu, pouze u ScrumBanu je možné jejich průběh různé modifikovat k potřebám daného týmu. Pro ilustraci lze třeba využít Daily Standup Meeting, který má ve Scrumu jasnou agendu – jednotlivci během patnácti minut vysdílejí

s týmem, co dělali včera za úkoly, co budou dělat dnes a zda mají nějaké problémy s pokračováním v dané práce. V případě ScrumBanu lze v rámci této schůzky zajít dále než ke sdílení stavu a vytváření závazků k další aktivitě (Reddy, 2015).

3.6.2 Vizualizace a pull systém

Metodika využívá jeden z hlavních pilířů, na kterém stojí Kanban, a tím je dobrá vizualizace práce za využití Kanban tabule. Je vhodné, aby i u této metodiky byly všechny požadavky dobře viditelné a tím byla zajištěna transparentnost v týmu i mimo něj (Ponomareff, 2017). Skladba takové tabule, která bude vizualizaci navrhovat se moc neliší od toho, jak je představena teorií vycházející z metodiky Kanban. Jedním z rozdílů je, že ScrumBan počítá s dalšími typy požadavků, které je doporučeno v rámci tabule rozlišit barvami, aby bylo jasné, o jaký typ požadavku jde (Reddy, 2015).

Dalším krokem je nastavení limitu maximálního počtu rozdělaných úkolů a zajištění fungujícího pull systému. Nastavení limitu je snazší na exekuci, neboť jde o hledání optimálního čísla, kolik otevřených úkolů může jedinec mít najednou, aby byla stále zabezpečena nejvyšší efektivita. Najít takovou optimální hodnotu bude vyžadovat několik retrospektiv, pozorování a nastavování změn. Ovšem vytvořit fungující pull systém znamená motivovat všechny členy takovým způsobem, aby v případě, kdy mají volněji a zbýval například jeden úkol v backlogu, tak aby si ho dotyčný přiřadil k sobě a začal na něm samostatně pracovat. Jde o práci s lidmi a týmem jako celkem, v případě velké fluktuace v týmu může docházet k nekonzistencím, a je nutné nastavit správné zásady a pravidla pro členy týmu, které musí být průběžně kontrolovány a zlepšovány (Ponomareff, 2017).

Využití ScrumBanu podle Reddyho jsou následující (Reddy, 2015):

- Pomáhat týmům urychlit přechod na Scrum od dalších vývojových metodik.
- Poukázat na nové schopnosti, které mohou týmům a organizacím pomoct překonat výzvy, kterým čelí během aplikování Scrumu.
- Pomáhat organizacím vyvíjet nové procesy a postupy podobné Scrumu, které fungují.

Praktická část

4 Analýza současného stavu

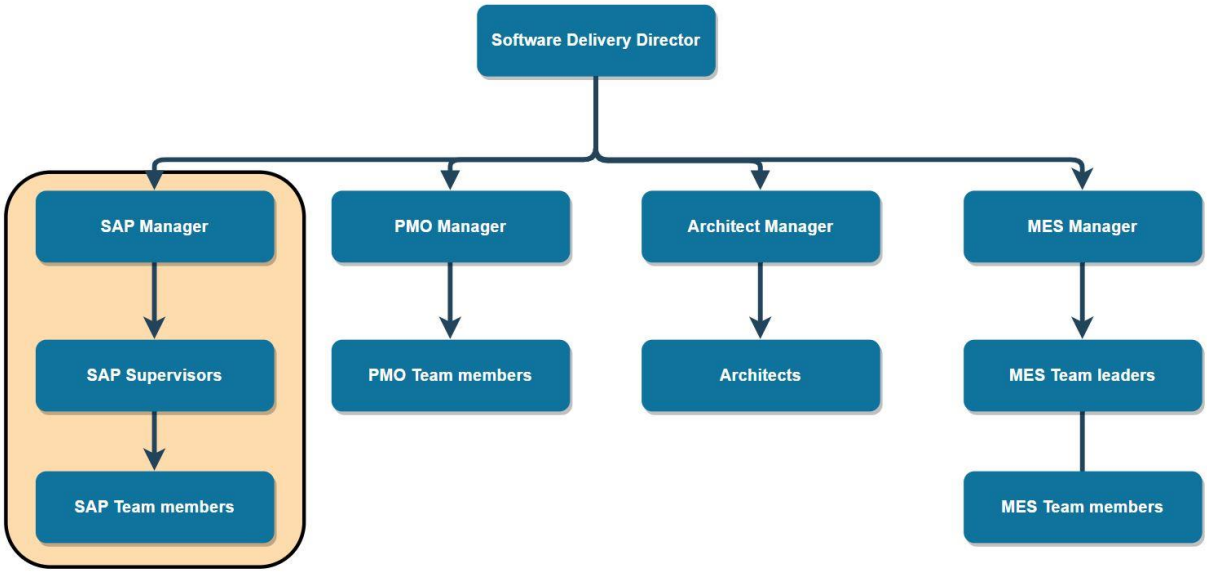
Následující část práce představí oddělení zodpovědné za vývoj a údržbu softwaru ve výrobní společnosti, organizační strukturu a interní procesy jednoho z vývojových týmů, nástroje a praktiky, které tým využívá v rámci životního cyklu vývoje softwaru. Kapitola bude zakončena výčtem problémů, které byly identifikovány na základě výstupů získaných metodou osobního rozhovoru a pozorování. Provedeno bylo několik rozhovorů s členy SAP týmu a aby byl zajištěn objektivní náhled na věc, rozhovory byly uskutečněny i s projektovými manažery, kteří úzce spolupracují se SAP týmem. Další metodou získávání dat a informací pro lepší pochopení situace v týmu bylo pozorování. Autor práce se osobně zúčastnil několika týmových schůzek, kde byl v roli pozorujícího.

4.1 Představení podniku

Analýza vývojového týmu probíhala na oddělení vývoje softwaru v mezinárodní výrobní společnosti, která má sídlo v Pardubickém kraji. Ačkoli společnost aktivně podporuje spolupráci se studenty, vyhrazuje si zachování anonymity a mlčenlivosti o zásadních informacích, které by mohli poškodit nejen společnost samotnou, ale i její zákazníky či dodavatele. Z tohoto důvodu nebude společnost jmenována a nebudou zveřejněné citlivé informace v největším detailu.

Vybraná společnost je světovým leaderem v produkci spotřební elektroniky a patří mezi nejvýznamnější exportéry v České republice. Na území České republiky má více výrobních závodů, avšak softwarové oddělení, které vyvíjí a udržuje výrobní systémy, pracuje i pro závody mimo Česko. V současné době je na vybraném oddělení přibližně kolem 100 zaměstnanců, kteří jsou rozděleni do týmů podle toho, jaký produkt vyvíjejí. Před čtyřmi lety došlo k obměně ve vedení oddělení, které přišlo s vidinou změnit způsob myšlení a přístup k vývoji softwaru, tzv. agilní transformací. Během těchto uplynulých čtyř let bylo možné sledovat, že každý z týmů zkouší různé metodiky a postupy, aby dosáhly nalezení vhodného a optimálního nastavení pro své potřeby. Z výsledků pozorování lze vyvodit, že každý z týmů je na úplně jiné úrovni, co se týká zralosti a sebeorganizace.

V rámci této práce byl vybrán tým zodpovědný za systém SAP, který podporuje většinu firemních procesů. SAP tým je zodpovědný nejen za vývoj softwaru na míru dle požadavků byznysu, ale také za údržbu systému a podporu uživatelů na denní bázi.



Obrázek 5 - Organizační struktura softwarového oddělení

Zdroj: vlastní zpracování

4.2 Zákazníci

Zákazníkem vyvíjeného SAP systému na míru jsou zaměstnanci z různých oddělení, které se starají o procesy výrobní společnosti. SAP ERP systém je schopen podporovat různé podnikové procesy, a proto může být zákazníkem uživatel z jakékoli oblasti, kde implementace SAP dává smysl a má přínos. Mezi nejčastější zákazníky v této společnosti patří týmy z těchto odvětví:

- logistika
- finance a controlling
- podpora prodeje
- plánování výroby
- řízení dodavatelského řetězce

Míra složitosti zadání se různé liší požadavek od požadavku. Uživatelé mohou přicházet, jak s potřebou změnit sloupec ve výstupním reportu, tak s požadavkem na vytvoření programu například pro komplexní plánování materiálů na výrobní linky.

Podle komplexnosti, urgentnosti, náročnosti a potřeby zapojení dalších týmů kvůli potřebné integraci s jiným systémem, může zákazník vytvořit žádost v podobě incidentu, ticketu nebo projektu.

4.3 Vývojový tým

Role v rámci SAP týmu jsou rozděleny následovně:

- **Manažer** – manažer týmu má v rámci své role na starost vedení lidí, nábor nových lidí, řízení interních procesů a strategie jeho týmu.
- **Architekt** – jde o seniorní roli, která určuje pravidla a mantinely vývoje systému tak, aby byl udržován domluvený standard a nezvyšoval se technický dluh.
- **Týmový Supervizor** – plní roli leadera v rámci dalšího dělení týmu, neboť celý SAP tým je rozdělen po jednotlivých oblastech (viz. výše). Mezi jeho odpovědnosti patří rozvoj a podpora ostatních členů týmu, určování vize v dané systémové oblasti, rozhodování.
- **Konzultant** – role, která přebírá požadavek od zákazníka, jenž následně spolu s architektem ale i zákazníkem konzultují a dávají dohromady návrh pro nejvhodnější řešení. Konzultantem by měla být osoba, která velmi dobře zná a chápe podnikové procesy z uživatelského pohledu a zároveň má stejně dobré technické znalosti, aby byl schopen vytvářet technické zadání pro programátora.
- **Programátor** – na základě definovaného technického zadání od konzultanta píše kód v jazyce ABAP, což je programovací jazyk určený pro vývoj podnikových aplikaci v SAP Netweaver.

Jak již bylo zmíněno, SAP tým dodává produktové změny pro několik různých podnikových oblastí, a protože každá z oblastí je velmi specifická na znalosti, je tým rozčleněn do stejného počtu mikro týmů jako je počet oblastí, tj. čtyři týmy a jejich značení je následující:

- **MM & WM** – Materials Management & Warehouse Management
- **FI & CO & AM** – Financial Accounting & Controlling & Asset Management
- **SD** – Sales and Distribution
- **PP** – Production Planning
- **ABAP –** Programmers

Tyto menší týmy se skládají z dvou až čtyř konzultantů, kteří dále spolupracují s programátory, kteří již nejsou nijak děleni podle specifických znalostí, a proto je lze využívat obecně pro všechny typy požadavků.

4.4 Fáze procesu vývoje

V této kapitole bude popsán životní cyklus vývoje softwaru od vzniku požadavků po nasazení na produkční prostředí a následnou údržbu.

4.4.1 Inicializace požadavku

Vzhledem k tomu, že jde o tým zaměřený nejen na vývoj nových funkcionalit, ale i na běžnou podporu výrobních procesů, kde může dojít například k výskytu incidentu v kritických procesech, které mohou zastavit výrobu, je nutné popsat všechny vstupy skrz které se inicializuje požadavek na vývojový tým.

Incident

Prvním zmíněným typem požadavku je incident, jenž je definován jako problém, který má dopad na výrobu či jiné kritické byznysové operace. Tyto typy požadavků se dále dělí podle závažnosti / priority. Přístup k jejich řešení je deklarován v Incident Management proceduře, kterou manažer SAP týmu vytvořil. V proceduře je mimo definici závažnosti popsán způsob podpory, tedy v jakých případech bude jaký tým incident řešit. Dále dokument stanovuje podmínky SLA (angl. Service-level agreement), což představuje smlouvu mezi zákazníkem a poskytovatelem služby, který se tím zavazuje k jejímu dodržení. Na to navazují kapitoly o kontaktech, způsobu eskalace či grafické zobrazení cyklu řešení incidentu. Požadavek na řešení incidentu může vytvořit kterýkoli uživatel SAP systému s přístupem k Jira ServiceDesku a zároveň incidenty mají přednost před dalšími typy požadavků, tudíž jsou realizovány přednostně a bez procesu plánování.

Neprojektové požadavky

Dalším způsobem vytvoření požadavku je založení neprojektové ticketu, který může být založen každým zaměstnancem společnosti, který má přístup do Jira systému. Než se úkol dostane do procesu plánování, je nutné, aby byl ticket schválen přímým nadřízeným žadatele. V případě, že je ticket schválen, dochází k přiřazování priority. Tuto část stále zajišťují lidé z byznysu, kteří byli nominováni do pozice podobné definici Product Ownera vycházející z metodiky Scrumu. V tomto případě tedy dochází k tomu, že někde existují zástupci,

kteří určují priority za své divize, čímž dochází k tomu, že existují například tři neprojektové tickety s prioritou jedna a s tím následně musí pracovat SAP tým v rámci plánování. V tomto případě není výstupem žádný dokument obsahující zadání požadavku, pouze vytvořený Jira ticket, kde je popsané, co zákazník požaduje.

Projekt

Každá z výrobních divizí si vede své projektové portfolio, skrz které se řídí dílčí projekty směrem k softwarovému oddělení. Na základě potřeb a vyhodnocení návratnosti investice se projekty začínají. Tato část spadá vyloženě pod zákazníka ačkoli zkoumaný SAP tým projekt začíná zajímat již v této fázi, kdy byznysové týmy vyžadují úvodní konzultace ohledně možných řešení, aby bylo možné spočítat návratnost investic alespoň z hrubých odhadů. V případě, že se zákazník rozhodne projekt realizovat, přechází se k založení projektu v nástroji Jira a následnému určení priority projektu. V této fázi nebudou ještě všechny požadavky stoprocentně zřejmé, a proto projekt nebude ještě rozdroben do jednotlivých User Stories. V rámci využívání Jira nástroje, SAP tým nevyužívá možnosti tvorby Epiců, které by mohly agregovat jednotlivé User Stories do větších celků. Výstupným dokumentem z této fáze je BRD (angl. Business Requirements Document), který bude obsahovat rámcový seznam požadavků bez detailního rozboru.

Projekty jsou prioritizovány v rámci jednotlivých divizí, což znamená, že softwarové oddělení musí pracovat s několika řadami důležitosti a plánovat kapacity svých týmů závisle na určeném pořadí projektů.

4.4.2 Definice zadání

V případě incidentů je zadání většinou jasné. V produkčním systému došlo k chybě, zákazník popíše proces, který nefunguje. V lepším případě je zákazník schopen poskytnout bližší informace, jako třeba příklad, na kterém se systém nezachoval podle očekávání. S tím následně konzultanti pracují a hledají příčinu problému. Většinou probíhá přímá komunikace mezi konzultantem a reportérem incidentu skrz Jira systém, aplikaci Skype či MS Outlook.

Pokud dojde k zaplánování neprojektového ticketu tak nejprve dochází k detailnějšímu popisu zadání požadavku. Tato fáze probíhá přímo mezi žadatelem a přiřazeným konzultantem, který si konkrétní ticket nechal přiřadit při plánovacím meetingu. Konzultant se snaží získat bližší informace od autora zadání, aby následně mohl poskytnout výstup z analýzy, tedy návrh řešení. V rámci návrhu konzultant poskytuje i odhad pracnosti, který

musí být schválen nadřízeným žadatele. V případě, že dojde ke schválení je nutné zajistit prioritu ticketu a dostatek potřebných kapacit konzultanta a ABAP programátora na další plánovací schůzce.

Definice zadání u projektů probíhá odlišně než u předchozího případu. U projektů je výchozím bodem dokument BRD, s kterým následně projektový tým v rámci analýzy pracuje. K této fázi dochází v případě, že má ticket dostatečnou prioritu, aby bylo možné nechat zaplánovat potřebné kapacity vývojového týmu. Následně probíhají schůzky projektových týmů, kde se definuje zadání za spolupráce zákazníka s členy vývojových týmů. Výstupem je detailní BRD dokument, který jasně formuluje požadavky, které se v rámci projektu mají realizovat. Tyto požadavky již dostávají formu User Stories a jsou vytvořeny v Jira systému. Na základě takto připraveného BRD dokumentu je pak možné vytvářet rozpočet projektu a plán realizace, který je nutné nechat schválit stakeholdery projektu.

4.4.3 Vývoj aplikace

Incidenty jsou řešeny přednostně, tudíž ze není prostor pro čekání na přiřazení priorit. Týmy se těmto ticketům věnují, pokud možno okamžitě, tak aby byl zajištěn chod produkce. V praxi to tedy znamená, že jsou veškeré další aktivity upozaděny těmto případům.

Pokud jsou neprojektové úkoly správně zaplánovány, tak se konzultant v rámci této fáze věnuje detailní přípravě technického zadání, které následně předává ABAP programátorovi. Komunikace mezi konzultantem a programátorem probíhá skrz systém GitLab, kde konzultant vytváří kopii Jira ticketu a kam zadává technickou specifikaci. V GitLabu je připravená Kanban tabule, kde si ABAP programátoři berou jednotlivé zadání ke zpracování. Aby nedocházelo k úplným nekonzistencím mezi zaplánovaným časem konzultanta a programátora na stejný časový úsek, jsou jednotlivé úkoly plánovány na konkrétní týdny i programátorům, nejen konzultantům. Realitou ale bývá, že k těmto situacím i tak dochází, což znamená zpoždění dodání produktu k fázi testování.

V případě vývoje projektů se projektové týmy řídí podle jednotlivých plánu projektů a plánovací proces je odlišný od procesu předchozího. V tomto případě Projektový manažer plánuje kapacitu členům projektového týmu v plánovacím nástroji mimo Jira systém, kde dochází k synchronizaci Projektových manažerů každý týden ve čtvrtek, tedy den před plánovací schůzkou SAP týmu. Dochází tedy ke dvojímu plánování. Prvním krokem Projektových manažerů je zajistit to, že členové týmu budou mít řádně zaplánované projektové aktivity takovým způsobem, že nebude požadovaný čas přesahovat třiceti

hodinovou kapacitu, kterou konzultanti mají vyhrazenou na projektové aktivity. Druhým krokem je zajištění toho, že jsou zaplánované aktivity v souladu s projektovým portfoliem a plánem projektu. V případě, že jsou tyto kroky hotové, dochází k plánování SAP týmu, který na základě těchto vstupů plánuje konkrétní plán jednotlivců na následující týden.

Každý z konzultantů tedy pracuje každý týden s různou skladbou požadavků, které by mu v ideálním případě měly pokrýt celotýdenní kapacitu a měl by být zaměřen pouze na tyto zaplánované činnosti.

Během samostatné přípravy technického zadání a následného programování může dojít k nejasnostem, které jsou řešeny v přímé diskuzi mezi zákazníkem – konzultantem – programátorem. V případě projektu jsou tyto nejasnosti řešeny na projektových schůzkách, které si řídí Projektový manažeři. Pokud jde o požadavky měnící architekturu řešení, dochází k zapojení architektů do fáze realizace, aby došlo k potvrzení správného návrhu řešení.

Před předáním produktu zákazníkovi do fáze testování je konzultant s programátorem povinen provést interní testování na vývojovém prostředí.

4.4.4 Testování a implementace

SAP tým nemá vyhrazenou roli testera, tudíž je tato část procesu zastoupena konzultantem společně se zákazníkem. Konzultant po interním testování programátora předává realizovaný požadavek na testovacím prostředí zákazníkovi, který testuje podle jím připravených testovacích scénářů. V případě, že jde o incident nebo neprojektový ticket, záleží na dohodě mezi konzultantem a zákazníkem, zda otestovaný a funkční inkrement bude implementován na produkční prostředí. Připravené User Stories, které mohou být importovány na produkční prostředí, musí zákazník odsouhlasit skrz Jira systém a následně jsou každý všední den kromě pátku prováděny transporty na ostrý systém.

U projektů se implementace odehrává jinak a odvíjí se od velikosti projektu. V principu ale jde o to, že se nenasazují jednotlivé User Stories postupně, ale jako ucelený balík, neboť ve většině případů zákazník potřebuje mít nasazený celý celek, ne pouze jednu část funkcionalit. Samozřejmě realizují se i projekty, kde situace nevyžaduje nasazování celého balíčku, ale lze postupovat postupným nasazováním menších celků. Ve fázi přechodu na živý systém u projektu dochází ke koordinaci všech vývojů a jsou importovány společně podle potřeb zákazníka. Rozhodnutí vydává Projektový manažer zodpovědný za projekt.

4.4.5 Dokumentace

Během vývoje si konzultanti a programátoři zároveň tvoří technickou dokumentaci dle svého uvážení, neboť do této doby nebyla tvorba dokumentace v podstatě nijak řízená, což vedlo k tomu, že tým nemá centralizované místo, kde by se dokumenty ukládaly a kam by se mohli konzultanti, a především nový členové týmu podívat v případě potřeby. Nyní dochází k zavádění procesů, které mají tvorbu dokumentace zajistit a také je určena jednotná struktura těchto dokumentů. Ve všech případech se má dokumentace tvořit během fáze realizace a má být společně předána zákazníkovi po úspěšném testování a nasazení na produkční prostředí.

4.4.6 Údržba

Po implementaci funkcionality na základě neprojektového požadavku předává SAP konzultant Jira ticket na zákazníka do stavu Done, kdy je nyní na zákazníkovi, aby prověřil na produkčním prostředí, zda funkcionality plní, co požadoval. Pokud je vše v pořádku, zákazník uzavírá Jira ticket a už se k této problematice nevrací. Pokud ovšem nalezne žadatel nějakou chybu, tak jí reportuje skrz Jira ticket a dává do stavu Bugs, který si zase přebírá konzultant a připravuje opravu společně s programátorem. Následně probíhá stejný proces testování a implementace na produkci jako bylo popsáno v předchozí části. Pokud dojde k nalezení chyb po implementaci projektových požadavků, jsou tyto nedostatky reportovány skrz zákazníka na Projektového manažera, který koordinuje jejich odstranění. Dále záleží na situaci a stavu projektu. Pokud dojde k nalezení chyby až po čase, kdy už byl projekt uzavřen, realizuje se oprava skrz neprojektový ticket ale s vyšší prioritou, aby došlo k opravě a nasazení opravené verze co nejdříve. Jestliže je projekt stále otevřený a zjistíme chybu po transportu nové produkční verze, tak je projektový tým ve fázi podpory a přednostně reaguje na tyto reportované problémy.

4.5 Procesy a využití praktik pro řízení vývoje softwaru

4.5.1 Porady

Týdenní plánování

SAP tým má společně s manažerem týmu nastavené týdenní schůzky, které se zúčastňují všichni členové týmu, aby mohli prodiskutovat a zaplánovat prioritní neprojektové a projektové User Stories na následující týden. Tato schůzka probíhá každé páteční dopoledne a trvá dvě a půl hodiny. Do plánování zasahují dvě roviny – první je plánování projektových User Stories na základě pořadí určeného projektovými týmy zákazníků, a druhá, kdy žadatelé mimo projektové týmy určují pořadí priorit pro neprojektové User Stories a jsou nezávislé na projekty. To vede k tomu, že SAP tým nemá jednu osobu, která by zastřešovala priority v rámci jejich Produktového backlogu, ale jde o kombinaci prioritních řad, s kterými musí tým v rámci plánovací meetingu pracovat. Během schůzky jsou manažerem týmu nejprve procházené požadavky na prioritní projektové User Stories, pro které je každému členovi z týmu vyhrazen část v rozmezí ¾ jejich pracovního času. Ve zbývajícím čase jsou následně rozplánovány důležité neprojektové úkoly. Členům týmu se plánuji úkoly na celých 40 hodin za týden a nerezervuje se čas pro případnou potřebu řešení incidentu či jiných urgentních problémů, které se ale vyskytují pravidelně.

Týdenní standup meeting

V rámci zmíněných mikro SAP týmů, tedy týmů rozdělených podle oblastí, které zastřešují, mají nastavené týdenní standup schůzky, kde členové probírají své tickety a řeší problémy, kterým čelí. Tato iniciativa je viditelná pouze u jednoho oblastního týmu, a to SAP MM & WM. Ostatní týmy tyto schůzky nemají.

Projektové schůzky

V rámci projektu si každý z Projektových manažerů určuje svůj způsob vedení lidí a i způsob, jakým spolu členové týmu budou komunikovat. Často se ale praktikuje to, že jsou naplánované týdenní schůzky v rozmezí od patnácti do třiceti minut, kde jednotlivci reportují stav jednotlivých User Stories a sdílejí své problémy, pokud nějaké jsou. Díky tomu je zaručena synchronizace v rámci jednoho projektu, už se zde ale neodehrává žádná schůzka, která by pomohla synchronizaci na úrovni všech projektů. Proto členové SAP týmu v podstatě přeskakují z projektu na projekt a je pouze na nich, jaký si udržují přehled o všech aktivitách, které je čekají.

Měsíční retrospektiva

Jako zbytek oddělení i SAP Týmový Supervizoři jsou zváni na měsíční retrospektivu, která je iniciována ředitelem oddělení. Tato schůzka probíhá každý poslední pátek v měsíci a jde o dvou hodinovou schůzku, která probíhá podle teorie Retrospektivy vycházející ze Scrumu. Všichni zúčastnění členové dostávají prostor na to, aby sdíleli s ostatními, co se podle nich za poslední měsíc podařilo, co se nepodařilo a v čem by měli i nadále pokračovat. Po sběru těchto vstupů je druhá část schůzky věnována diskuzi nad jednotlivými body, kde je cílem si říci, co mohou být další kroky pro zlepšení situace.

4.5.2 Nástroje

Jedním z aspektů řízení vývoje softwaru je také využívání vhodných nástrojů, které podporují týmy v dosažení vyšší efektivity a z toho důvodu budou představeny i nástroje, které SAP tým využívá.

- **Jira –** komplexní softwarový nástroj určený pro řízení projektů a problémů při vývoji softwaru. Jde o produkt, který je zaměřený a připravený na podporu vývojových týmů, které se drží agilních metodik jako je Scrum či Kanban. Nabízí flexibilní a uživatelské funkcionality využitelné pro řízení požadavků, sledování progresu aktivit pracovníků, vizualizaci úkolů či například plánování.
- **Jira Servicedesk –** nadstavbová platforma k systému Jira, která je vhodná pro větší společnosti, neboť umožňuje to, že uživatelé, nepotřebují vlastnit Jira licence, které jsou placené. Díky tomu může větší množství zaměstnanců vytvářet požadavky skrz Jira ServiceDesk a zároveň není nutné vydávat zbytečné náklady za Jira licence.
- **GitLab –** verzovací software, který SAP tým využívá pro předávání technických zadání ABAP programátorům, kteří mohou díky GitLabu vyhledávat a přebírat části kódu, které už někdo vytvořil a uložil do systému.
- **Word –** známý produkt z balíčku Microsoft Office, jenž funguje jako textový editor, který je v rámci zkoumaného týmu využíván pro tvorbu dokumentace.
- **Confluence –** další z produktů společnosti Atlassian, který umožňuje vzdálenou spolupráci členů týmů, udržovat znalosti na jednom místě a spoustu dalších možností. Systém je dobře integrovatelný s Jira systémem, tudíž je možné vytvářet různé přehledy na základě dat z Jiry v Confluence.

- **OneNote –** produkt společnosti Microsoft, určený k vytváření a ukládání poznámek, jenž lze sdílet s dalšími uživateli. Z tohoto důvodu někteří členové využívají tento nástroj pro dokumentaci a její šíření v rámci týmu.
- **Sharepoint** – jde o platformu určenou pro správu dokumentů. Někteří členové týmu stále využívají pro své potřeby, ale jde spíše o přežitek, neboť pro tyto účely byla stanovena strategie využívání systému Confluence. Bohužel, ne všichni členové jsou schopni přistoupit na požadované změny, a proto stále využívají pro ukládání své dokumentace tuto možnost.
- **Microsoft Outlook –** hojně využívaný nástroj pro komunikaci se zákazníkem a napříč týmem samotným. Někteří členové týmu dokonce využívají Outlook jako jediný nástroj, kde diskutují procesní a technické zadání, díky čemuž se vytrácí transparentnost o stavu řešeného úkolu a možnost automatického ukládání technické dokumentace.
- **Skype –** nástroj pro komunikaci členů týmu mezi sebou a se zákazníkem.

4.6 Analýza problémů v týmu

Výzkumná část této práce spočívá v identifikaci zásadních problémů, které vycházejí z odpovědí zaměstnanců softwarového oddělení, s kterými byl veden strukturovaný rozhovor. Všechny odpovědi byli sepsány a následně autorem klasifikovány do kategorií, které jsou níže popsané v tabulce č. 1. Rozhovor byl veden s celkovým počtem čtrnácti respondentů, z čehož deset tázaných bylo ze zkoumaného SAP týmu. Zbývající čtyři respondenti byli z týmu PM

Problém	Název problému	Popis problému
P1	**Komunikace a spolupráce mezi týmy**	• Různé přístupy k práci, každý z členů vyžaduje jiný způsob spolupráce, komunikace. • Nedostatečná koordinace mezi lidmi pracujícími na ticketu, prostoje způsobují zapomenutí informací o řešeném ticketu. • Komunikace, transparentnost informací. Vše se řeší až když nastane problém, ale proaktivní sdílení informací chybí. • Nestandardizovaný způsob komunikace mezi všemi členy zainteresovanými ve vývoji. • Nedostatečné soft-skills vývojářů. Edukace může vést ke zlepšení. • Nedostatečně specifikovaná zadání od zákazníků, potřeba lepší vedení komunikace/specifikace. • Nesdílení znalostí mezi členy týmu, jednotlivci si drží své znalosti, špatné zastoupení v případě potřeby.

P2	**Plánování práce**	• Kapacitní nedostatky vs. množství požadavků = nedostatečná kvalita dodaného produktu, volba rychlého řešení před kvalitním. • Plánování vs. realita = odvedená práce často neodpovídá zaplánované práci (dělá se něco jiného, co v plánu nebylo). • Porušování procesu prioritizace, lidé nerespektují zaplánované úkoly a žádají od konzultantů práci i mimo zaplánované. • Dochází k nedodržení zaplánované práce. • Vždy dochází k nedodržení prvně odhadnutých termínů a k přeplánování.
P3	**Nástroje**	• Nevyužívání plného potencionálu nástrojů, které mají týmy k dispozici, a naopak využívání velkého množství nástrojů pro stejné účely. • Nestandardizované nástroje např. pro znalostní databázi, dokumentaci. • Nedostatečné tréninky se základními systémy, které týmy používají. • Pomalé nástroje (špatně nastavená infrastruktura systému, brzdí v práci a nabádá k nevyužívání systému).
P4	**Neochota přistoupit na změny / kultura organizace**	• Zajetý cyklus práce, neochota se učit novým věcem či něco měnit. • Převažují lidí, kteří jsou v této společnosti dlouho a nemají chtíč akceptovat změny. • Nový zaměstnanci přinesou myšlenky na změny, ale díky kultuře týmu zapadnou do zajetých kolejí.
P5	**Neznalost agilních metodik**	• Nikdo na oddělení týmy / jednotlivce needukuje o agilních metodikách a ani top po nich nevyžaduje. • Většina respondentů nezná agilní metodiky, neví, o čem jsou a jak se dají aplikovat. • Neznalost rolí vycházejících z agilních metodik a jejich význam (např. Scrum Master).
P6	**Motivace**	• Nedostatečná motivace, radost z práce, nedostatečné ocenění.
P7	**Chybí proces pro neustálé zlepšování**	• Neprovádí se pravidelné retrospektivy, z kterých by týmy mohly čerpat zpětnou vazbu pro následné zlepšení. • Není vyžadováno z manažerských pozic.
P8	**Pozdní dodání produktu**	• Nejsou nastaveny správně automatické procesy, např. pokud je vytvořen nový tiket, některé týmy nejsou schopny automaticky zadání přebrat a řešení čeká na eskalaci zákazníka. • Neefektivní nastavení průtoku práce

Tabulka 1 - Identifikované problémy na základě rozhovorů

Zdroj: vlastní zpracování

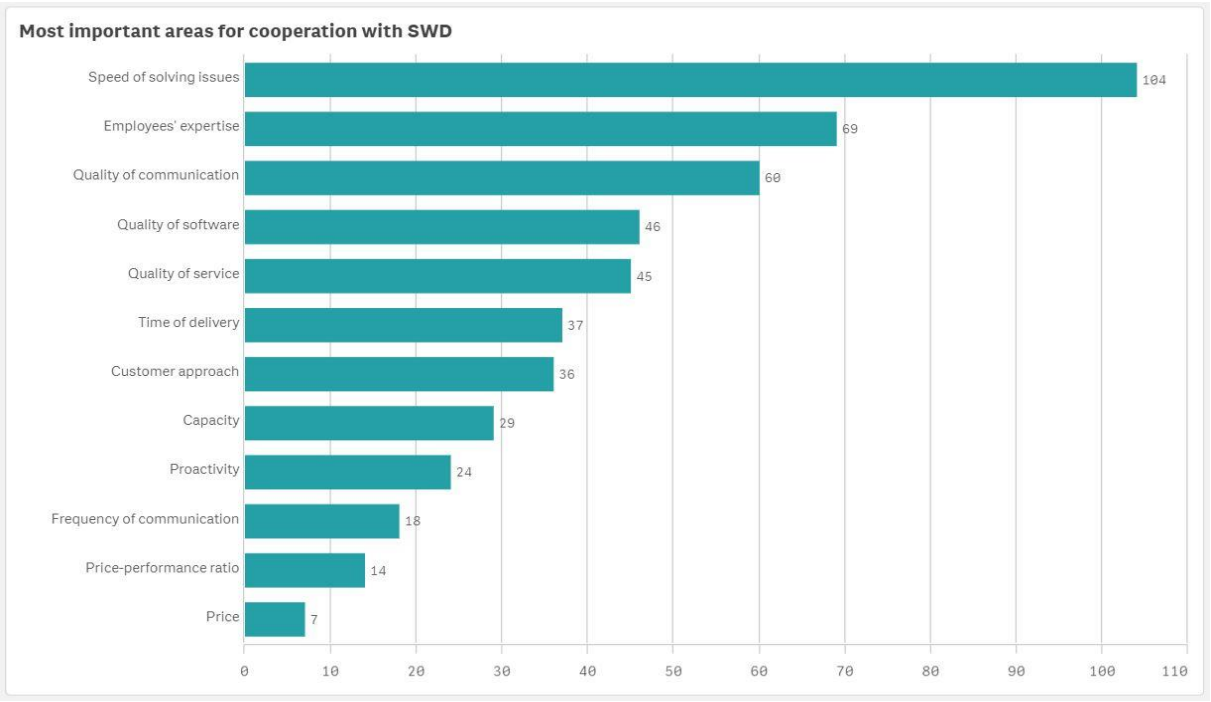
5 Sběr dat k vytvoření návrhu

Předchozí část práce poskytla seznam problémů, s kterými se SAP tým potýká a pro které chce autor práce navrhnout vhodné řešení za pomoci implementace agilních metodik či alespoň jejich částí. Pro zajištění vytvoření vhodného a použitelného návrhu v kontextu výše popsané společnosti, budou v následující kapitole představeny další dva soubory dat, které popisují potřeby zákazníka zkoumaného SAP týmu a aktuální trendy v agilním světě.

5.1 Zákaznický dotazník

V rámci získávání relevantní zpětné vazby realizuje softwarové oddělení každoroční dotazník zaměřený na spokojenost zákazníka s ohledem na poskytované služby. Dotazníkové šetření se skládá z široké sady otázek, avšak pro účely této práce budou vybrané pouze ty, mající přidanou hodnotu pro následnou tvorbu návrhu opatření vůči identifikovaným problémům v přechozí kapitole.

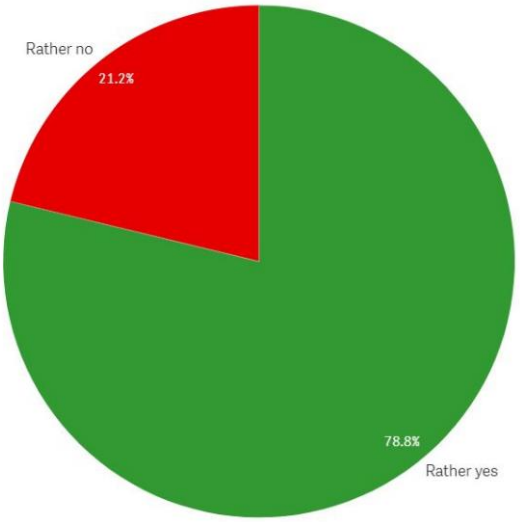
Dotazník byl rozeslán mezi všechny zaměstnance, kteří přišli do kontaktu se SAP týmem v rámci realizace jejich požadavku. Konečným součtem vyplněných dotazníků je 313, z toho 59 respondentů působí na vedoucí pozici. Každý z respondentů mohl vyplnit maximálně tři a minimálně jednu odpověď.



Graf 1 - Seznam nejdůležitějších oblastí pro kooperaci s SAP týmem

Zdroj: vlastní zpracování

Z grafu č. 1 je patrné, že pro zákazníky patří mezi tři nejdůležitější aspekty spolupráce rychlost řešení problémů, odborné znalosti zaměstnanců a kvalita komunikace. Na dalších příčkách se nachází oblasti jako kvalita dodaného softwaru, kvalita služeb, doba dodání či prozákaznický přístup. Jako nejméně podstatný aspekt vyšla cena za vývoj softwaru.

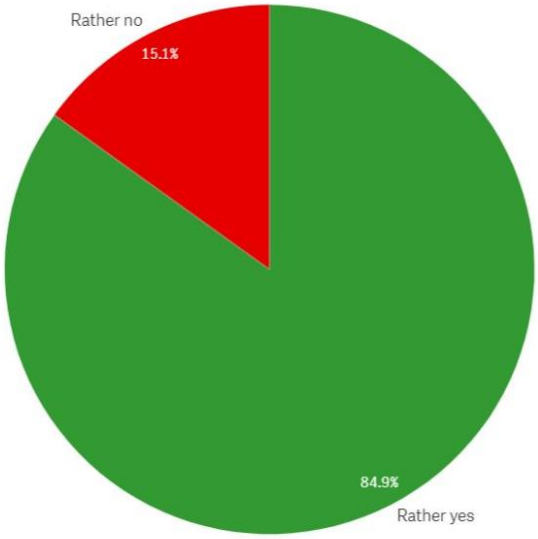


Graf 2 - Spokojenost s rychlostí řešení problémů

Zdroj: vlastní zpracování

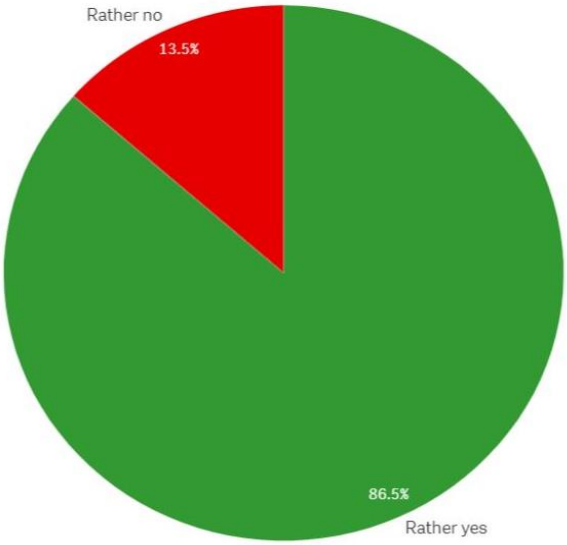
Graf č. 2 zobrazuje spokojenost zákazníků s rychlostí řešení problémů, které jsou reportované na SAP tým. Jelikož jde o aspekt, který je nejvíce vnímaný zákazníkem, lze konstatovat, že je zde spousta prostoru pro zlepšení tohoto bodu, neboť přes 20 % odpovědí bylo klasifikováno negativně.

Následující graf č. 3 prezentuje výsledek spokojenosti s expertízou zaměstnanců SAP týmu. Z výsledků je patrné, že zde převyšuje spokojenost o 6,1 % oproti předchozí otázce. Graf reprezentuje skutečnost, že skladba SAP týmu je vyvážená a je zde stále prostor pro zvýšení znalostí.



Graf 3 - Spokojenost s odbornými znalostmi zaměstnanců

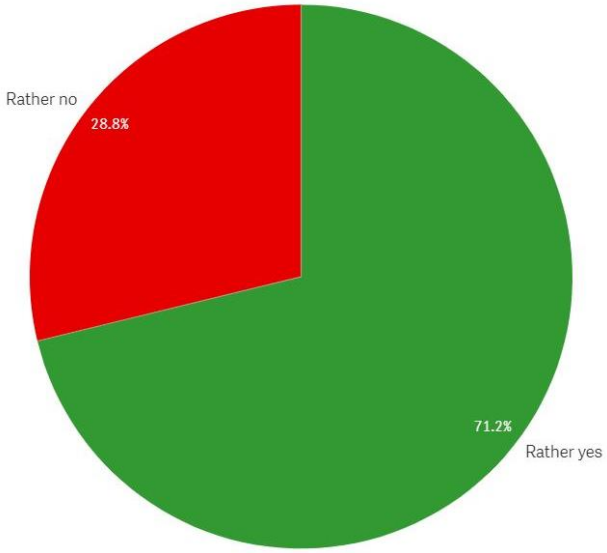
Zdroj: vlastní zpracování



Graf 4 - Spokojenost s kvalitou komunikace

Zdroj: vlastní zpracování

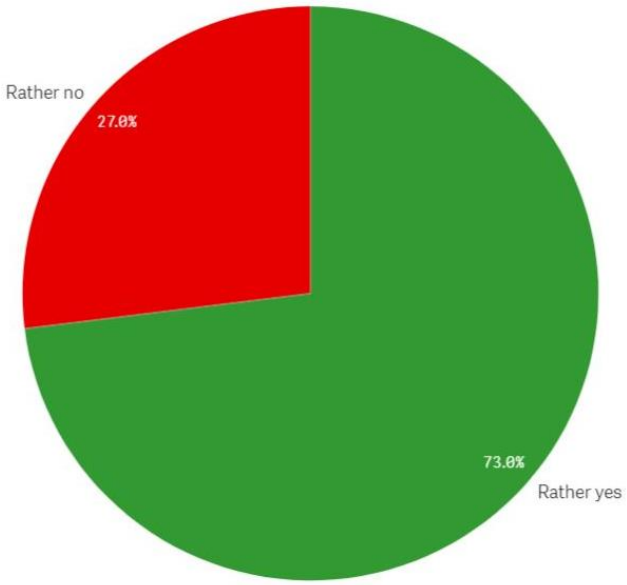
Třetí nejdůležitější oblastí pro zákazníka byla kvalita komunikace a podle grafu č. 4 vychází spokojenost s tímto aspektem nejslibněji z již zobrazených grafů v této kapitole. Pozitivní odezvu v hodnotě 86,5 % lze přisuzovat tomu, že se oddělení za poslední dva roky zaměřilo právě na zlepšení toho aspektu a nyní to sklízí své ovoce.



Graf 5 - Spokojenost s dobou dodání produktu

Zdroj: vlastní zpracování

Jako další graf je uvedena spokojenost s dobou dodání produktu, a to z toho důvodu, že v oblasti vývoje softwaru jde o velice důležité kritérium a zároveň výsledná spokojenost vyšla jako jedna z nejhorších. Necelých 30 % respondentů není spokojeno s dobou realizace jejich požadavků, což může být způsobeno různými příčinami. Některé z nich bude určitě možné najít mezi již zmíněnými problémy SAP týmu.



Graf 6 - Spokojenost s kvalitou dodaného softwaru

Zdroj: vlastní zpracování

Posledním grafem č. 6 je spokojenost s kvalitou dodaného softwaru. V porovnání s předchozími grafy je zřetelné, že i zde je dostatek prostoru pro zdokonalení. Právě toto kritérium bývá častým důvodem, proč se společnosti snaží implementovat agilní metodiky do svých procesů.

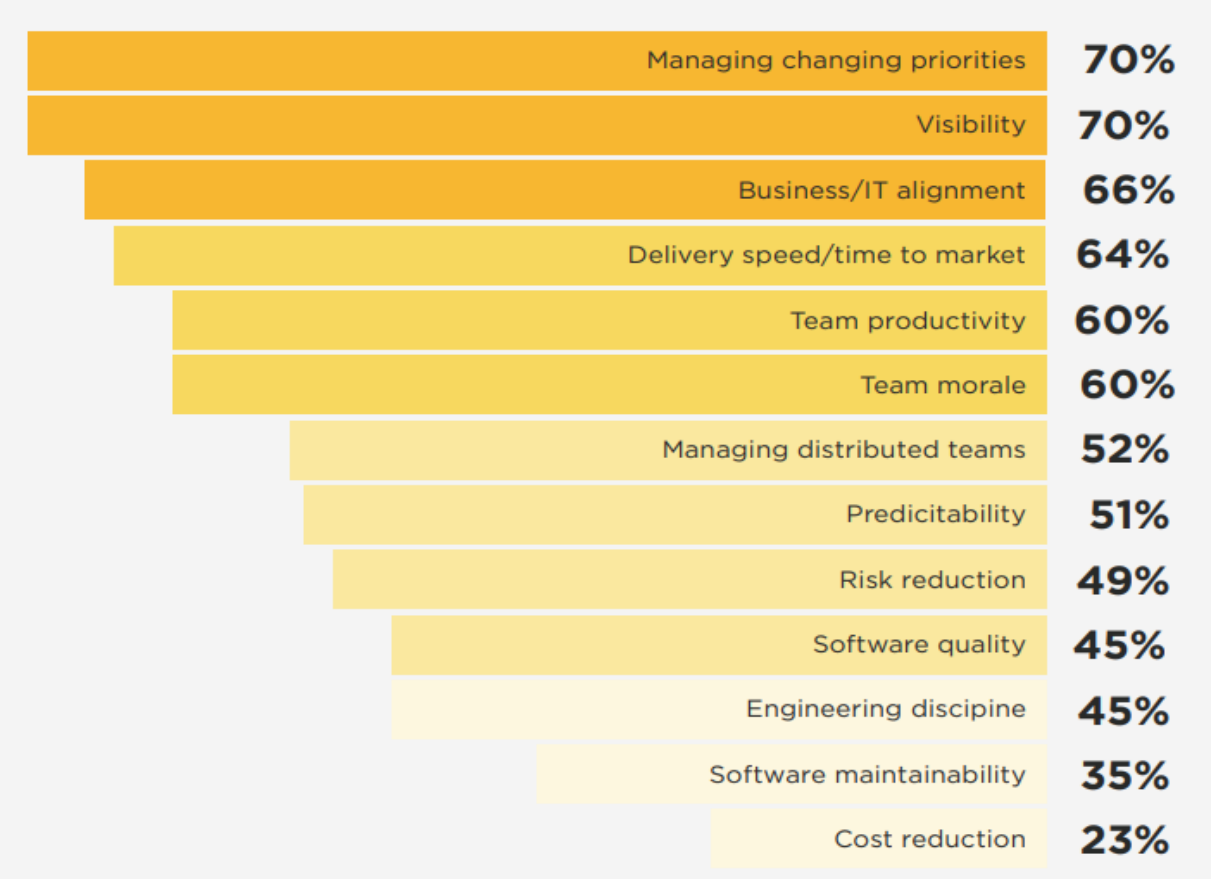
Dílčí závěr výstupu zákaznického dotazníku

Dle výstupu ze zákaznického dotazníku realizovaného v průběhu ledna 2021 víme, s kterými aspekty spolupráce může SAP tým být spokojen a také na které by se měl zaměřit a uzpůsobit takové změny, aby došlo ke zlepšení budoucí kooperace a spokojenosti zákazníka.

5.2 State of Agile Report

Dalším podkladem pro následující kapitolu zaměřenou na návrh vhodných agilních metodik a praktik s ohledem na identifikované problémy, je další dotazníkové šetření, které již od roku 2006 provádí americká společnost digital.ai. Od té doby každým rokem vydávají nové výsledky z jejich každoročního šetření, které reflektují nejnovější trendy ve využívání agilních metodik a nástrojů po celém světě. Dotazník nese název „State of Agile Report“ a poznatky využité v této kapitole jsou konkrétně z patnáctého vydání, které vyplnilo 1 382 tázaných (Digital.ai, 2021).

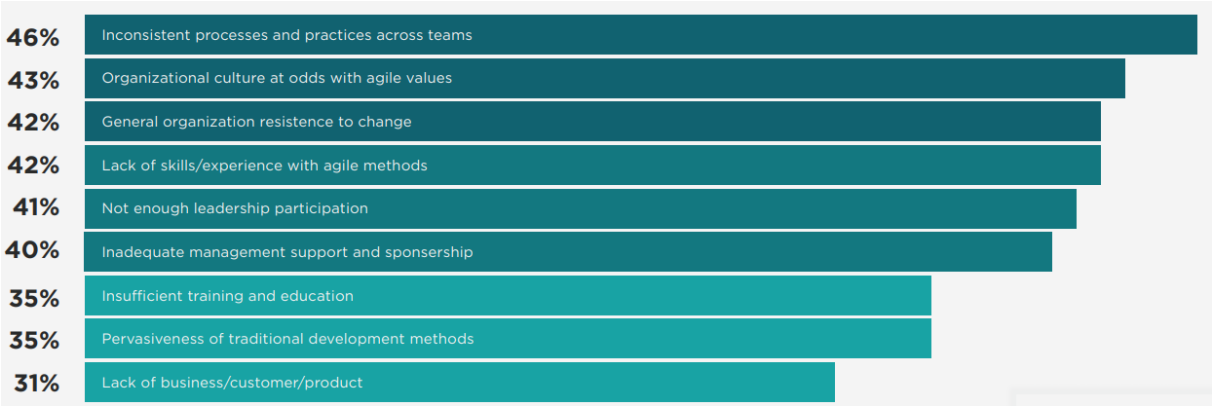
Cílem této kapitoly je představit si aktuální trendy v agilním prostředí a přínos využití agilních metodik, na které bude následně brán zřetel při výběru vhodných agilních metodik a praktik v rámci následující části práce zaměřené na návrh pro SAP tým ve zkoumané společnosti.



Graf 7 - Dopad implementace agilních metodik v organizacích

Zdroj: (Digital.ai, 2021)

Na prvním uvedeném grafu v této podkapitole lze vidět, jaký má implementace agilních metodik vliv na jednotlivé oblasti v rámci organizace podle odpovědí tázaných. Mezi první tři příčky se řadí zlepšení řízení změn priorit, transparentnost a sladění byznysu s IT. Dalšími pozitivně ovlivněnými jsou doba dodání produktu, týmová produktivita a morálka.

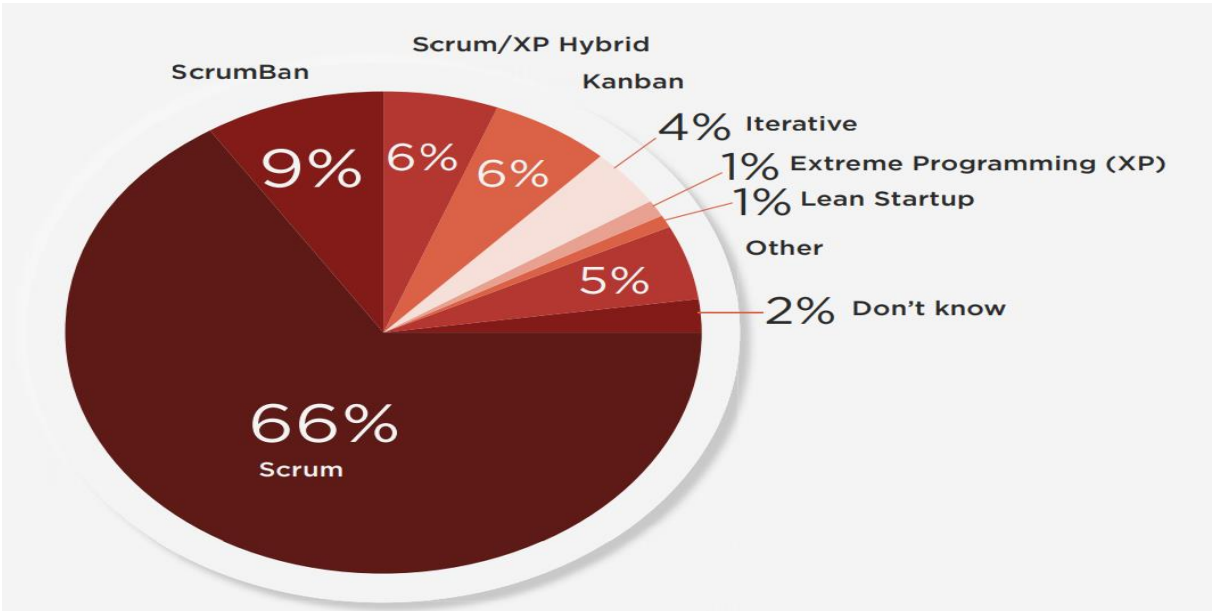


Graf 8 - Zásadní bariéry bránící adaptaci na agilní metodiky

Zdroj: (Digital.ai, 2021)

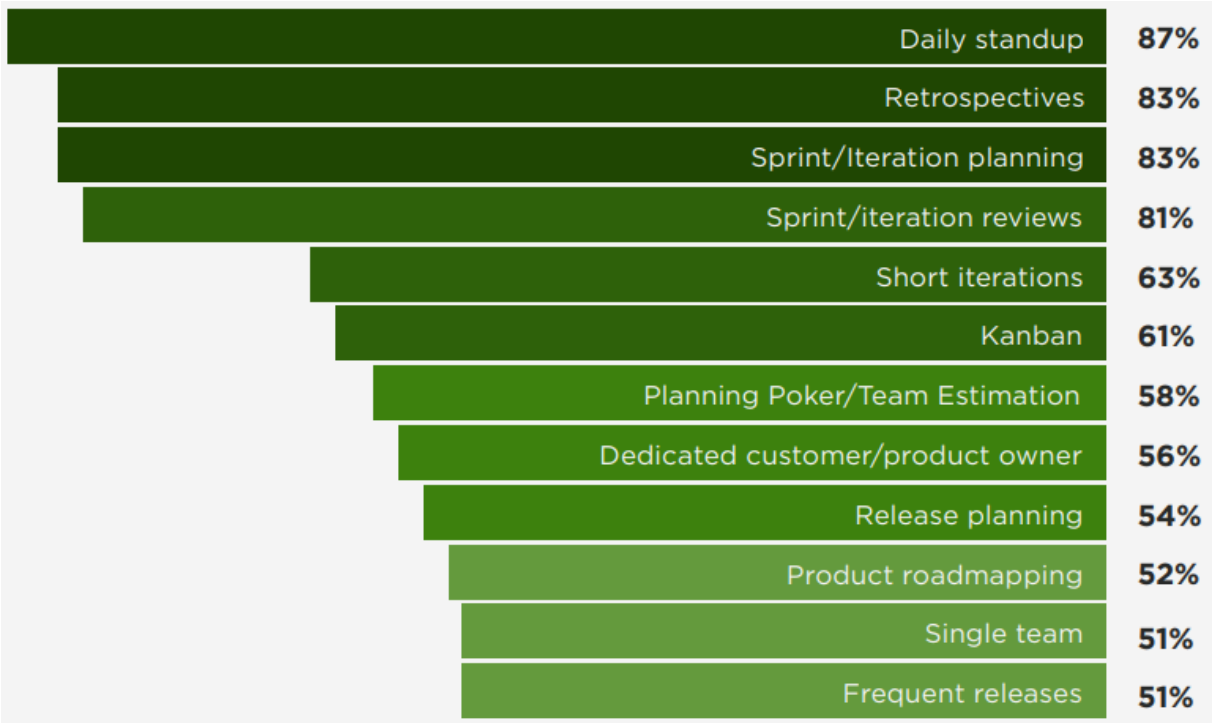
Graf č. 8 představuje nejzásadnější výzvy, kterým organizace, snažící se prosadit agilní přístup, čelí. Mezi hlavní bariéry patří nekonzistence procesů a praktik skrz týmy, organizační kultura, odpor ke změnám či například nedostatek znalostí agilních metodik. Z tohoto grafu si můžeme všimnout jistého souznění se zmíněnými problémy v rámci SAP týmu z kapitoly 4.6.

Následující graf č. 9 poukazuje na nejpoužívanější agilní metodiky na úrovní týmů. Na první příčku se řadí původní verze Scrum frameworku, přičemž některé další pozice byly obsazené modifikacemi Scrumu jako třeba ScrumBan nebo Scrum / XP Hybrid. Podobné množství hlasů jako obdrželi zmíněné modifikace, dostal i framework Kanban.



Graf 9 - Nejpoužívanější agilní metodiky na úrovni týmů

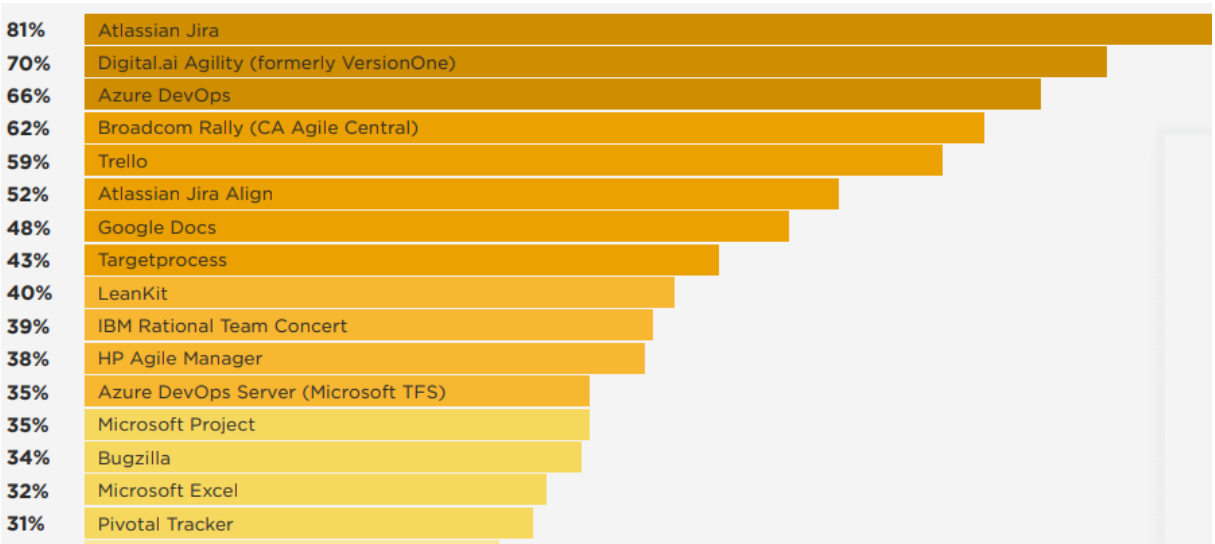
Zdroj: (Digital.ai, 2021)



Graf 10 - Nejpoužívanější agilní techniky a praktiky

Zdroj: (Digital.ai, 2021)

Mezi nejpoužívanější agilní události se řadí Daily standup, Retrospektiva a Sprint Planning, které vycházejí ze Scrumu. Mezi další hojně praktikované techniky se řadí dříve zmíněný Kanban, který je vhodný pro vizualizaci a omezení rozdělané práce.



Graf 11 - Nástroje pro agilní metodiky

Zdroj: (Digital.ai, 2021)

Na grafu č. 11 je zobrazen seznam nástrojů, které by respondenti dotazníku doporučili ostatním společnostem pro podporu agilních metodik v případě vývoje softwaru. Nejvyšší doporučení s 81 % obdržel systém Jira od společnosti Atlassian, který již lze najít na seznamu používaných nástrojů zkoumaného SAP týmu.

Dílčí závěr State of Agile Reportu

Díky aktivitě společnosti Digital.ai lze využít výstupy z celosvětového šetření, které mohou být přínosné pro jakýkoliv tým, plánující implementaci agilních metodik do životního cyklu vývoje softwaru. V tomto případě budou výsledky zobrazené v této kapitole využity pro sestavení návrhu, který bude stát na srovnání aktuální situace v týmu s teoretickými východisky, která budou podpořena právě praktickými doporučeními vycházející z těchto dat.

6 Návrh

Cílem této kapitoly je navrhnout opatření a procesy, které povedou ke zlepšení situace s ohledem na identifikované problémy v rámci SAP týmu. Vzhledem ke komplexnosti popisovaného prostředí řešeného podniku, budou navrhnuty změny, které vycházejí z agilních metodik a praktik, a zároveň se budou držet výstupů z dvou dříve zmíněných dotazníků, které budou složit jako výchozí body, stanovující očekávání a možnosti.

6.1 Kritéria

Ze zákaznického reportu představeného v kapitole 5.1 je zřejmé, která kritéria jsou pro zákazníky SAP týmu, a tedy i softwarové oddělení důležité a kritické. Zároveň výsledná data poukazují i na ta kritéria, která jsou vhodnými kandidáty pro zlepšení. Mezi tato kritéria patří:

- **Rychlost řešení problémů** – neboli minimalizace času od vytvoření ticketu po jeho realizaci, předání k testování a nasazení na produkci.
- **Odborné znalosti členů týmu –** zajistit procesy vedoucí k zvyšování znalostí a odbornosti členů v týmu.
- **Komunikace –** častá interakce mezi zainteresovanými osobami zvyšuje efektivitu a transparentnost, díky čemu dochází k rychlejšímu řešení problémů.
- **Doba dodání produktu –** zajištění dodávání produktů ve smluvených termínech.
- **Kvalita dodaného softwaru –** eliminovat nadbytečnou chybovost v průběhu realizace požadavků a zajistit tak vyšší spokojenost s dodaným softwarem.

6.2 Uvažované agilní metodiky

Potřebná kritéria již byla určena a nyní je třeba si definovat jaké agilní přístupy budou zvažovány pro sestavení návrhu, který bude brát na zřetel zmíněná kritéria a zároveň identifikované problémy. Pro tuto část budou využity výsledky aktuálních trendů agilního světa vycházející ze State of Agile Reportu, který je popsaný v kapitole 5.2. Do výběru jsou zaražené tyto tři metodiky, z kterých bude dále čerpáno:

- **Scrum**
- **Kanban**
- **ScrumBan**

6.3 Organizační struktura

V reakci hned na několik problémů (P1, P5, P8) viz. tabulka č. 1 je vhodné začít se změnou v samotné organizační struktuře týmu. Z vybraných metodik pro sestavení návrhu ušitého na míru konkrétnímu týmu, pouze framework Scrum definuje role, které by bylo vhodné zakomponovat do procesů SAP týmu.

6.3.1 Role Scrum Mastera

Zásadní změnou pro eliminaci problémů (P1, P5, P8) by mělo být vytvoření pozice pro Scrum Mastera, který by měl na starost SAP tým. Podle Scrumu jde o nezbytnou a pro správné fungování vyžadující roli, která nesmí chybět. Na druhou stranu během rozhovorů byla respondentům položena i dílčí otázka, co si myslí a ví o této roli. Většina odpovědí se nesla v negativní duchu, neboť jednotlivci buď nemají žádnou zkušenost anebo mají, ale špatnou. Proto v případě implementace této role do týmu by bylo vhodné najít kandidáta, který je nadšený do agilního přístupu, má velkou dávku empatie a rád pracuje s lidmi. Především poslední bod bude zásadní, neboť zde narážíme na problémy, které se spolu v jistých aspektech pojí. Protože jedním z hlavních příčin, proč tým nevyužívá žádnou z agilních metodik, je jejich neznalost, dochází tím také k tomu, že členové týmu mají mylné představy o tom, jak by měl Scrum Master fungovat a jaké přínosy by vlastně mohl mít. Z tohoto důvodu si tedy vyvozují to, že tuto funkci tým nepotřebuje.

Z identifikovaných problémů je ale zřejmé, že právě tato role by mohla přinést poznání, co to agilní metodiky jsou a jak se dají realizovat v praxi. Vhodné by bylo, kdyby působil i jako tzv. agilní kouč a měl přesah i přes další agilní praktiky. Důležitým faktorem je, aby osoba zastupující tuto roli byla schopna si získat přirozený respekt od jednotlivých členů týmu, což může být značná výzva v prostředí, kde jsou vztahy a procesy určeny členy, kteří působí v týmu dlouhé roky. S ohledem na tuto skutečnost je doporučeno, aby šlo o osobu, která nepůsobí v oddělení na některé jiné pozici, ale aby šlo o nového kandidáta. Jeho prvním úkolem bude jasné definování a vysvětlení, co jeho pozice a role obnáší a jak si představuje spolupráci s jednotlivými členy a týmem jako celkem. Komunikace by měla být co nejupřímnější tak, aby mezi Scrum Masterem a vývojovým týmem vznikla synergie a důvěra. Jeho dlouhodobým cílem je vytvoření samoorganizovaného týmu.

6.3.2 Role Product Ownera

V tomto případě by bylo velmi náročně využít role Product Ownera přesně podle předepsaných pravidel, neboť požadavky přicházejí z několika divizí a pro jednu osobu by bylo velmi náročné vědět, co má větší nebo menší prioritu a čemu by se měl vývojový tým následně věnovat. Z tohoto důvodu je doporučeno, aby existovala tato role na úrovní každé divize, což znamená, že by požadované úpravy za každou divizi byly řízené skrz jejich Product Ownera, který by udržoval jasné pořadí priorit za jejich byznys. Takto udržovaný Produktový Backlog by vypadal následovně – viz. obr. č. 6. Díky tomu by byl jasně stanovený seznam prioritních User Stories, které by si vývojový tým následně mohl plánovat do svých iterací.

TP	Priority	Summary	Status
↑	1	FII reporting	TESTING (UAT)
↑	1	L5->L10 BOM setting and planning for ISR products	WAITING FOR CUSTOM...
↓	1	KH3 - zoutdel úprava	TESTING (UAT)
↑	1	ZMTRANS2, ZMARGIN: rozhodné datum pro výpočet marže G	IN PROGRESS
↓	1	FATP - druhy beh (hard wo)	NEW
↓	2	Změna sortovací/segmentační logiky - přídání N9K produktů n	IMPLEMENTATION ESTI...
↓	2	CN Inventory snapshots	ANALYSIS
↓	2	KH3 PMI - zwmsboxidchng kontrola na cílovou SJ	TESTING (UAT)
↑	2	Úprava USD účtů ČSOB v sapu	TESTING (UAT)
↑	2	Victoria/Forex project - pre-analysis, workshops,	ANALYSIS
↑	3	Validace účtování rozvaha x podrozvaha	ANALYSIS ESTIMATE A...
↑	3	KH3 - analýza na ošetření situace PMI (výpadek v processingu)	IN PROGRESS
↑	3	Ekokom - ZEK_FINAL discrepancy	ANALYSIS

Obrázek 6 - Návrh Produktového Backlogu

Zdroj: vlastní zpracování

6.3.3 Vývojový tým

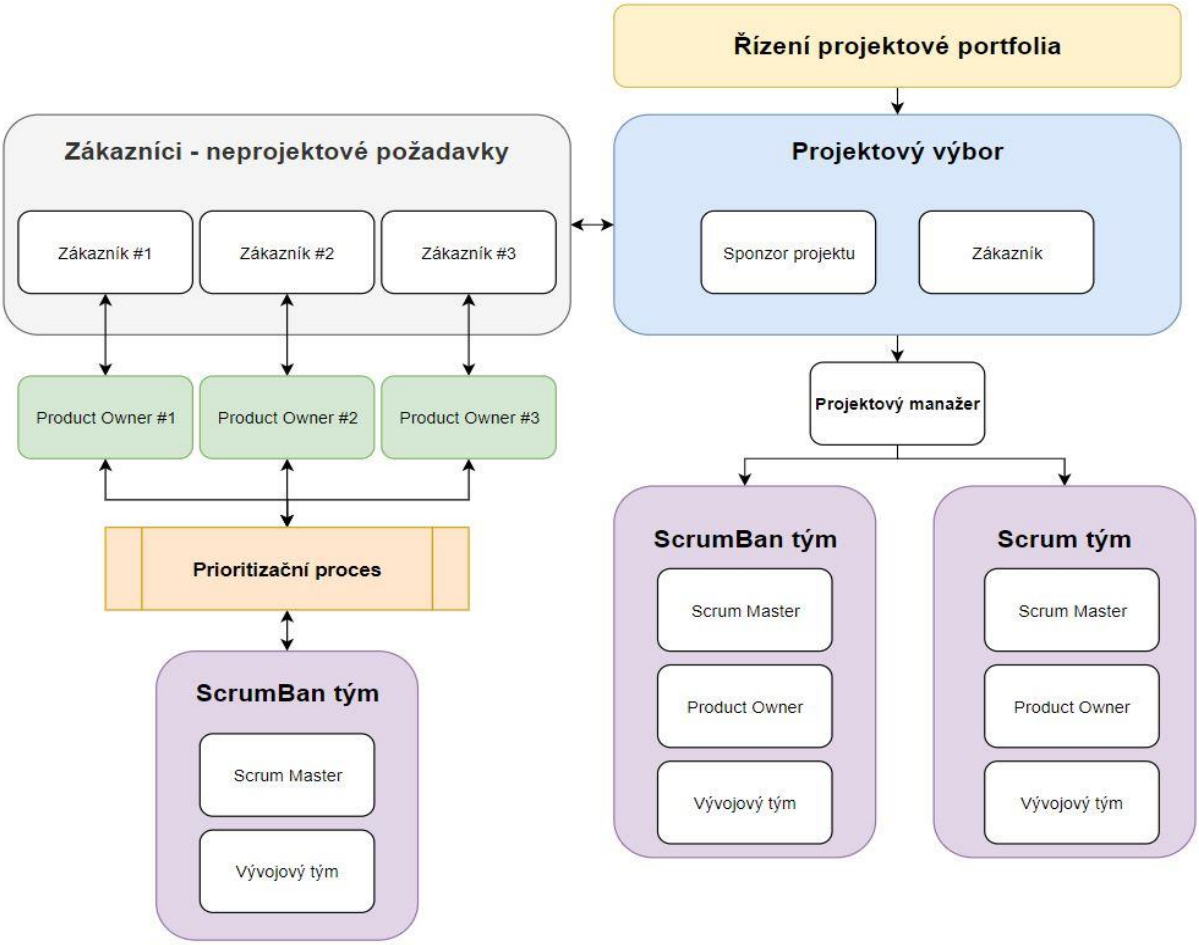
V původním nastavení týmu figurují pouze dvě role – konzultant a programátor. Konzultantovi odpovědnosti začínají od převzetí požadavku, přes provedení analýzy a návrhu řešení, po předání zadání programátorovi a následné testování se zákazníkem. Z podstaty přístupu k vývoji tohoto softwaru není nutné, dělit role v rámci týmu na analytiky či testery. Doporučením pro tento organizační celek je ale to, aby se skrz rovnoměrnou dělbu práce

zajistila větší zastupitelnost a rozšířila znalost různých procesů skrz všechny členy týmu. Tím je myšleno to, aby se jednotlivci nezavazovali jen k realizaci požadavků jednoho zákazníka, ale aby se jejich pole působnosti cíleně střídalo. Tento přístup může mít pozitivní dopad na dobu dodání produktu, neboť je možné zaručit zastupitelnost v případě výpadku některého z členů týmu, popřípadě pro využití dalších kapacit na aktuální implementaci, čímž by došlo také ke zrychlení dodávky, protože by nebylo nutné zajistit dlouhé vysvětlování požadavku a návazných procesů, kterých se změna týká.

Změnou myšlení směrem k agilnímu přístupu, otevřenosti a transparentnosti, dojde ke zlepšení problémové oblasti týkající se komunikace a spolupráce v rámci týmu i mimo něj. Je doporučeno zaměřit se na tým a jednotlivce, ujistit se, že rozumí agilnímu přístupu a že jsou společně s vedením oddělení na stejné lodi. Z teoretického východiska Scrumu je doporučeno, aby tým seděl společně v jedné místnosti, ale vzhledem k aktuální situaci způsobené koronavirovou pandemií a zavedenými restrikcemi v jejím důsledku, není aktuálně toto doporučení realistické. Proto budou následovat další doporučení pro udržení kontaktu mezi členy týmu v následující kapitole zaměřené na schůzky a porady.

6.3.4 Role projektového manažera

Protože značná část požadavků zákazníka je realizována skrz projektové řízení, tak je nezbytné mít i nadále roli projektového manažera v rámci organizační struktury. To je v souladu s teorií Scrumu, neboť i podle tohoto frameworku se tato role nemá plně rušit. Co se ale s tímto přístupem mění jsou kompetence a odpovědnost projektového manažera. Ten se i nadále stará o přípravu projektového plánu a rozpočtu, následné sledování pokroku a reportování. Kde už ale jeho pravomoc končí je zasahování do řízení práce vývojového týmu.



Obrázek 7 - Návrh organizační struktury

Zdroj: vlastní zpracování

Pro vizuální představu byl vytvořen model organizační struktury SAP týmu v návaznosti na zákazníka, aby bylo jasné, jaký postup se následuje u každého z typu požadavků.

6.4 Návrh implementace agilních praktik a technik do vývojového procesu

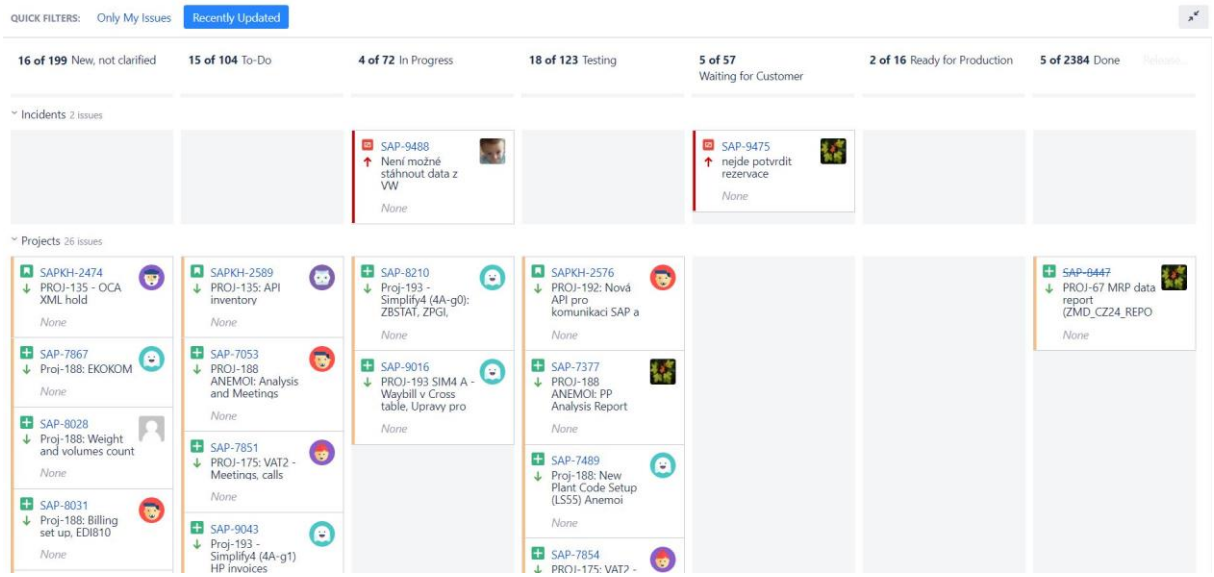
V rámci tvorby návrhu vývojového procesu byl brán zřetel na teoretická východiska z agilních metodik – Scrum, Kanban a ScrumBan. S ohledem na komplexnost prostředí není doporučeno využít pouze obsah jedné metodiky, ale využít dílčí komponenty, které by mohly dohromady tvořit fungující celek a přinést tížené benefity.

6.4.1 Návrh inicializace požadavku

Z důvodu rozdílnosti typů zadání, které se k SAP týmu dostávají od zákazníků, je doporučeno, aby se sjednotil prioritní seznam požadavků – tj. Produktový Backlog. V praxi by to znamenalo řízení priorit nejen u neprojektových User Stories ale také u těch projektových. Díky tomu by byla zajištěna vyšší transparentnost toho, na čem kdo zrovna pracuje, kolik práce zbývá k dokončení a v jakém stavu se User Story aktuálně nachází. Odstranilo by se tím nepřesné plánování na úrovni projektů, kde se aktuálně plánují kolikrát nadbytečně vyšší časy, které nakonec nejsou pro zaplánovaný projekt stejně vykázány.

6.4.2 Kanban tabule

Vizualizace by byla zajištěna Kanban tabulí, který byl v rámci této práce vytvořen takovým způsobem, aby byl použitelný pro všechny členy SAP týmu. Nyní je totiž situace taková, že neexistuje jednotná Kanban tabule, ale hned několik a díky tomu je místo transparentnosti zajištěn chaos.



Obrázek 8 - Návrh Kanban tabule

Zdroj: vlastní zpracování

Jednotlivé User Stories jsou v Kanban tabuli roztříděné podle jejich statusů a názvy sloupců představují fázi, ve které se zrovna nacházejí. Fáze jsou rozděleny následovně:

1. **New, not clarified –** agreguje všechny User Stories, které čekají v Produktovém Backlogu než jim bude přiřazena dostatečná priorita aby mohly být zaplánovány do „To-Do“ fáze. Počet User Stories v tomto sloupci není nijak limitován.
2. **To-Do –** seznam zaplánovaných User Stories, které byly díky své prioritě předání do stavu To-Do a nyní čekají až si je členové týmu rozeberou k realizaci. Z relativně nové metodiky ScrumBan vychází doporučení, že lze určit i maximální počet zaplanovaných User Stories. To v tomto případě znamená, že by ze začátku bylo doporučeno plánovat maximálně pět ticketů na jednotlivce. Pro zajištění vysoké efektivity bude nutné realizovat retrospektivu, aby byla najita optimální hodnota této limitace.
3. **In Progress –** tickety v této fázi značí to, že se konzultant či programátor zavázal k jejich dodání. Patří sem User Stories, které jsou ve stavu analýzy, návrhu či implementace. Rozlišení, zda je ticket u konzultanta nebo programátora by šlo řešit přiřazením k jednotlivým uživatelům. Připravená tabule zobrazuje, kdo je aktuálně k ticketu přiřazen, a proto není nutné vytvářet ještě další sloupec, který by toto rozlišoval.
4. **Testing –** v případě, kdy je ticket vyvinut a interně otestován, předává se do statusu testing, což je impuls pro zákazníka, který nyní musí vyzkoušet chování funkcionality podle předem připravených testovacích scénářů. Pokud je funkcionalit funkční podle vyspecifikovaných potřeb, zákazník potvrzuje funkčnost a posouvá ticket do stavu Ready for Production. Pokud není, ticket jde zpět do fáze In Progress a konzultant s programátorem upravují řešení.
5. **Waiting for Customer –** status pro identifikování User Stories, které čekají na vstupy od zákazníka. Může jít o do upřesnění specifikace, schválení návrhu či jinou podobnou aktivitu.
6. **Ready for Production –** seznam otestovaných požadavků, které jsou připravené k implementaci na produkční prostředí.
7. **Done –** konzultantem uzavřené User Stories, které již jsou transportovány na produkční systém. Takto označené tickety ještě musí zákazník potvrdit, aby došlo k jejich archivaci a zmizely z Kanban tabule.

6.4.3 Definice zadání

V této fázi vývoje softwaru byly identifikovány dvě hlavní překážky. První je způsob popisování zadání jednotlivých User Stories do Jira systému a druhou způsob, jakým spolu komunikují konzultanti, programátoři a žadatel požadavku.

První zmíněný lze pozorovat u většiny požadavků. Tickety jsou sice vytvořené, ale informace, které obnášejí, jsou spíše administrativního rázu. To vychází jak z nastavení struktury formuláře, skrz který se User Stories vytvářejí, tak z disciplíny jednotlivých zákazníků a jejich zvykům. Jako opatření pro zmírnění tohoto problému, jsou navrhnuty dva kroky:

1. **Zrevidovat strukturu formuláře pro založení User Story**
2. **Edukovat zákazníky a vyžadovat dobře strukturovaný popis User Story, který byl představen v kapitole 3.4.3**

Tento postup může zajistit rychlejší orientaci konzultanta / programátora v požadavku a transparentnost informací pro případnou zastupitelnost.

Pro minimalizování druhého problému týkajícího se způsobu komunikace mezi zainteresovanými osobami v rámci řešení ticketu je doporučení takové, že by pro účely do upřesnění zadání či evidence testování měl sloužit pouze nástroj Jira, tak aby byli všichni stejně informovaní a nedocházelo ke zbytečnému zjišťování informací či jejich šíření v okamžiku kdy, již nejsou pravdivé.

6.4.4 Vývoj aplikace

V okamžik, kdy jsou User Stories ve fázi realizace, probíhá častá komunikace mezi konzultantem a programátorem. Konzultant vytváří technické zadání, které programátor reviduje a následně implementuje. Tato komunikace probíhá skrz systém GitLab, čímž se dochází k duplikaci místa řešení User Story a k neefektivnímu využití času, protože oba musí přistupovat do dvou systému, vytvářet duplicitní ticket a udržovat komunikaci na dvou místech. Zde je doporučením zrevidovat přínosy využívání GitLabu a porovnat argumenty propagující tento systém s možným využitím pouze systému Jira. V případě, že by bylo toto doporučení přijato a výsledkem by bylo jednotné využívání Jira systému, muselo by dojít k revizi pracovního postupu a nastavení Kanban tabule, aby byl zajištěný efektivní průchod práce.

6.4.5 Testování a implementace

Jakmile dojde k dokončení vývoje požadované funkcionality je programátor první osobou, která otestuje její funkčnost a následně informuje konzultanta skrz GitLab systém změnou stavu User Story. Zde přebírá ticket opět konzultant, který provede testy dodané funkcionality a její návaznosti na jiné procesy nastavené v systému. Pokud tato část projde přes něj, je informován zákazník, jenž začne s finálním testováním. Zde je poukázáno pouze na nedostatečný přehled o komunikaci během testování mezi konzultantem a zákazníkem, proto je doporučeno navádět konzultanty k tomu, aby tuto formu komunikace vyžadovali. Tato snaha a její progres by měli být revidovány v rámci týmové retrospektivy.

6.4.6 Dokumentace

S ohledem na určenou současnou vizi oddělení, kdy se již rozběhly snahy o zavedení jednotného přístupu k tvorbě dokumentace, neobsahuje tato práce jiný návrh. Jediná rada týkající se této části zasahující do vývoje softwaru, která může podpořit vynakládané úsilí o zavedení dokumentace, bude popsána v kapitole 6.6.1, která je zaměřená na návrh nástroje pro dokumentaci v rámci SAP týmu.

6.4.7 Údržba

Aktuálně nastavené procesy zajišťující údržbu a podporu po nasazení nových funkcionalit na produkční prostředí jsou podle autorova názoru nastavené správně a z toho důvodu není potřeba implementovat nějaká opatření. Pro pokrytí kritických problémů již byla vytvořena Incident procedura, která popisuje role a jejich zodpovědnosti.

6.5 Schůzky a porady

Jedním z hlavních problémů je nízká informovanost, spolupráce a celková komunikace členů v týmu. Z tohoto důvodu jsou v rámci této kapitoly navrženy nové typy schůzek za účelem zvýšení informovanosti jednotlivců, sladění aktivit vedoucích k efektivnější spolupráci a v neposlední řadě pro zlepšení celkového fungování týmu skrz získanou zpětnou vazbu z retrospektiv.

6.5.1 Projektové plánování

Pro zajištění informovanosti nejen přes SAP tým, ale i další vývojové týmy, by bylo vhodné implementovat projektové plánování, jehož cílem by bylo poskytnout informace o realizovaných a nadcházejících projektech lídrům týmů, kteří by následně sdíleli tato zjištění s jejich týmy. Tato schůzka by mohla probíhat buď na měsíční nebo kvartální periodě a obsah sdílených informací by byl na abstraktní úrovni bez zacházení do detailu. Důležité je poznamenat to, že tento typ schůzky nevychází ze zmíněných agilních metodik, z kterých je jinak čerpáno v ostatních částech návrhu. Jde pouze o autorův názor.

6.5.2 Týmové plánování

Týmové plánování už ale vychází z událostí vydefinovaných ve frameworku Scrum. Ačkoli není v rámci návrhu uvažována implementace celého Scrumu, i přesto je možné využít alespoň dílčích částí, které mohou mít pozitivní dopad na fungování SAP týmu. Zároveň musí být vzata v potaz skutečnost, že SAP tým již má zažitý týdenní plánovací mítink, kde se plánují jednotlivé User Stories na následující týden. Po zvážení těchto faktů je výsledným doporučením využít částečně teorie popisující Scrum událost – Sprint Planning. V tomto případě ale nepůjde o stoprocentní obdobu tohoto typu schůzky, protože z organizační struktury vyplývá, že existuje více než jeden Product Owner, který určuje priority. Pro potřeby SAP týmu je tedy doporučeno předat tuto zodpovědnost na SAP týmového manažera, který již obdobnou funkci zastává. Nedostatky jsou v identifikaci Product Ownerů za jednotlivé zákazníky, kteří by prioritní seznam udržovali aktuální. Proto by se měl týmový manažer domluvit se zástupci zákazníků, kdo bude jejich Product Owner a nastavit společné procesy.

Následně si členové týmu vybírají User Stories, které následující týden budou realizovat, podle priorit. Po výběru by měl týmový manažer zajistit aktualizaci statusů jednotlivých ticketů v Kanban tabuli a přesunout je do To-Do sloupce. Jinak lze tedy říct, že obdobná plánovací schůzka k Sprint Planningu již existuje, pouze za jiné terminologie a bez definování cíle v rámci každé iterace.

6.5.3 Týdenní standup meeting

Jde o již zavedený typ schůzky, které jeden z mikro týmů již pravidelně uskutečňuje. V rámci zvyšování kvality spolupráce, odstraňování překážek a sdílení znalostí je doporučeno tuto iniciativu rozšířit i na další týmy rozdělené podle oblastí. Za exekuci tohoto doporučení by měl být zodpovědný týmový manažer, Scrum Master a jednotliví týmový lídři.

6.5.4 Retrospektiva oddělení

Další již částečně fungující a opakující se schůzkou je měsíční retrospektiva oddělení. Částečně fungující proto, že účast na tomto meetingu nabývá klesajícího trendu a převážně týmový lídři SAP týmu nemají příliš velký zájem o participaci na tomto mítinku. Návaznost klesajícího trendu účasti na retrospektivě je ve spojení s neochotou přijmout změny či celou firemní kulturu. Jednotlivci mívají tento mítink za ztrátu času a raději se vymlouvají na jiné události, které líčí jako důležitější. Z těchto důvodů je doporučením klást důraz na účast na této schůzce, mít jasně danou agendu tohoto mítinku a vysvětlit všem účastníkům přínos této schůzky.

6.5.5 Retrospektiva týmu

Retrospektiva je velmi důležitý nástroj pro získání zpětně vazby, která je nezbytná pro posouvání a zlepšování jednotlivců i týmu jako celku. Díky zpětně vazbě a otevřenosti jsou si členové týmu schopni uvědomit nefungující procesy, postupy a praktiky, kterých se drží někteří i řadu let. Návrh tedy obsahuje implementaci retrospektivy i na úrovní SAP týmu, jehož agenda by měla být řízená Scrum Masterem, popřípadě týmovým manažerem. Retrospektiva by měla probíhat na bázi měsíčních cyklů, v případě velkého odporu členů týmu je doporučeno zavést delší prodlevy mezi schůzkami nebo hledat cestu a pochopení k jednotlivcům skrz Scrum Mastera.

6.6 Nástroje

I přesto, že agilní manifesto klade důraz na zaměření se na lidi a interakce, bude v rámci návrhu věnována kapitola doporučení nástrojů a jak je využívat. Agilní přístup tvrdí, že v samoorganizovaných týmech je nejefektivnější, když si sami týmy vytvoří paletu nástrojů, které vhodně doplňují jejich interní procesy a napomáhají k jejich vyšší produktivitě. Z výsledků analýzy současného stavu ale bylo zjištěno, že právě využití nástrojů se jeví jako zásadní problém SAP týmu. Z výčtu všech využívaných systémů a aplikací jednotlivými

členy týmu je příčina zřejmá – tým nemá standardizované interní procesy a nástroje, které k tomu využít.

6.6.1 Návrh nástroje pro dokumentaci

Prvním jasně potvrzujícím důkazem výše zmíněné příčiny jsou systémy pro udržování dokumentace či znalostní báze týmu. Systémy, které jsou nyní pro tuto činnost použity jsou:

- **Word**
- **OneNote**
- **Confluence**
- **Sharepoint**

Doporučením je standardizovat využívání systému Confluence, jehož pořízení bylo iniciováno právě softwarovým oddělením pro tyto účely. Z přechozí věty se může zdát, že toto doporučení je již aplikované a mělo by fungovat. Realita je ovšem jiná. Většina konzultantů a programátorů odmítá aktivně využívat tento nástroj z řady důvodů, s kterými ale nikdo nepracuje, a tak dochází k velmi pomalému až žádném pokroku. Dodatečným doporučením tedy je, aby se této problematice začalo více věnovat vedení týmu či navrhovaná role Scrum Mastera, která by mohla členy týmu vést tímto směrem. Nepochybně jedním z důvodů nepřijetí této změny je nedostatek znalostí a možností systému. Toto lze napravit pravidelným školením, edukací a častou komunikací s jednotlivci pro získání jejich zpětné vazby.

6.6.2 Návrh pro využívání Jira systému

Dalším sofistikovaným nástrojem, který SAP tým používá, je Jira systém. Jde o nástroj plný ideálních funkcionalit využitelných pro agilní řízení projektů a problémů v rámci vývoje softwaru. Z výstupů z rozhovorů a pozorování je zřejmé, že tým sice využívá tento nástroj, ale pouze ve velmi omezené míře. A příčina, proč tomu tak je, je totožná s příčinou nevyužívání Confluence – konzultanti a programátoři neznají plný potencionál tohoto systému. Doporučením je zde tedy obdobné tomu předchozímu. Je nutné zapojit vedení týmu či najít Scrum Mastera, který bude mít dostatek času na edukaci a školení jednotlivců. Jde o zásadní krok, který by mohl díky automatizacím interních procesů, které systém umožňuje, zlepšit hned několik aspektů, které zákazník hodnotil negativně. Zlepšením této problematiky by došlo ke zlepšení komunikace, spolupráce, plánování práce, doby dodání produktu, a hlavně by bylo jasné, jak se systém má používat. Pro podporu tohoto návrhu byla

vytvořena Kanban tabule, která může být prvním krokem pro následnou standardizaci vyžívání nástroje – viz obr. 8.

6.7 Dílčí závěr kapitoly – seznam doporučení

Dílčí závěr kapitoly popisující jednotlivé oblasti životního cyklu vývoje a pro ně vhodná doporučení, na které by se měl tým zaměřit a pokusit se implementovat do svého prostředí. V následující části je seznam identifikovaných překážek, kterým tým čelí a pod nimi je vždy seznam doporučení, která by měla týmu pomoct překonat tyto problémy.

P1 – Komunikace a spolupráce mezi týmy

- Otevření role Scrum Mastera
- Určení zodpovědných Product Ownerů
- Zavést jednotnou Kanban tabuli pro celý tým
- Zrevidovat strukturu formuláře pro založení User Story
- Edukovat zákazníky a vyžadovat dobře strukturovaný popis User Story
- Vyžadovat komunikaci pouze skrz Jira systém
- Implementovat projektové portfolio a plánování

P2 – Plánování práce

- Určení zodpovědných Product Ownerů
- Vést tým ke sdílení znalostí, mít multifunkční tým
- Sjednotit Produktový Backlog pro všechny typy požadavků
- Zavést jednotnou Kanban tabuli pro celý tým
- Implementovat projektové portfolio a plánování

P3 – Nástroje

- Zavést jednotnou Kanban tabuli pro celý tým
- Edukovat lidi o možnostech nástroje Jira
- Zrevidovat přínosy využívání GitLabu a porovnat argumenty propagující tento systém s možným využitím pouze systému Jira
- Standardizovat týmové procesy a nástroje – Jira a Confluence

P4 – Neochota přistoupit na změny / kultura organizace

- Edukovat členy týmu o agilních metodikách a přínosu
- Zajistit prostředí pro možnost využití agilních metodik

P5 – Neznalost agilních metodik

- Otevření role Scrum Mastera
- Edukovat a vést tým skrz pomocí využití agilních praktik, her a dalších technik, které pomohou jednotlivcům lépe pochopit agilní přístup

P6 – Motivace

- Neustále vysvětlovat lidem přínosy agilního přístupu, vymýtit negativní názory
- Vést tým k samoorganizaci, dát lidem možnost vykonávat rozhodnutí

P7 – Chybí proces pro neustálé zlepšování

- Zavést týdenní standup schůzky v rámci mikro týmů
- Zavést retrospektivu a pracovat s výstupy z těchto schůzek

P8 – Pozdní dodání produktu

- Otevření role Scrum Mastera
- Vést tým ke sdílení znalostí, mít multifunkční tým
- Optimalizovat prioritizační proces, zavést projektové portfolio
- Zavést Kanban tabuli, metriky průtoku práce a proces neustálého zlepšování vycházející z principů Kanbanu

7 Shrnutí výsledků

V první části se diplomová práce zaměřila na popis tradičního a agilního způsobu vývoje softwaru. Následně byla provedena analýza současného stavu v rámci fungování jednoho z týmů softwarového oddělení nejmenované výrobní společnosti. Bylo provedeno několik osobních rozhovorů se zaměstnanci pracujícími ve zkoumaném týmu, na jejich základě byly identifikovány zásadní problémy, kterým tým čelí a které jim brání v podávání lepší výkonnosti. Hlavním cílem práce byla identifikace těchto reálných problémů a následné vytvoření vhodných opatření na základě agilních metodik, která by měl tým implementovat do svého prostředí pro zlepšení situace.

V první řadě byl popsán vznik oboru softwarového inženýrství, jenž se datuje k šedesátým letem minulého století. S rychlým růstem poptávek na vývoj softwaru vznikaly také problémy, které vyústily v softwarovou krizi, díky které vznikly tradiční metodiky vývoje softwaru. Tyto metodiky udávaly kurz, kterým se softwarové společnosti řídili a některé řídí i dnes. Jelikož spousta nedostatků v rámci řízení softwarového vývoje přetrvávala i po zavedení těchto metodik, vznikly metodiky agilní, které přinesli odlišný pohled a přístup na životní cyklus vývoje softwaru.

Pro identifikace hlavních problémů týmu bylo nejprve nezbytné provést hlubší analýzu jednotlivých částí vývojového procesu, který tým aplikuje při realizaci požadavků zákazníka. Kromě těchto procesů byli metodou pozorování prozkoumány i doplňující praktiky, techniky a nástroje, které tým využívá. Z popisu je zřejmé, že tým byl okrajově seznámen z agilními metodikami, neboť některé ceremoniály a artefakty využívá už za současného stavu. Ovšem z výstupů z rozhovorů vychází, že jednotlivý členové týmu vlastně neznají teorii a přínos aplikování agilních metodik. Výsledkem této části je seznam zobrazující osm zásadních problémů, kterým tým čelí a k nim doporučená opatření.

Pro sestavení účelného návrhu bylo využito dalších dvou vstupů. Prvním byl výstup z dotazníkového šetření, které je každým rokem realizováno softwarovým oddělením zkoumané společnosti. Díky tomu bylo zřejmé, které oblasti a kritéria spolupráce jsou pro zákazníky důležité a zároveň, kde je největší prostor pro zlepšení. Druhým poznáním byly výsledky z externího dotazníkového šetření realizovaného společností Digital.ai, která publikuje výsledky tohoto průzkumu od roku 2006. Výstupem z tohoto průzkumu byla identifikace aktuálních trendů ve světě v rámci využívání agilních praktik, technik a nástrojů, na jejichž základě byla připravována následná doporučení v šesté kapitole.

Dále se praktická část zaobírala tvorbou návrhu doporučení na základě kombinování vstupů z předchozích průzkumů a analýz s ohledem na specifické prostředí SAP týmu. Cílem nebylo navrhnout implementaci některé z agilních metodik se stoprocentním dodržením její teorie, ale spíše využít jednotlivých částí, které vycházejí z agilních metodik a zároveň jsou aplikovatelné do komplexního prostředí, kde SAP tým působí. Návrhy byly tvořeny na podstatě a principech tří metodik – Scrum, Kanban a ScrumBan, které patří mezi čtyři nejvyužívanější agilní metodiky podle State of Agile Reportu z roku 2021. V kapitole šest jsou popsány jednotlivá doporučení podle oblastí, ke které se vážou. Kapitola je uzavřena tabulkou problémů a seznamem doporučení, která by měla vést ke zlepšení a odstranění identifikovaných překážek.

Diplomová práce splnila definovaný cíl. Hlavní přínos tvoří seznam překážek, kterým SAP tým čelí a seznam doporučení, která by měla být implementována pro eliminaci těchto problémů v rámci dané společnosti.

8 Závěry a doporučení

Motivací pro vypracování této diplomové práce byla vize autora, který se pohybuje v prostředí analyzovaného softwarového oddělení, jenž má za cíl zvýšit efektivitu při řízení vývoje procesu v rámci SAP týmu, zlepšit týmovou spolupráci a zvýšit povědomí o agilních metodikách a praktikách.

Praktická část byla postavena na základech teoretického poznání tradičních a agilních přístupů získaných nejen z odborné literatury, ale také z méně odborných internetových zdrojů, typicky blogů, které jsou zaměřené na agilní metodiky.

Definovaná doporučení jsou stanovena pro zkoumaný SAP tým, a proto není vhodné využít výstupů z této práce pro jiné týmy, neboť každý tým je složen z jiných jedinců, kteří tvoří jinak fungují celek a čelí tak odlišným překážkám. Tato diplomová práce však může vést jiné týmy k tomu, jak postupovat v případě uvědomění, že něco nefunguje, nebo potřeby dělat něco lépe.

V rámci diplomové práce nedošlo k samotné implementaci definovaných doporučení, ale je to osobním cílem autora. Po nasazení těchto doporučení a následné revizi jejich přínosů, je doporučeno pokračovat ve směru prosazování povědomí o agilním přístupu a hledat možnosti a prostory pro neustálé zlepšování, neboť přesně o tom agilní metodiky jsou.

9 Seznam grafů

10 Seznam obrázků

11 Seznam tabulek

12 Citovaná literatura

Agile Alliance. What is Extreme Programming (XP)? *Agilealliance.* [Online] Agile Alliance. [Citace: 7. Srpen 2021.] Dostupné z: https://www.agilealliance.org/glossary/xp/.

Ahmad, Muhammad Ovais, Markkula, Jouni a Oivo, Markku. 2013. *Kanban in software development: A systematic literature review.* Santander : IEEE, 2013. ISBN 978-0-7695-5091-6.

Aljaber, Tareq. The iron triangle of planning. *Atlassian.* [Online] Atlassian.com. [Citace: 10. Srpen 2021.] Dostupné z: https://www.atlassian.com/agile/agile-at-scale/agile-iron-triangle.

Beck, Kent. 2002. *Extrémní programování.* Praha : Grada Publishing, spol. s.r.o., 2002. ISBN 80-247-0300-9.

Digital.ai. 2021. 15th State of Agile Report. *explore.digital.ai.* [Online] Digital.ai, 9. Červenec 2021. [Citace: 8. Srpen 2021.] Dostupné z: https://explore.digital.ai/state-of-agile/15th-state-of-agile-report.

Germanov, Iliyan. 2019. Kanban vs Scrum vs Scrumban: What Are The Differences? *Ora.* [Online] Ora.pm, 12. Srpen 2019. [Citace: 20. Červenec 2021.] Dostupné z: https://ora.pm/blog/scrum-vs-kanban-vs-scrumban/.

Holbeche, Linda. 2018. *The Agile Organization: How to build an engaged, innovative and resilient business.* Croydon : CPI Group (UK) Ltd., 2018. ISBN 978-0-7494-8265-7.

Kadlec, Václav. 2004. *Agilní programování: metodiky efektivního vývoje softwaru.* Brno : Computer Press, 2004. ISBN 80-251-0342-0.

Kniberg, Henrik. 2007. *Scrum and XP from the Trenches: How we do Scrum.* místo neznámé : C4Media Inc., 2007. ISBN 978-1-4303-2264-1.

Ladas, Corey. 2008. *Scrumban - Essays on Kanban Systems for Lean Software Development.* Seattle : Modus Cooperandi Press, 2008. ISBN 978-0-578-00214-9.

Myslín, Josef. 2016. *Scrum: průvodce agilním vývojem softwaru.* Brno : Computer Press, 2016. ISBN 978-80-251-4650-7.

Ponomareff, Dimitri. 2017. Scrumban - Blending Agile, Scrum and Kanban into a methodology that works for you. *Kanban Zone.* [Online] Kanban Zone, 21. Září 2017. [Citace: 3. Srpen 2021.] Dostupné z: https://kanbanzone.com/2017/scrumban-blending-agile-scrum-kanban/.

Radigan, Dan. Working with WIP limits for kanban. *Atlassian.* [Online] Atlassian.com. [Citace: 4. Srpen 2021.] Dostupné z: https://www.atlassian.com/agile/kanban/wip-limits.

Reddy, Ajay. 2015. *The Scrumban [R]Evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban.* Crawfordsville : Pearson Education, Inc., 2015. ISBN 978-0134086217.

Scrum. What is a Developer in Scrum? *Scrum.* [Online] Scrum.org. [Citace: 1. Srpen 2021.] Dostupné z: https://www.scrum.org/resources/what-is-a-scrum-developer.

—. What is a Sprint Retrospective? *Scrum.* [Online] Scrum.org. [Citace: 2. Srpen 2021.] Dostupné z: https://www.scrum.org/resources/what-is-a-sprint-retrospective.

Shore, James a Warden, Shane. 2008. *The Art of Agile Development.* Sebastopol : O'Reilly Media, Inc., 2008. ISBN 978-0-596-52767-9.

Sommervile, Ian. 2013. *Softwarové inženýrství.* Brno : Computer Press, 2013. ISBN 978-80-251-3826-7.

Šochová, Zuzana a Kunce, Eduard. 2019. *Agilní metody řízení projektů.* Brno : Computer Press, 2019. ISBN 978-80-251-4961-4.

Šochová, Zuzana. 2018. *Skvělý ScrumMaster.* Brno : Computer Press, 2018. ISBN 978-80-251-4927-0.

The Agile Manifesto. 2001. Agilemanifesto. *Manifesto for Agile Software Development.* [Online] 2001. [Citace: 8. Srpen 2021.] Dostupné z: https://agilemanifesto.org/.

Verheyen, Gunther. 2013. *Scrum - A Pocket Guide.* Zaltbommel : Van Haren Publishing, 2013. ISBN 978-90-8753-720-3.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Pavel Bílek**
Studium:	I1800721
Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	Informační management
Název diplomové práce:	**Využití agilních metodik v projektovém řízení zaměřené na vývoj SW**
Název diplomové práce AJ:	Use of agile methodologies in project management focused on SW development

Cíl, metody, literatura, předpoklady:

Rešerše metodik, výhody/nevýhody, kombinace, praxe, case study, doporučení pro firmu

Sommerville, I. (2013). *Softwarové inženýrství*. Computer Press, Albatros Media as.

Kunce, E., & Šochová, Z. (2019). *Agilní metody řízení projektů*. Computer Press.

Březina, A. (2020). Agilní transformace Proč bývá tak křehká?, Kopp.

Arlow, J., & Neustadt, I. (2007). *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Computer Press.

Garantující pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	doc. Ing. Hana Tomášková, Ph.D.
Oponent:	doc. Ing. Pavel Čech, Ph.D.

Datum zadání závěrečné práce: 21.1.2020