VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV MANAGEMENTU

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF MANAGEMENT

APLIKACE AGILNÍ METODIKY SCRUM A VYUŽITÍ PODPŮRNÝCH SOFTWAROVÝCH NÁSTROJŮ

UTILIZATION OF AGILE SCRUM AND USAGE OF SUPPORT SOFTWARE TOOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ TOŠNER
AUTHOR

VEDOUCÍ PRÁCE — Ing. LENKA SMOLÍKOVÁ, Ph.D.
SUPERVISOR

BRNO 2015

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2014/2015
Ústav managementu

ZADÁNÍ DIPLOMOVÉ PRÁCE

Tošner Jiří, Bc.

Řízení a ekonomika podniku (6208T097)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

Aplikace agilní metodiky Scrum a využití podpůrných softwarových nástrojů

v anglickém jazyce:

Utilization of Agile Scrum and Usage of Support Software Tools

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Návrh řešení a přínos návrhů řešení
Závěr
Seznam použité literatury

Seznam odborné literatury:

BUCHALCEVOVÁ, Alena. Metodiky budování informačních systémů. Vyd. 1. V Praze: Oeconomica, 2009, 205 s. ISBN 9788024515403.
DOLEŽAL, Jan, Pavel MÁCHAL a Branislav LACKO. Projektový management podle IPMA. 1. vyd. Praha: Grada, 2009, 507 s. Expert (Grada). ISBN 9788024728483.
PROCHÁZKA J. a C. KLIMEŠ, 2011. Provozujte IT jinak: Agilní a štíhlý provoz, podpora a údržba informačních systémů a služeb. Praha: Grada Publishing. 288 s. ISBN 978-80-247-4137-6.
RUBIN, K. S., 2012. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional. 452 s. ISBN 9780137043293.
ŠOCHOVÁ, Zuzana a Eduard KUNCE. Agilní metody řízení projektů. 1. vyd. Brno: Computer Press, 2014, 175 s. ISBN 9788025141946.

Vedoucí diplomové práce: Ing. Lenka Smolíková, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2014/2015.

L.S.

prof. Ing. Vojtěch Koráb, Dr., MBA	doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Ředitel ústavu	Děkan fakulty

V Brně, dne 28.2.2015

ABSTRAKT

Agilní metodiky vývoje software jsou velmi populární pro jejich efektivitu a flexibilitu. Tato diplomová práce je zaměřena na agilní metodiku vývoje software nazvanou Scrum. Nejprve je uvedena základní charakteristika a srovnání tradičních a agilních metodik. Pozornost je věnována zejména zmíněné metodice Scrum, která je v praktické části znázorněna na příkladu využití konkrétní firmou.

K organizaci metodiky Scrum je vhodné využít některý ze softwarových nástrojů. Proto je uveden také přehled a základní srovnání nejpoužívanějších podpůrných softwarových nástrojů, pro organizaci této metodiky. Závěrem práce jsou doporučení ke zlepšením pro zkoumaný tým, na základě identifikovaných nedostatků.

KLÍČOVÁ SLOVA

agilní metodiky, vývoj software, Scrum

ABSTRACT

Agile software development methodologies are very popular for their efficiency and flexibility. This thesis focuses on agile software development methodology called Scrum. Basic description and comparison of traditional and agile methodologies is mentioned at first. Attention is paid to methodology Scrum which is shown on an example of usage by a specific company.

It is convenient to use some software tool for organization of Scrum methodology. Therefore, an overview of the most common tools and basic comparison of software tools for organizing this methodology is presented. Conclusion of this thesis is recommendation for improvements for the investigated team, based on issues which were identified.

KEYWORDS

agile methodologies, software development, Scrum

BIBLIOGRAFICKÁ CITACE

TOŠNER, J. *Aplikace agilní metodiky Scrum a využití podpůrných softwarových nástrojů.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2015. 81 s. Vedoucí diplomové práce Ing. Lenka Smolíková, Ph.D..

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 20. května 2015

……

podpis studenta

PODĚKOVÁNÍ

Rád bych poděkoval všem členům zkoumaného vývojového týmu, za cenné zkušenosti s agilní metodikou Scrum. Díky patří také vedoucí této diplomové práce, která mi pomohla svými cennými radami.

OBSAH

ÚVOD

„Projekt je jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení předem stanoveného cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji.“ Takto definuje projekt Mezinárodní organizace pro standardizaci. K naplnění stanoveného cíle projektu v rámci omezení časem i zdroji je však nutné perfektně zvládnout jeho řízení. Jakou metodou toho však docílit? (1 str. 2)

Projektové řízení v oblasti vývoje software vycházelo do konce 20. století z tzv. rigorózních, neboli těžkých metodik (typický vodopádový model). Ty byly často kritizovány pro nedostatek flexibility a nadbytek byrokracie. Na přelomu 20. a 21. století však došlo k průlomu. Firmy i jednotlivci začali pracovat na nových možnostech řízení vývoje software a shodou okolností měla většina nových modelů hodně společného: individuální přístup, rozdělení produkčního období do krátkých cyklů, omezení byrokracie, častou komunikaci se zákazníkem apod. V roce 2001 se sešlo několik zástupců z oblasti vývoje software za účelem sjednocení těchto postupů. Výsledkem této schůzky byl vznik tzv. „Manifestu agilního vývoje software“. Tím byla oficiálně založena disciplína tzv. agilních, nebo také lehkých metodik. (2)

Vznik agilních metodik však stále neznamená úplný přechod softwarového vývoje na agilní způsob. Pro některé typy projektů jsou stále vhodné tradiční metodiky, které mají stále své výhody.

Od vzniku kategorie agilních metodik již uběhlo téměř 15 let. Za tu dobu vzniklo několik konkrétních druhů agilních metodik a dokonce i jejich poddruhů. Každý z těchto modelů má však své pozitivní i negativní stránky, a každý z modelů je vhodný pro jiný druh vývoje, produktu, složení týmu, nebo i požadavků trhu. Výběr nejvhodnějšího modelu je tedy velmi důležitým rozhodnutím projektových manažerů, nebo manažerů vývojového týmu.

Tato práce se zabývá jednou z nejpopulárnějších agilních metodik, která se nazývá Scrum development process. Je to jedna z mála metodik, která definuje nejen myšlenku, ale popisuje organizaci celého procesu vývoje. Scrum definuje i doporučené složený

vývojového týmu a role a úkoly jednotlivých členů, včetně samotného zákazníka. Teorie této metodiky se může zdát velmi jasná a snadno implementovatelná do praxe. Jak však uvidíme v praktické části této práce, i vývojové týmy nadnárodních společností mohou mít v některých oblastech této metodiky nedostatky, které snižují výslednou agilitu celého vývoje.

V této práci uvidíte, mimo teoretického přehledu o agilních a tradičních metodikách, jak metodiku Scrum využívá konkrétní vývojový tým velké společnosti. Dočtete se, které oblasti Scrum procesu jsou zde nastaveny podle doporučení ve většině literatury, ale také o oblastech, které nejsou vhodně nastaveny. Tyto nedostatky budou vysvětleny a budou navrženy vhodné změny, ke zdokonalení vývoje metodikou Scrum.

Informace dostupné v této práci mohou posloužit jiným vývojovým týmům, které chtějí zdokonalit využití metodiky Scrum, nebo tuto metodiku zavést ve své společnosti. Práce může posloužit samozřejmě také jednotlivcům, kteří chtějí rozšířit své znalosti o této agilní metodice.

1 CÍLE PRÁCE, METODY A POSTUPY ZPRACOVÁNÍ

1.1 Cíle práce

Teoretická část této diplomové práce bude zaměřena na tradiční a agilní metodiky. Bude uvedena stručná charakteristika základních tradičních i agilních metodik a srovnání hlavních principů těchto směrů. Podrobněji bude popsán zejména agilní model Scrum development process, na který je tato práce zaměřena.

V praktické části bude nejprve představena konkrétní společnost a vývojový tým, u kterého bude zkoumán proces řízení vývoje software. Bude provedena analýza současného způsobu využití metodiky Scrum u zkoumaného týmu, ve srovnání s teorií. Před hlavní částí této práce bude také uveden přehled vybraných softwarových aplikací pro metodiku Scrum, včetně jejich srovnání. Hlavním přínosem této práce budou doporučení ke zlepšení současného procesu, k efektivnějšímu a agilnějšímu vývoji. Ke každému doporučení bude uveden i očekávaný přínos ze změny a také s ní spojené náklady. Podstatná část práce bude zaměřena na využití podpůrných nástrojů pro metodiku Scrum.

Hlavním cílem této práce je tedy zhodnocení způsobu využití modelu Scrum konkrétním vývojovým týmem, srovnání teorie s praktickými zkušenostmi a doporučení změn, které povedou ke zdokonalení procesu vývoje metodikou Scrum a zvýšení agility.

1.2 Metody a postupy zpracování

V rámci teoretické části jsem nejprve nastudoval odbornou literaturu, zabývající se tradičním a agilním směrem vývoje. Na základě získaných poznatků jsem vytvořil popis základních metodik a srovnání tradičního a agilního směru. Studium literatury jsem následně zaměřil na detaily agilní metodiky Scrum a doporučené postupy.

V praktické části jsem nejprve dohledal veřejně dostupné informace o zkoumané společnosti. Jelikož jsem zaměstnancem této společnosti, mohl jsem pozorováním získat informace o současném způsobu organizace týmů a fungování vývoje zkoumaným

vývojovým týmem. Dalším zdrojem informací byly praktické poznatky a zkušenosti členů vývojového týmu a product ownerů, které jsem získal přímým dotazováním. Informace získané tímto pozorováním a přímým dotazováním jsem shrnul v analýze současného stavu.

Následně jsem studoval dostupné internetové zdroje a články, k získání přehledu o dostupných softwarových nástrojích pro metodiku Scrum. Na základě dostupných informací jsem zvolil několik nejvýznamnějších zástupců a vytvořil souhrn podstatných informací a ukazatelů o těchto aplikacích. Metodou srovnání jsem následně zhodnotil vhodnost těchto nástrojů pro různé druhy vývoje a velikosti týmů.

Srovnáním teoretických poznatků z teoretické části s výsledky pozorování současného způsobu využití metodiky Scrum, jsem identifikoval nedostatky. K těmto nedostatkům jsem, s využitím informací z nastudované literatury a analýzy dostupných softwarových nástrojů a doporučil vhodná řešení. U navržených řešení jsem identifikoval předpokládané náklady a přínosy, přičemž některé údaje jsem zjistil z interních zdrojů a některé z veřejně dostupných. Současnou situaci a předpokládaný stav po doporučených změnách jsem ohodnotil soustavou ukazatelů, které jsem stanovil subjektivně, na základě vlastního hodnocení. Tato hodnocení jsem pro přehlednost znázornil také pomocí paprskových grafů. V závěru jsem shrnul nejdůležitější poznatky zmíněné v této práci a zmínil identifikované nedostatky současného stavu s doporučeným řešením.

2 TEORETICKÁ VÝCHODISKA PRÁCE

2.1 Tradiční metodiky vývoje

Softwarové inženýrství je disciplína, která se zabývá problémy při vývoji komplexních softwarových systémů. Rozlišujeme tradiční (rigorózní) a agilní (lehké) metodiky. Do každé z těchto dvou skupin můžeme zařadit několik různých modelů.

Před vznikem agilních metodik, ke kterému oficiálně došlo na přelomu 20. a 21. století, byly k vývoji software využívány tradiční metodiky projektového řízení. Pro znázornění důvodů vzniku agilních metodik, si nejprve uvedeme stručnou charakteristiku tradičních metodik.

Tradiční metodiky vývoje kladou důraz na přesnou specifikaci všech požadavků na počátku projektu a na jejich podrobnou dokumentaci a smluvní odsouhlasení. Koncový zákazník není ve styku s vývojovým týmem, takže nemůže do procesu vývoje nijak zasahovat. V případě změn požadavků v průběhu začíná zpravidla celý proces od začátku. Vývojový tým se řídí pouze dokumentací. Zákazník dostává objednaný software po částech, nebo celý až na konci.

2.1.1 Vodopádový model

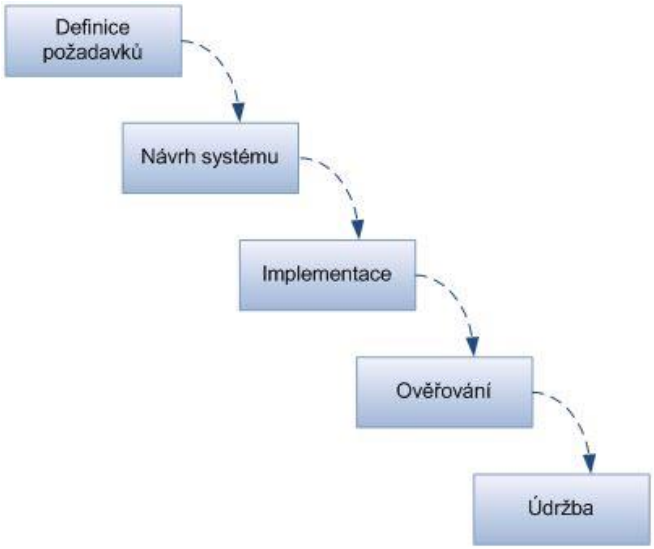
Historicky nejstarším a také základem téměř všech modelů je tzv. Vodopádový model. Různé modifikace tohoto modelu můžeme nalézt ve většině současných modelů. Spočívá v rozdělení celého procesu vývoje sekvenčně do jednotlivých etap, bez iterací[1] (viz. Obr. 1). Zásadní podmínkou zde je, že nesmíme překročit k následující etapě, dokud není předchozí etapa kompletně dokončena a schválena. Je také nutné postupovat v posloupnosti kroků, vždy pouze o jednu úroveň. Pokud však během některé fáze zjistíme problém, je možné se vrátit o úroveň zpět a tento předchozí krok opakovat (3 stránky 66 - 68).

Tento model byl považován za průlom, v oblasti softwarového inženýrství. Jeho největší nevýhodou však je, že pokud v průběhu vývoje dojde ke změně požadavků,

[1] **Iterace** může být použito ve stejném významu jako *opakování* (lat. *iteretur* – opakovat). Principem iterace je opakování určitého procesu v měnícím se kontextu (v dynamických jevech).

musíme projít celý proces od začátku, vše znovu zdokumentovat, schválit atd. Obdobně se postupuje také v případě, kdy dojde k novým poznatkům, nebo problému, při vykonávání některého z kroků. V takovém případě se proces vrací o jeden, či více kroků zpět k přepracování.

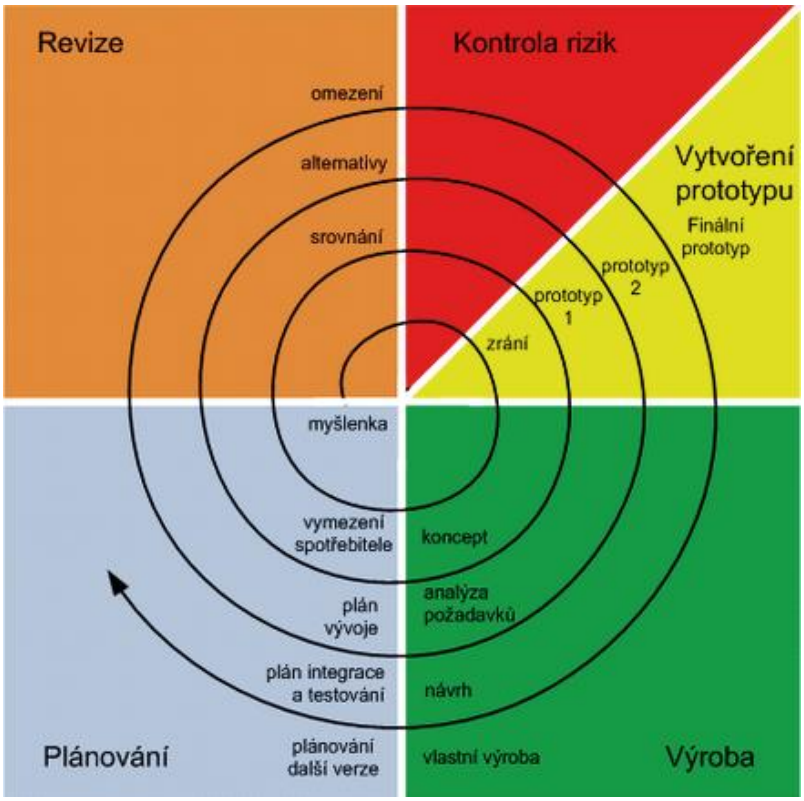
Zákazník po celou dobu průběhu projektu neví, kdy bude projekt skutečně dokončen a co vlastně vývojový tým právě dělá. Stále se však tento model využívá v menších firmách u menších projektů, pro jeho jednoduchost.



Obr. 1 Vodopádový model, zdroj: (4)

2.1.2 Spirálový model

Dalším významným zástupcem tradičních metodik je Spirálový model. Tento model se skládá ze čtyř fází (Plánování, Vyhodnocení, Analýza rizik a Zpracování), které se několikrát opakují (viz. Obr. 2). Je zde tedy zaveden iterativní – cyklický – přístup. Díky několikanásobnému průchodu všemi fázemi je možné v každé fázi lépe pochopit problémy a odstranit je při dalším průchodu. K úspěchu projektu díky tomu není nutné mít kompletní a přesnou specifikaci požadavků už v prvním kroku, ale je možné je v jednotlivých cyklech dopracovávat a upřesňovat. Dalším zlepšením oproti vodopádu je opakující se analýza rizik, díky které je možné už v průběhu projektu odhalit skryté problémy, které by mohly ohrozit úspěch projektu. Díky tomu je tento model někdy řazen do skupiny „risk-driven approach“ (3 stránky 73 - 74). Nevýhodou tohoto modelu je však jeho komplexnost, díky které se nehodí pro menší projekty.



Obr. 2 Spirálový model, zdroj: (4)

2.1.3 Rational Unified Process (RUP)

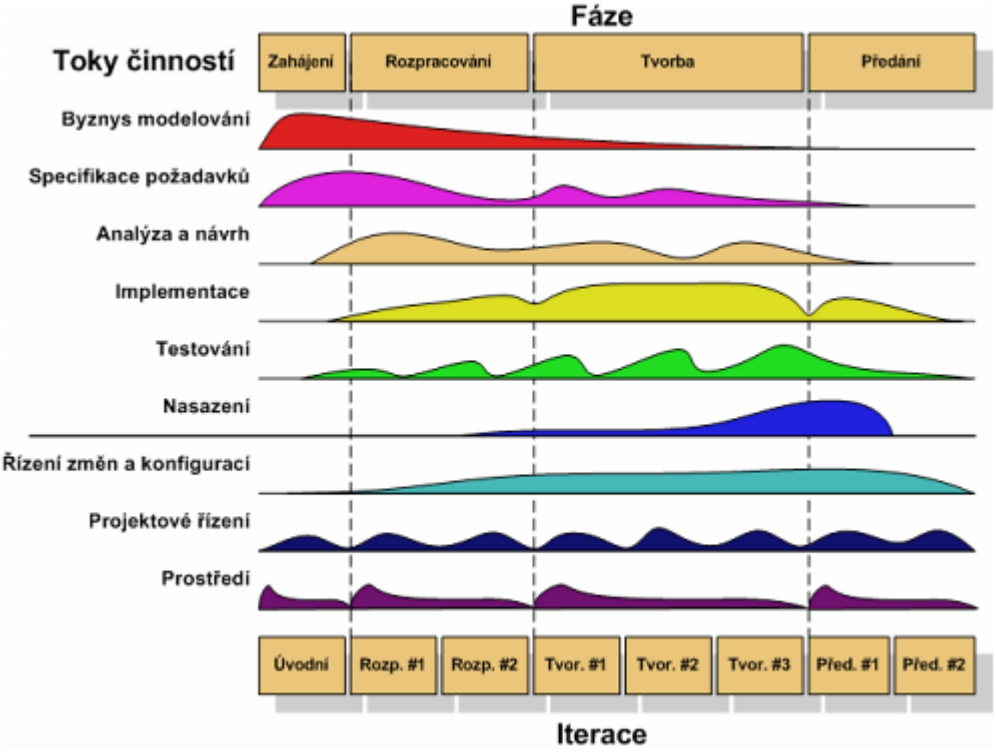
Model RUP pracuje s iterativními cykly, které na sebe navazují. Na konci každého cyklu (iterace) je výstupem funkční produkt, který se v dalších cyklech doplňuje o další požadované funkce (viz Obr. 3). Systém je tak vyvíjen ve verzích, které lze průběžně ověřovat a případně i měnit (3 stránky 82 - 85).

Obr. 3 Iterace vývoje produktu, zdroj: (5 str. 13)

Velkou výhodou tohoto systému je průběžné dodávání funkčních částí produktu. V případě předčasného ukončení projektu má zákazník za doposud vynaložené prostředky a čas alespoň částečně funkční produkt.

Tato metoda je použitelná pro jakýkoliv rozsah projektu, díky možnosti ji upravit pro specifické potřeby projektu. Vhodná je však spíše pro větší projekty a týmy, jelikož klade důraz na analýzu, plánování, řízení a dokumentaci.

Jednotlivé cykly RUP jsou rozděleny do 4 fází: zahájení, rozpracování, tvorba a předání. Obr. 4 znázorňuje, které činnosti jsou prováděny v jednotlivých fázích jednoho RUP cyklu.



Obr. 4 Schéma procesu RUP, zdroj: (5 str. 12)

Model RUP je úzce propojen s programovacím jazykem UML, který umožňuje vytváření softwarových modelů. V průběhu času se UML stal jazykem, obecně využívaným k dokumentaci softwarové architektury (5 str. 15).

2.2 Agilní metodiky vývoje

Moderní trendy ukazují potřebu rychlejšího a flexibilnějšího vývoje software, s vynaložením co nejnižších nákladů. Klasické metody si však zakládají na podrobném plánování, dokumentaci a přesné definici požadavků před začátkem vývoje. Tato neshoda vedla k potřebě nových metodik, pružnějších a rychlejších, které by byly schopny reagovat na měnící se potřeby i v průběhu projektu.

2.2.1 Manifest Agilního vývoje

Roku 2001 skupina 17 odborníků z oboru softwarového inženýrství sepsala tzv. „Manifest Agilního vývoje software“. Tímto manifestem oficiálně vznikla sjednocená kategorie Agilních[2] metodik vývoje software. Zároveň byla také vytvořena Aliance pro agilní vývoj. Ve výše zmíněném manifestu se zakladatelé shodli na čtyřech hlavních hodnotách agilního vývoje:

*„**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.“ (2)

Agilní manifest také obsahuje **12 principů agilního vývoje** (6):

1. Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru. Pro zákazníka má vyšší hodnotu funkční část kódu, než grafický (UML) návrh architektury a modulů.
2. Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka. Agilní metodika změny přímo očekává, proto se nepracuje na ničem, co není právě potřeba.

[2] **Agilní**, z anglického jazyka můžeme přeložit jako *hbitý*, nebo *mrštný*. *Agilní metodiky* se také někdy nazývají jako *lehké* (*„light-weight“*).

3. Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody. Oproti tradičním metodám zde trváme na co nejkratších iteracích.

4. Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu. Díky tomuto principu je možné v průběhu vývoje postupně upřesňovat požadavky a podle toho přímo ovlivňovat právě probíhající vývoj.

5. Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci. Lidský faktor je klíčem k úspěchu a je nutné důvěřovat ve schopnosti jedinců.

6. Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace. Agilní přístupy upřednostňují osobní kontakt před komunikací psanou formou.

7. Hlavním měřítkem pokroku je fungující software. Tento princip upozorňuje na nutnost hodnocení výsledku zejména podle funkčnosti vytvořeného softwaru.

8. Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale. Krátkodobá nutnost přesčasů k úspěchu projektu je přípustná. Pokud je tato potřeba dlouhodobá, je to signálem problému v projektu.

9. Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu. Designu softwaru je nutné věnovat pozornost na denní bázi, jelikož časté změny požadavků v průběhu vývoje mohou znamenat také změnu celkového designu.

10. Jednoduchost – umění maximalizovat množství nevykonané práce – je klíčová. K úspěšnému zvládnutí neustálých změn je nutné, aby veškeré procesy a postupy byly co nejjednodušší.

11. Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů. Důvěra a komunikace vede ke kreativitě, která je základem dobrých návrhů.

12. Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti. Je nutné tedy neustále hledat možnosti zlepšení a zefektivnění.

Pro agilní přístup je klíčová schopnost průběžných dodávek funkčních verzí systému v krátkých časových intervalech. Díky tomu je možné průběžně měnit požadavky a zajistit požadovanou kvalitu výsledku. Vzhledem ke krátkým cyklům je také vysoká pravděpodobnost, že dodržíme časový plán a náklady. Oproti tomu u tradičních metod je pro dlouhou dobu trvání projektu velká nejistota dodržení časového harmonogramu i nákladů. Problematické řešení změn v průběhu projektu také znamená nejistotu funkčnosti software podle aktuálních potřeb zákazníka, na konci projektu. Znázorněno obrázkem Obr. 6 strana 23.

Pro agilní vývoj je typický **iterativní a inkrementální vývoj s krátkými iteracemi**, kdy vývoj probíhá v krátkých cyklech a nové funkcionality jsou dodávány postupně. Zákazník tak průběžně získává přidanou hodnotu a může obratem poskytovat zpětnou vazbu ohledně dalšího vývoje a potřebných změn. Tím je eliminována nejistota, že na konci dostane něco, co vlastně nechtěl. **Komunikace mezi zákazníkem a vývojovým týmem** je tedy nutná k vývoji softwaru přesně podle potřeb zákazníka a k odstranění nedostatků a případných chyb. Zákazník by se měl také podílet na plánování jednotlivých iterací. Dalším faktorem je **průběžné testování.** To je vzhledem k dynamicky se měnícímu systému a požadavkům nezbytné provádět **průběžně a automatizovaně**. Testovací plán by měl být připraven už před implementací nové části, k eliminaci časové prodlevy. Vždy je také nutné testovat celý systém, ne jen nově implementovanou část. (7 str. 33)

Z výše uvedených poznatků tedy vyplývá, že hlavním principem agilních metodik je co nejrychleji vyvinout systém podle předběžných požadavků a na základě zpětné vazby od zákazníka systém upravit do finální podoby. Umožnit změny je totiž efektivnější, než se jim snažit zabránit byrokracií.

Čím více zapojíme zákazníka do procesu plánování, tím větší přidanou hodnotu pro něj bude mít výsledek na konci každého sprintu. Dokumentace požadavků nehraje zásadní roli, oproti tomu schopnost dynamicky software upravovat podle potřeb je naprosto zásadní.

Z principu agilního vývoje vzniklo mnoho konkrétních myšlenek a modelů. Několik základních bude představeno v následujících podkapitolách.

2.2.2 XP – Extreme programming

Extrémní programování je jednou z nejznámějších agilních metodik. Mnoho lidí se dokonce mylně domnívá, že agilní vývoj znamená XP. Tato metoda je jakýmsi extrémem agilního programování. Snaží se dovést do extrému vše, co se osvědčuje. Například, pokud se osvědčuje párové programování, budeme ho provádět neustále, pokud se osvědčuje testování, budeme testovat neustále a všichni. Tento pohled zaměřuje hlavně na tyto body:

- jednoduchost (maximálně jednoduchý systém, který splňuje požadavky)
- průběžné revize (párové programování, vzájemná kontrola)
- nové návrhy (pokud se osvědčují, neustále navrhovat zlepšení - refaktorizace)
- testování
- extrémně krátké iterace (těsná spolupráce se zákazníkem a dodávání po nejmenších částech, iterace v minutách až hodinách)

Metoda XP vychází z přesvědčení, že jediným správným a nezpochybnitelným řešením je aktuální zdrojový kód. Je vhodná pro menší projekty a týmy. (8)

2.2.3 FDD – Feature-driven development

Vývoj řízený vlastnostmi se zaměřuje na vývoj po velmi krátkých iteracích (typicky 2 týdny), kdy výsledkem každé iterace je jedna, či více, dílčích funkcí, které mají přidanou hodnotu pro zákazníka. Základním stavebním kamenem jsou zde tedy vlastnosti výsledného produktu. FDD umožňuje velmi dobré sledování průběhu projektu, díky dodávkám hotových částí každé 2 týdny.

Tato metodika využívá, na rozdíl od XP, objektové modelování pomocí jazyka a diagramů UML. Proces FDD začíná vytvořením obecného modelu, který se pak převádí do podoby seznamu funkcionalit. Tyto dvě fáze jsou sekvenční. Následující 2 fáze (návrh a implementace) jsou pak iterativní. (9)

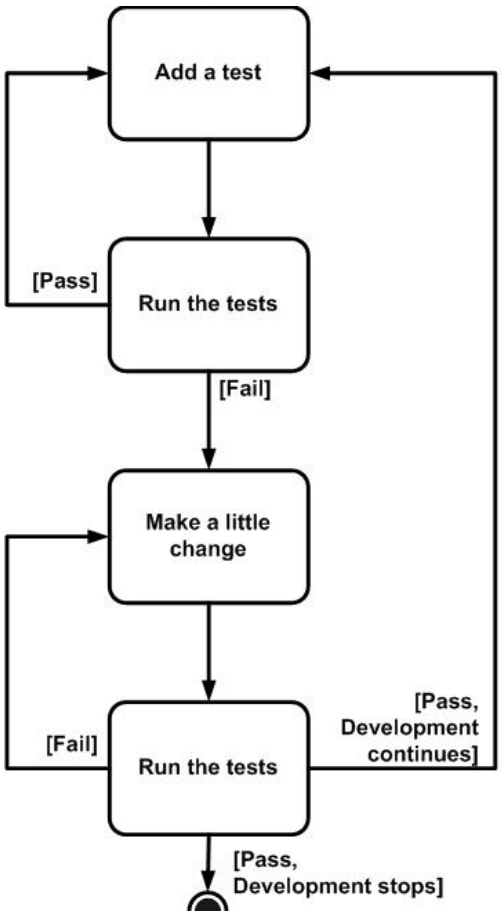
2.2.4 TDD – Test-driven development

Vývoj řízený testováním je myšlenka, která je zaměřená výhradně na vývoj na základě testů. Testování tedy nebere jako součást procesu, ale jako jeho základ (viz Obr. 5).

Podle metodiky TDD se na úvod procesu neřeší specifikace a design ale je nutné definovat kvalitní sadu testů. Na základě těchto testů pak vývojový tým tvoří program k jejich provedení s požadovaným výsledkem. Vyvíjeny však jsou pouze funkce, nezbytně nutné k provedení definovaných testů. Kvalita testovací sady je u této metodiky naprosto zásadní pro dosažení požadované funkčnosti produktu. (10)

Vývoj podle TDD probíhá v následujících krocích (viz. Obr. 5):

- definování testovací sady pro novou funkci
- spuštění testovací sady na současném systému (zjištění nedostatků)
- úprava systému
- znovu spuštění testovací sady (v případě neúspěchu, návrat o krok zpět)



Obr. 5 Proces Test-driven development, zdroj: (10)

2.2.5 SCRUM development process

Scrum je jednou z nejpoužívanějších agilních metodik současnosti. Jako i další agilní metodiky funguje na iterativním a inkrementálním principu.

Celý proces metody Scrum začíná definováním požadavků – vytvořením **Product backlogu**[3]. Jednotlivé cykly jsou nazývány **sprinty** a začínají tím, že **Product owner**[4] rozhodne, co chce vyvinout v následujících třiceti dnech. Vývojový tým během sprintu pracuje na dohodnutých úkolech a na konci tohoto období předvede Product ownerovi, co vytvořili. Na základě této demonstrace se Product owner rozhodne, na čem by měl vývojový tým pracovat v dalším sprintu. Tohoto procesu se účastní i **Scrum master**[5], který plní roli moderátora a zajišťuje plánování sprintů. (11)

Vzhledem k tomu, že je tato práce zaměřena právě na metodiku Scrum, bude tato metoda bude podrobněji rozvedena v kapitole (2.4).

2.2.6 ASD – Adaptive software development

Adaptivní vývoj softwaru je založen na opakujících se procesech *spekulace, spolupráce* a *učení*, které mají přednost před plánováním, návrhem a realizací. Vývojový tým se má rozhodnout na základě vlastních zkušeností a projektový manažer nemá mít obavu ze ztráty kontroly nad projektem. Hlavní roli zde hraje lidský faktor a důvěra ve schopnosti vývojového týmu, s minimalizací organizace a dohledu. (12)

2.2.7 Metodiky Crystal clear

Crystal clear, je podskupina agilních metodik, které vycházejí z názoru, že sebelepší metodika nemůže být vhodná pro všechny druhy projektů. Prvním krokem projektu by tedy mělo být upravení metodiky, podle vhodnosti pro konkrétní projekt. Všechny tyto metodiky však vycházejí ze společného základu, kterým je časté dodávání výsledků a komunikace se zákazníkem. Metodika se pak upravuje v závislosti na velikosti projektu a vývojového týmu. (13)

[3] **Product backog** je v podstatě *koš všech požadavků produktu.*
[4] **Product owner**, neboli *zákazník*.
[5] **Scrum master**, neboli *správce scrumu*.

2.2.8 Lean development

Lean development není exaktní metodika, ale souhrn principů, které vedou k efektivnějšímu a rychlejšímu vývoji a minimalizaci plýtvání. Tato metoda pochází z Japonského systému Lean manufacturing, zaměřeného na výrobní systémy. Jde o absolutní odstranění všeho, co není nezbytně nutné k dokončení procesu, nebo co by mohlo snížit jeho efektivitu a zvýšit náklady. Zakladatelé tvrdí, že je v průměrných podnicích možné vyvinout software za třetinu času, s využitím třetiny nákladů a s třetinovou chybovostí. (14)

Metodika Lean development je postavena na následujících principech:

- odstranit vše, co není nutné (děláme jen to, co tvoří hodnotu pro zákazníka)
- tvořit kvalitu (je efektivnější chyby nedělat, než je opravovat)
- učení se (důležitá je zejména zpětná vazba)
- vývoj tažený poptávkou (rozhodnutí dělat tak pozdě, jak jen to je možné)
- dodáváme produkt tak rychle, jak je to jen možné
- tvorba motivujícího prostředí, organizační struktury a nástrojů
- neustálé zlepšování (nespoléhat na ověřené praktiky, hledat zlepšení v každém procesu) (14)

2.3 Srovnání agilních a tradičních metodik

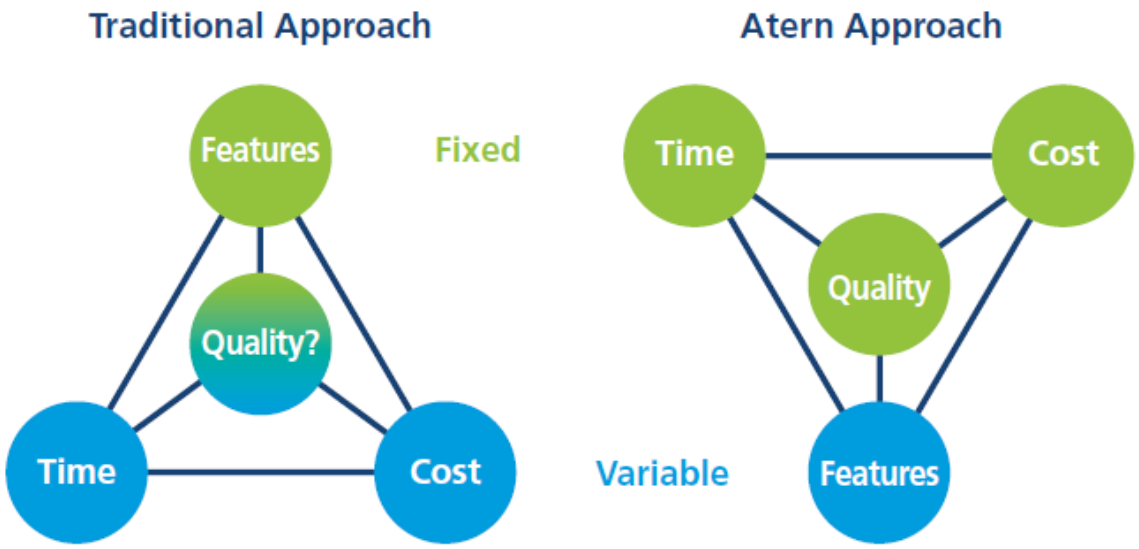
Agilní metodiky se od tradičních liší v základním pohledu na systém řízení projektů. Hlavním principem agilních metodik je pružnost, rychlost a flexibilita. Agilní řízení si uvědomuje, že kompletní a finální zadání by nemělo vznikat na začátku projektu (v čase největší nejistoty), ale mělo by být dolaďované v průběhu projektu, spolu s novými poznatky. Úzká spolupráce se zákazníkem zde také posiluje jeho důvěru a informovanost o průběhu.

Tradiční těžké metodiky jsou oproti tomu zaměřeny na kompletní specifikaci požadavků a jejich schválení už v úvodu projektu, detailní plánování a dokumentaci. Změny požadavků v průběhu projektu jsou komplikované a často vedou k přepracování celého projektu. Zákazník není zapojen do průběhu projektu a nemá tak informaci o

aktuálním stavu vyvíjeného produktu. Je tedy po dobu celého vývoje v nejistotě, že výsledný produkt nebude odpovídat jeho představám.

Může také nastat situace, kdy je projekt z nějakého důvodu **neočekávaně ukončen** (nedostatek finančních prostředků, produkt již není potřeba, atd.). V tradičním případě dostane zákazník rozpracovaný produkt, ale žádná z jeho částí nemusí být funkční, nebo je funkční část, která pro zákazníka není důležitá. V agilním pojetí si zákazník v každé vývojové iteraci určuje, které části produktu jsou pro něj nejpřínosnější, a sám rozhoduje o tom, co bude v daných iteračních obdobích vytvářeno. Při ukončení projektu v průběhu tak přesně ví, které části jsou už hotové a v jakém stavu je vývoj celého produktu.

Následující obrázek Obr. 6 znázorňuje **fixní** a **variabilní** položky, v podání tradičního a agilního vývoje. U tradičních metodik jsou fixní vlastnosti produktu (předem dané požadavky) a proměnlivé jsou náklady a doba vývoje. Tradiční metodiky naopak fixně dodržují dobu vývoje a náklady (počet iterací) a dávají volnost v požadavcích na vlastnosti produktu (možné změny požadavků v průběhu). Kvalita je u tradičních metodik neznámá, kvůli nejistotě zákazníka o průběhu vývoje a finální podobě produktu.



Obr. 6 Srovnání tradičního a agilního pojetí vývoje, zdroj: (15)

V následující tabulce Tab. 1 je uvedeno přehledné srovnání významných oblastí obou směrů.

Tab. 1 Srovnání tradičních a agilních metod řízení, zdroj: (16)

Oblast	**Tradiční směr**	**Agilní směr**
Proces vývoje	Podrobně definovaný, opakovatelný proces.	Empirický proces, vyžadující adaptaci na konkrétní situaci.
Životní cyklus vývoje	Vodopádový životní cyklus, nebo iterativní s dlouhými cykly.	Inkrementální (přírůstkový) vývoj s velmi krátkými iteracemi – cykly.
Role zákazníka	Zákazník nemá možnost zasahovat do průběhu projektu. Po schválení požadavků zákazník dostane až finální produkt.	Spolupráce se zákazníkem v průběhu projektu. Zákazník se aktivně zapojuje, má možnost měnit požadavky a určovat priority na začátku každé iterace.
Kvalita	Větší důraz na kvalitu procesu projektu, než na kvalitu výsledného produktu pro zákazníka.	Zaměření na priority a užitnou hodnotu produktu pro zákazníka.
Změny požadavků	Snaha o minimalizaci změn (požadavky jsou definovány a plánovány na začátku procesu). Změny jsou možné pouze prostřednictvím „řízení změn“.	Změny jsou přijímány (přírůstkové shromažďování požadavků v průběhu procesu, plánování v iteracích), požadavky jsou přehodnocovány.
Rozsah činností procesu	Podrobně popsaný proces, obsahující všechny kroky.	Eliminace nepotřebných činností, snaha o minimalizaci. Zaměření na jednoduchost procesu
Lidský faktor	Všichni jedinci jsou nahraditelní. Zaměření na procesy a rozsáhlou dokumentaci.	Lidé jsou klíčový faktor úspěchu projektu. Využití schopností jedinců, důraz na znalosti a kvalifikaci.
Týmové know-how	Specializace jednotlivých členů týmu na konkrétní roli. Šíření informací formou písemné dokumentace.	Kolektivní řešení problémů, spolupráce a aktivní sdílení informací.

2.4 SCRUM development process

Scrum development proces (dále jen Scum) je **iterativní**[6] a **inkrementální**[7] agilní metodikou, zaměřenou na řízení vývoje software. Podle mnoha autorů je Scrum pro svůj úspěch nejčastěji využívanou metodikou současnosti. Scrum není jen souhrn principů, ale také souhrn konkrétních pravidel, postupů a rolí jednotlivých účastníků, pomocí kterých je možné efektivně řídit vývoj softwaru.

Kenneth a Rubin vysvětlují, že rámec metodiky Scrum se skládá z *rolí* (sjednocených ve Scrum tým), *aktivit a artefaktů*. (7 str. 14)

2.4.1 Scrum tým a jednotlivé role

Podle zakladatelů metodiky Scrum, jsou role rozděleny do dvou skupin, pigs (prasata) a chickens (kuřata). Rozdíl mezi prasetem a kuřetem na projektu, do kterého vkládá prase maso a kuře vejce, je v tom, že prase je zapojeno přímo, zatímco kuřete nepřímo, problém se ho pouze týká. (17)

Mezi „pigs“ patří tedy role, které jsou aktivně a přímo zapojeny v procesu Scrumu, čímž tvoří tzv. **Scrum tým**. Patří mezi ně tyto:

- **Product Owner** je osoba, která zodpovídá za nastavené priority a za to, co se bude v příštím sprintu implementovat, aby byla maximalizována hodnota pro zákazníka. V praxi je do této role dosazován přímo zákazník, nebo jeho zástupce.
- **Scrum Master** má na starosti řízení procesu Scrum a jeho neustálou optimalizaci. Jde o roli moderátora a prostředníka, který má za úkol oddělit programátory od zákazníka (vede jednání o prioritách a plánu vývoje pro následující sprint). Role Scrum mastera může být přidělena buď analytikovi, manažerovi, nebo v ideálním případě Projektovému manažerovi. Neměl by ji však zastávat některý z programátorů.
- **Vývojový tým** je skupina programátorů a analytiků, kteří jsou přiřazeni do Scrum týmu a budou se společně zabývat vývojem produktu.

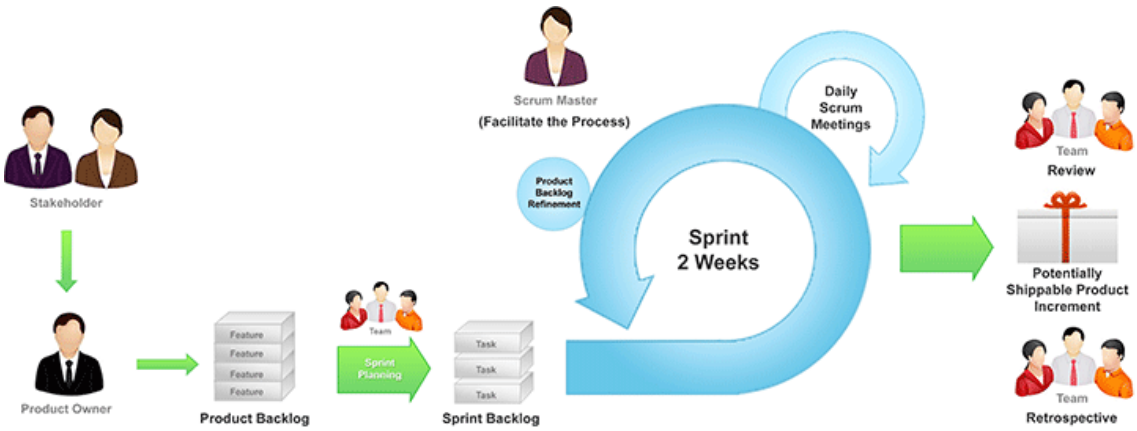
[6] **Iterativní** – opakování určitého procesu v cyklech (iteracích)
[7] **Inkrementální** – dodávání produktu průběžně po dílčích funkčních částech (inkrementech)

Rolí „chickens“ se Scrum proces pouze týká, ale nemusí v něm být zapojeny přímo. Mohou z něj například jen využívat informace o plánování, nebo přispívat svými názory. Jsou to role:

- **Stakeholdeři** jsou lidé na straně zákazníka, které výsledný produkt ovlivní, ale nejsou zodpovědní za jeho vývoj. Mohou to být například uživatelé, testeři, apod. Stakeholdeři mohou dávat Product ownerovi své názory a připomínky.
- **Manažeři** nejsou přímo zapojeni do vývoje produktu, proto jsou zařazeni do skupiny chickens. Z pohledu Scrumu jsou zodpovědní za vytvoření prostředí, ve kterém bude možné efektivně pracovat. (7 str. 14)

2.4.2 Aktivity a artefakty

Následující diagram znázorňuje celý průběh procesu Scrum, včetně artefaktů a aktivit, a jak na sebe postupně navazují.



Obr. 7 Proces vývoje podle metodiky Scrum, zdroj: (15)

Celý proces začíná u Product ownera, který má na základě požadavků od Stakeholderů vizi požadovaného produktu. Protože může být požadovaný produkt objemný, prvním krokem je příprava produktového backlogu (anglicky „grooming“). V tomto kroku je představa produktu rozložena do dílčích požadovaných funkcí, které jsou dle priorit seřazeny do produktového backlogu (koš všech nedokončených požadavků pro Scrum tým).

Požadavky z backlogu, které měly přiřazenu nejvyšší prioritu, jsou následně zařazeny do nejbližšího vývojového cyklu („sprintu“). Finální podoba plánu nadcházejícího sprintu („sprint backlog“) je diskutovaná a schválená Scrum masterem (za stranu vývojářů) a Product ownerem (za stranu zákazníka) během jednání, které se nazývá „sprint planing“.

Po schválení sprint backlogu může být odstartován nadcházející sprint (vývojové období). Toto období může trvat jeden až čtyři týdny. V rámci Sprintu se vývojový tým a Scrum master setkávají každý den ráno na schůzce, zvané „Denní scrum“ (Daily Scrum Meeting), kde je diskutován aktuální stav rozpracované práce, možné překážky a problémy a plán práce na jeden den.

Po uplynutí jednoho Sprintu je Produkt ownerovi doručen funkční inkrement produktu, ke které se vývojový tým zavázal při plánování sprintu. Závěr sprintu je také spojen se dvěma jednáními (hodnocení sprintu a retrospektiva) na kterých se diskutuje úspěšnost dokončeného sprintu.

Následuje návrat k plánování dalšího sprintu a sprintového backlogu a celý proces se opakuje, dokud není úspěšně vyvinut kompletní produkt, nebo jeho funkční část, z produktového backlogu. Takový funkční celek pak může zákazník uvést do produkce, prostřednictvím release. (7 str. 17)

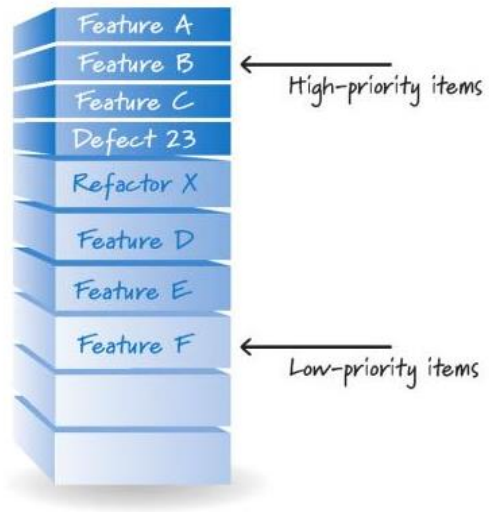
2.4.2.1 Produktový backlog

Jedním z hlavních pravidel Scrumu je, že nejdříve se dělají ty části, které mají největší přidanou hodnotu. Úkolem Produkt ownera je, na základě informací od Stakeholderů a Scrum týmu, vytvořit jednotlivé požadavky v produktovém backlogu, udržovat je aktuální a seřazené podle jeho priorit (viz. Obr. 8). Aby byl Product owner shopný seřadit jednotlivé části podle priorit, potřebuje znát jejich cenu. Tuto cenu předběžně odhadují analytici a development tým, v dohodnutých měrných jednotkách, například points[8], mandays[9], manhours[10] apod. Na základě tohoto odhadu může Product owner srovnat přidanou hodnotu a náklady každé části a seřadit je podle priorit.

[8] Points – body mohou být použity jako jakási virtuální měrná jednotka ceny dílčího úkolu.
[9] Mandays – měrná jednotka, která znamená jeden den lidské práce jednoho vývojáře.
[10] Manhours – jedna hodina práce vývojáře.

U nových produktů obsahuje produktový backlog pouze nové části, kdežto u probíhajícího vývoje produktu může backlog obsahovat také jiné úkoly, jako například změny již hotových funkcí, opravy defektů apod.

Produktový backlog se v průběhu celého vývoje neustále upravuje, doplňuje a mění. Tato aktivita se anglicky nazývá „Product backlog grooming“. (7 str. 19)



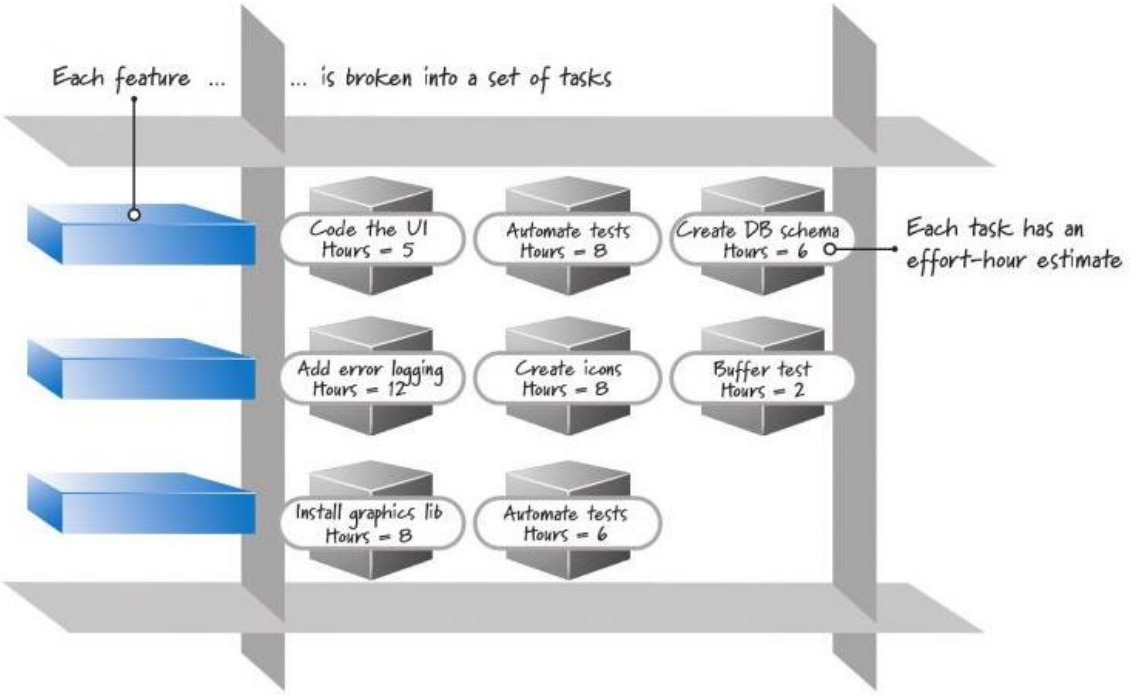
Obr. 8 Produktový backlog, Zdroj: (7)

2.4.2.2 Sprint planning a Sprint backlog

Ve Scrumu je práce vývojářů prováděna v iteračních cyklech, které se nazývají Sprinty. Délka tohoto cyklu může být od jednoho týdne, až po 1 měsíc. Jde o fixní období, které má přesně stanovené datum začátku a konce. Všechny sprinty by měly být stejně dlouhé, s tím že nový sprint navazuje na konec sprintu předešlého. Výsledkem každého sprintu by měla být potenciálně funkční část produktu, která je pro zákazníka použitelná. (7 str. 20)

Každému sprintu těsně předchází Sprint planning. Během tohoto sprint planningu se nejvýše postavené části z produktového backlogu přesunou do sprint backlogu. Tvoří se tedy plán částí, na kterých bude vývojový tým pracovat v následujícím sprintu a vývojový tým se tím zavazuje, že na konci sprintu dodá funkční inkrement produktu, složený z naplánovaných částí. Sprint planning probíhá za účasti Scrum týmu: Product owner, Scrum master a vývojový tým.

Pro přehlednost a objektivní odhad nákladů jednotlivých částí, je každá tato část rozložena do několika dílčích úkolů. U každého z těchto dílčích úkolů je pak proveden odhad pracnosti. Součet odhadů všech dílčích úkolů jedné části je pak celkový odhad dané části, viz Obr. 9. (7 str. 22)



Obr. 9 Rozklad částí produktu na jednotlivé úkoly, Zdroj: (7)

2.4.2.3 Provedení sprintu a denní Scrum

Jakmile se Scrum tým během plánování sprintu dohodne na úkolech pro nadcházející sprint (vytvoření a odsouhlasení sprint backlogu), může být tento sprint odstartován. Vývojový tým pak pod vedením Scum mastera vykonává všechny jednotlivé úkoly z backlogu sprintu. Členové vývojového týmu si sami určují pořadí, ve kterém budou vykonávat jednotlivé úkoly, na základě vlastních zkušeností, za účelem nejlepšího dosažení cíle sprintu.

Každý den aktuálního sprintu, ideálně ráno a ve stejnou dobu, se členové vývojového týmu setkají na 10 až 15 minut. Účelem tohoto denního Scrumu, je diskuze o vývoji sprintu a aktuálních problémech při vývoji. Všichni členové vývojového týmu tak mají přehled o aktuálním vývoji sprintu a problémech a mohou tak přispět k úspěšnému dokončení sprintu.

Běžný postup v denním sprintu je zodpovězení následujících tří otázek každým členem vývojového týmu (7 str. 24):

- Co jsem dokončil od posledního denního Scrumu?
- Na čem budu pracovat do příštího denního Scrumu?
- Jaké překážky mi brání k dokončení mých úkolů?

Během této diskuze upravuje Scrum master záznamy, o aktuálním vývoji úkolů ve sprintu (označuje hotové a rozpracované úkoly).

2.4.2.4 Výsledek sprintu

Ve Scrumu se za výsledek sprintu označuje „potenciálně doručitelný inkrement produktu“, který obsahuje všechny části, ke kterým se zavázal při plánování daného sprintu. Důležité je brát v úvahu dohodnutou definici toho, co bude považováno za hotové. V běžné praxi je za hotovou považovaná část produktu, která byla designovaná, vyvinutá, integrovaná, testovaná a dokumentovaná. Všechny tyto úkoly by měly být ve standardním případě hotové, abychom mohli označit část produktu za dokončenou. Skutečné nasazení tohoto inkrementu produktu je pak rozhodnutím Produkt ownera. (7 str. 26)

2.4.2.5 Přezkoumání a retrospektiva sprintu

Po ukončení sprintu následují dvě aktivity, za účelem kontroly a úpravy vyvíjeného produktu a organizace Scrumu.

Hodnocení sprintu („sprint review“) se účastní nejen Scrum tým, ale také stakeholdeři, sponzoři, zákazníci a ostatní vývojové týmy. Účelem je zhodnocení výsledku vynaložené práce vývojového týmu a zpětná vazba od všech zainteresovaných stran. Hodnocení sprintu se tedy zabývá hodnocením inkrementu produktu.

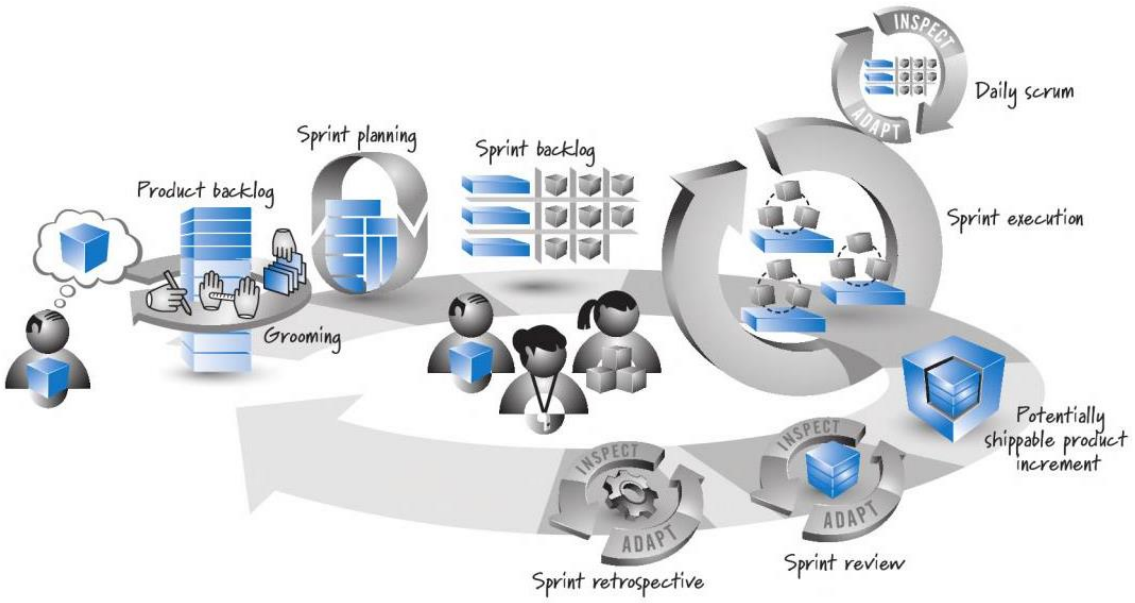
Retrospektiva[11] sprintu je příležitostí ke kontrole a úpravě procesu vývoje. Diskuze se účastní pouze Scrum tým a řeší přidanou hodnotu jednotlivých činností a praktické nastavení procesu metodiky Scrum. (7 str. 27)

[11] **Retrospektiva** – pohled zpět, do minulosti

2.4.2.6 Konec sprintu

Po doručení výsledku sprintu a hodnocení a retrospektivě sprintu, může být aktuální sprint oficiálně uzavřen.

Celý tento cyklus sprintů se opakuje, počínaje plánováním sprintu, dokud je potřeba vyvíjet, nebo upravovat další části produktu.



Obr. 10 Cyklus metodiky Scrum, Zdroj: (7 str. 14)

3 ANALÝZA SOUČASNÉHO STAVU

V této kapitole jsou prakticky využita teoretická východiska zmíněna v předchozí části práce. Pro analýzu aplikace agilní metodiky Scrum (jak je uvedeno již v názvu práce) byla vybrána nadnárodní komerční společnost, působící v oblasti bankovnictví. Konkrétně jde vybraný IT tým společnosti KBC Group NV, kterou v české republice známe jako mateřskou společnost Československé obchodní banky, a.s.. Společnost KBC a vybraný vývojový tým jsou blíže představeny v následující podkapitole 3.1.

Abychom mohli aplikovat metodiku Scrum v konkrétní firmě, je nutné nejprve analyzovat současnou situaci, včetně způsobu plánování a organizace, který je v současnosti využíván a také potřeby a požadavky na využívaný systém. Analýza současné situace a potřeb ve vybraném vývojového týmu banky KBC je zpracována v kapitole **Chyba! Nenalezen zdroj odkazů.**.

Abychom objektivně zvolili vhodný nástroj, k organizaci metodikou Scrum, je uveden přehled nejpoužívanějších dostupných nástrojů. Každý z nástrojů má některé výhody i nevýhody, ať už jde o funkce, přehlednost, nebo cenu. Přehled dostupných podpůrných nástrojů pro Scrum je uveden v kapitole 3.7.

V kapitole 4 jsou zpracovány doporučení a návrhy na změny, k efektivnímu využívání metodiky Scrum. U každého zmíněného návrhu je uveden i očekávaný přínos, ze zavedení této změny. Tato část je hlavním přínosem této studie a může posloužit firmám buď k zavedení metodiky Scrum, nebo ke zlepšení dosavadního způsobu organizace Scrumu.

3.1 Představení zkoumané společnosti

Společnost KBC Bank NV je součástí nadnárodní skupiny KBC Group NV, která je jejím jediným vlastníkem. KBC Bank i skupina KBC Group sídlí v Bruselu v Belgii, na adrese Havenlaan 2.

Geograficky skupina působí především na svých domácích trzích v Belgii, v České republice, na Slovensku, v Bulharsku a Maďarsku, přičemž působí také v Irsku a v omezené míře také v několika dalších zemích světa (KBC New York, KBC Shanghai,

KBC Singapore, KBC London, KBC France a další). Na konci roku 2014 skupina KBC na svých pěti domácích trzích a v Irsku obsluhovala zhruba 10 milionů klientů a zaměstnávala cca 36 tisíc zaměstnanců, z toho asi polovinu v zemích střední a východní Evropy. V následující tabulce Tab. 2 jsou znázorněny klíčové finanční ukazatele skupiny KBC. (18)

Tab. 2 Klíčové finanční ukazatele skupiny KBC, zdroj: (18)

Ukazatel		**31. 12. 2013**	**31. 12. 2014**
Celková aktiva	mld. EUR	238,7	245,2
Klientské úvěry a pohledávky	mld. EUR	120,4	124,6
Klientská depozita a dluhové cenné papíry	mld. EUR	161,1	161,8
Čistý zisk	mil. EUR	1 015,0	1 762,0
Ukazatel vlastního kapitálu	%	12,8	14,3
Poměr nákladů k výnosům	%	52,0	57,0

S platností od 1. ledna 2013 uspořádala KBC Group své aktivity na klíčových trzích do tří obchodních divizí – Belgie, Česká republika a Mezinárodní trhy. Divize Česká republika zahrnuje všechny obchodní aktivity KBC Group v České republice. Do aktivit v rámci České Republiky patří i centrum sdílených služeb[12], které bylo založeno roku 2011 v Brně. Do tohoto centra sdílených služeb byly postupně přesunuty některé back-office[13] týmy, zaměřené zejména na výkony sdílené pro více společností skupiny KBC Group. Účelem vytvoření tohoto centra bylo nejen snížení nákladů, ale také centralizace těchto sdílených back-office výkonů. Brněnské SSC momentálně provozuje následující činnosti:

- Platby (payments processing, claims and investigations, payments proces management, payments application management)
- Finanční trhy (back-office, middle-office, proces management, applicatin management)

[12] **Centrum sdílených služeb** – anglicky „shared service center“, neboli **SSC.**
[13] **Back-office** – část firmy, která zabezpečuje podpůrné administrativní funkce.

- ICT (test team, catalogue request processing)
- Účetnictví (accounts payable, accounts receivable, procurement)
- Finance (financial controlling)

Většina vývojových IT týmů KBC Group je momentálně centralizována přímo v Belgii. Pro některé vývojové týmy pracují i IT zaměstnanci z České republiky, Maďarska nebo Indie.

3.2 Popis zkoumaného vývojového týmu

Vzhledem k tomu, že KBC Group zaměstnává více jak 1000 IT pracovníků v několika zemích světa, nejsou zavedené postupy ani využívané nástroje jednotlivých týmů sjednoceny. Proto bude analýza zaměřena na konkrétní IT tým v rámci KBC Group.

Zkoumaný vývojový tým je zaměřený na vývoj platebních aplikací, pro zahraniční platby. Konkrétně jde o tyto:

- Retailový platební engine[14]
- Profesionální platební engine
- Embargo kontroly
- Webové front-end aplikace[15]
- Webová aplikace pro zpracovávání stížností
- Infrastruktura

Tento tým je složen z 30 členů, rozdělených na 2 části, Platební enginy a Platební infrastruktura. Tým **Enginy** se skládá ze 14 členů, sídlících v Bruselu v Belgii. Jeden dodatečný člen pracuje pro tento tým přímo z Indie. Tento tým má na starosti správu a vývoj retailového a profesionálního platebního enginu. Tým **Infrastruktura** se skládá také ze 14 členů, kteří však sídlí v Belgickém městě Mechelen. Také mají jednoho dodatečného člena v Indii. Tento tým má na starosti správu a vývoj infrastruktury a webových aplikací (embargo kontroly, front-end a aplikace pro zpracovávání stížností).

[14] **Engine** – jádro počítačového programu (z anglického „motor“)
[15] **Front-end aplikace** – aplikace dostupná uživatelům/zákazníkům, většinou jde o vstupní aplikace.

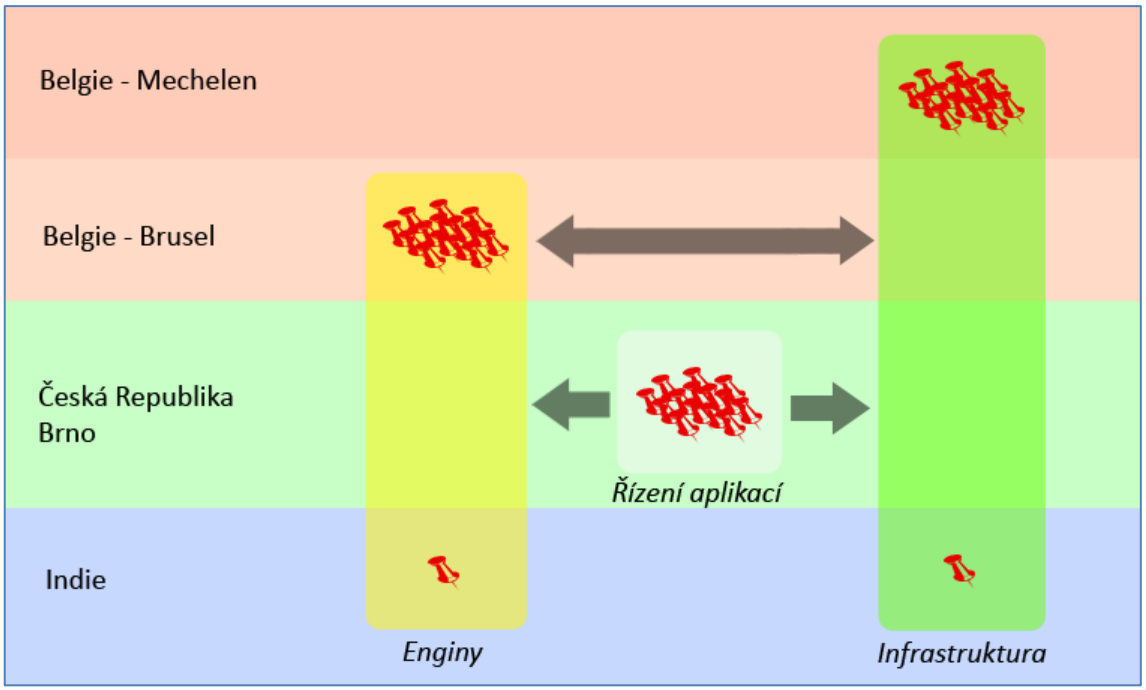
3.3 Struktura a geografické rozložení vývojového týmu

Jak již bylo uvedeno ve stručném popisu, celý vývojový tým je rozdělen na dvě části: **Enginy** a **Infrastruktura**. Každý z těchto menších týmu se skládá z patnácti členů, včetně manažera a včetně jednoho člena, pracujícího externě z Indie. Dohromady jde tedy o 28 členů, kteří fyzicky pracují na vývoji platebních aplikací a 2 manažeři. Část těchto dvou týmů, která pracuje v Belgii, nepřichází fyzicky do styku, jelikož tým Enginy pracuje v Belgickém městě Brusel a tým Infrastruktura v Belgickém městě Mechelen.

Řízení aplikací, vyvíjených tímto IT týmem, je umístěno v České Republice, v centru sdílených služeb v Brně. Jde o dva výše zmíněné týmy, Payments application management a Payments process management. Úkolem těchto týmů je jednat s koncovými zákazníky a uživateli těchto aplikací a rozhodovat o jejich budoucím vývoji. Vývojový IT tým je zde dodavatel IT služeb a oddělení Řízení aplikací je pro IT tým v roli zákazníka. Zkoumaný IT tým tedy vyvíjí zmíněné aplikace právě na základě rozhodnutí Brněnského týmu, který vývoj aplikací řídí.

Vzhledem k mezinárodnímu geografickému rozložení, není možná přímá fyzická komunikace mezi jednotlivými týmy. Dokonce i komunikace mezi členy v rámci jednoho týmu je ztížená, v případě člena pracujícího z Indie. Veškerá komunikace proto probíhá v Anglickém jazyce a elektronickými kanály, viz kapitola 3.5.

Struktura a geografické rozložení vývojového týmu a týmu řízení aplikací je znázorněna následujícím obrázkem Obr. 11.



Obr. 11 Struktura a geografické rozložení vývojového týmu, zdroj: vlastní

3.4 Stakeholdeři

Pod pojmem stakeholdeři rozumíme všechny zainteresované strany, které jsou do vývoje jakýmkoliv způsobem zapojeny. Pokud se zaměříme na zkoumaný vývojový tým, můžeme mezi významné stakeholdery zařadit tyto:

- **Koncoví uživatelé a zákazníci** – jsou to všichni koncoví uživatelé, kteří využívají aplikace, spravované tímto týmem. Hlavním zájmem koncových uživatelů, je bezproblémový chod těchto aplikací. Samozřejmě také požadují, aby aplikace obsahovaly všechny potřebné funkce.
- **Tým řízení aplikací** – jde o oddělení, které je ve vztahu k vývojovému týmu v roli klienta. Tým řízení aplikací definuje požadavky na vývoj, definuje architekturu systémů, zabývá se business testováním dodaných úprav systému. Mimo tyto role také vystupuje jako kontaktní bod pro všechny koncové uživatele, poskytuje jim podporu, a v případě požadavků na úpravu systému tyto požadavky přezkoumá a po odsouhlasení zařadí do aktuálního seznamu požadavků, pro vývojový tým.

- **Management domény plateb** – působí v roli držitele rozpočtů. V našem případě jde hlavně o rozpočet na provoz a údržbu systémů a rozpočet na projekty, v rámci domény plateb. Díky tomu má management poslední slovo, co se týká rozhodování o vývoji aplikací.
- **Projektový management** – v případě zahájení projektu, který má dopad na aplikace plateb, je vývojový tým přímo kontaktován projektovým manažerem. Projektový manažer definuje požadované změny a jejich načasování, nezávisle na aktuálním plánování změn, ze strany týmu řízení aplikací.
- **Ostatní IT týmy** – zejména jde o vývojové týmy, se kterými je zkoumaný vývojový tým v interakci. Například:
    - IT tým, který má na starosti údržbu serverů, na kterých jsou aplikace spuštěné.
    - Vývojové týmy okolních aplikací, se kterými jsou platební aplikace jakýmkoliv způsobem v interakci. Změny platebních aplikací, nebo těchto okolních aplikací na sebe mohou mít dopad. V takovém případě je nutná vzájemná spolupráce.
- **Audit** – je jednou z neopomenutelných stran, která může ovlivnit vývoj těchto aplikací. Zejména v oblasti zabezpečení a plnění právních požadavků. Může jít o audit interní, nebo i externí.
- **Risk** – neovlivňuje vývoj aplikací přímo, ale obdobně jako audit zkoumá možné nedostatky systémů a vydává doporučení ke zlepšení.

3.5 Využívané softwarové nástroje

Jako interní komunikační nástroje v rámci společnosti, využívá KBC Group téměř všechny dostupné kanály. Mimo standardní, jako například email a telefon, se v poslední době stává čím dál využívanější messenger, který obsahuje mimo jiné i možnost online hovoru, nebo videohovoru. Tyto online hovory a videohovory jsou nejen nízkonákladové, ale jsou možné i ve více lidech najednou a každý člen může přitom sedět u vlastního počítače. Jde o velmi efektivní způsob organizace jednání,

v případech geografického rozložení jednotlivých členů, jako je tomu například právě u zkoumaného vývojového týmu a jejich product ownerů – Belgie Brusel, Belgie Mechelen, Indie a Česká Republika. Tyto videohovory jsou využívány právě k jednáním, mezi vývojovým týmem a product owners.

K plánování vývoje software jsou v rámci KBC Group momentálně využívány dva komerční nástroje: *Jira* (viz 3.7.3) a *Scrumwise* (viz 3.7.2). V minulosti byl využíván pouze komplexnější nástroj Jira. Na základě jednání o pořízení jiného nástroje byla pořízena hromadná licence i pro jednodušší a nástroj Scrumwise.

Spolu se zavedením nástroje Scrumwise do společnosti, začal využívat metodiku Scrum i zkoumaný vývojový tým a zvolil si jednodušší a přehlednější nástroj Scrumwise. V současnosti tedy zkoumaný tým využívá nástroj Scrumwise. V rámci KBC Group jsou stále dostupné oba nástroje, Scrumwise i Jira.

Pro evidenci změnových požadavků je využíván externě dodávaný softwarový nástroj *ServiceNow*. Tento nástroj umožňuje také k řízení incidentů[16]. K evidenci defektů[17] je využíván nástroj *QualityCenter* od společnosti HP.

V současnosti jsou tedy pro evidenci incidentů, defektů, změnových požadavků a pro řízení vývoje metodikou scrum, využívány 3 různé aplikace. To vede k nárůstu administrativní náročnosti a duplicity některých informací.

3.6 Současná aplikace metodiky Scrum

Výše popsaný vývojový tým platebních aplikací již metodiku Scrum k plánování vývoje využívá. Vlastním pozorováním a také zapojením se do procesu Scrumu tohoto týmu v roli Product ownera jsem zjistil, že aktuální stav nevyužívá veškeré možnosti, které tato metodika nabízí. Způsob, jakým je Scrum development process využíván v současné době, je popsán v následujících podkapitolách.

3.6.1 Scrum tým a jednotlivé role

Jednotlivé role Scrum týmu byly teoreticky vysvětleny v kapitole 2.4.1.

[16] **Incident** – v IT prostředí se za incident považuje problém softwaru na Produkčním (reálném) prostředí
[17] **Defekt** – testováním změn odhalený nedostatek, na Akceptačním (testovacím) prostředí

3.6.1.1 Vývojový tým

Vývojový tým je skupina třiceti IT zaměstnanců, kteří jsou rozděleni do dvou částí: *enginy* a *infrastruktura*. Každý z těchto sub-týmů má na starosti jinou část vyvíjených aplikací, má vlastního liniového manažera a vlastní plánování. Geograficky také sídlí v jiném Belgickém městě. Tato fakta vela k rozhodnutí, že z pohledu Scrumu vystupují jako dva oddělené týmy.

3.6.1.2 Scrum master

V roli Scrum mastera pro tyto dva týmy vystupují jejich manažeři. Manažer každého z týmů není programátor a nepodílí se přímo na vývoji. Z pohledu Scrumu má v současné době na starosti tyto úkoly:

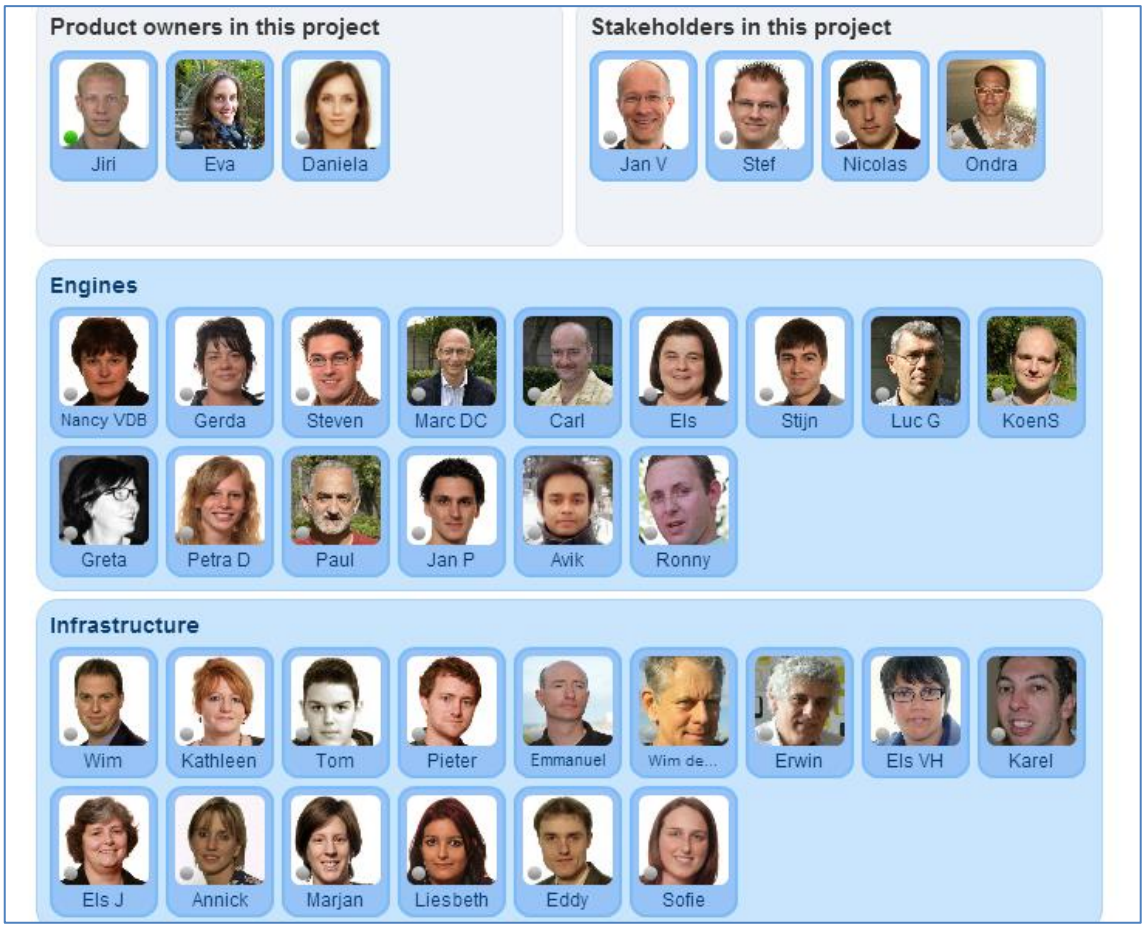
1. *příprava sprint backlogu*
2. *vedení plánování sprintu* (jednání s product owners o sestavení sprint backlogu)
3. *organizace a vedení denních Scrum setkání* (každodenní krátké setkání všech členů vývojového týmu)
4. *správa a aktualizace dat v nástroji Scrumwise* a další.

3.6.1.3 Product owner

Jako product owner vystupují tři členové Brněnského týmu řízení aplikací. Každý z těchto členů je product owner dvou (ze šesti) spravovaných aplikací (viz 3.2). Vzhledem k tomu, že každý z product ownerů spravuje aplikace, které se týkají obou vývojových týmů, účastní se všichni product owneři všech jednání dohromady, ať už s jedním, nebo druhým vývojovým týmem.

Mimo tří product ownerů, z týmu řízení aplikací, mohou v roli product ownera vystupovat i projektoví manažeři, jelikož mohou přidávat požadavky do produktového backlogu.

Z pohledu Scrumu je momentálně role product ownera nejasná, jelikož produktový backlog může doplňovat několik různých lidí, z několika různých týmů.



Obr. 12 Složení vývojového týmu a product ownerů, zdroj: vlastní

3.6.2 Scrum proces, aktivity a artefakty

Produktový backlog

Produktový backlog je v současné době veden ve dvou aplikacích. Každá z aplikací je však využívána pro jiné informace a k jinému účelu:

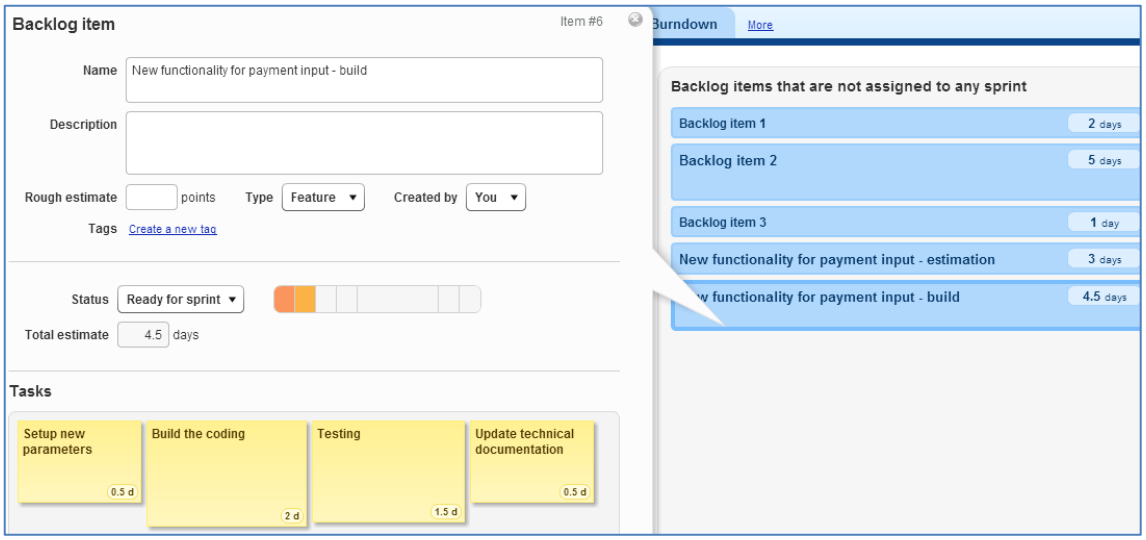
- ServiceNow: Databáze všech změnových požadavků kde každý požadavek prochází schvalováním a různými stádii. Jsou zde registrovány jak požadavky ze strany projektových manažerů, tak ze strany řízení aplikací (údržba a menší změny). Každá zaregistrovaná změna zde obsahuje detailní popis požadavku a po analýze ze strany IT také technické detaily o navrženém řešení a odhad ceny. Databáze změn v aplikací v ServiceNow je spravována business stranou (product owneři z týmu řízení aplikací). Priority položek v databázi požadavků jsou určeny číslem.

- Scrumwise: Pro potřebu plánování Scumu je backlog úkolů pro vývojový tým tvořen v aplikaci Scrumwise. Tento backlog vychází z databáze požadavků, vedených v aplikaci ServiceNow. Tento backlog tvoří a spravuje Scrum master na straně vývojového týmu. Priority v produktovém backlogu jsou určeny pořadím.

Backlog pro Scrum, obsahuje jmenovitý seznam položek. Jelikož pracuje vývojový tým na každé položce ve dvou fázích, je každá z položek uvedena dvakrát. Díky tomu je možné umístit každou z těchto fází do jiného Sprintu.

- 1. fáze: analýza řešení a odhad ceny (estimation)
- 2. fáze: provedení změny (build)

Každá z položek v produktovém backlogu Scrumu, je složena z několika dílčích úkolů, které je možné zobrazit po otevření každé backlog položky. Pro každý dílčí úkol je proveden odhad vývojovým týmem a tento odhad je k úkolům doplněn. Celkový součet odhadů dílčích úkolů backlog položky, je pak u této položky automaticky zobrazen. Každý dílčí úkol, nebo položka backlogu, jsou zobrazeny větší, či menší, v závislosti na doplněném odhadu ceny. Toto umožňuje používaný nástroj Scrumwise velmi pohodlně (viz Obr. 13).



Obr. 13 Produktový backlog a rozložení na dílčí úkoly, zdroj: vlastní

„Grooming“ produktového backlogu (doplňování, řazení podle priorit, updatování a promazávání) provádí product owneři v aplikaci ServiceNow. Na základě těchto změn

aktualizují Scrum masteři backlog Scrumu v aplikaci Scrumwise. Produkt owneři tedy v aplikaci Scrumwise neprování žádné změny, používají ji pouze k nahlížení.

Aktuální produktový backlog, pro vývojové týmy enginy a infrastruktura, obsahuje přes 200 jednotlivých požadavků. Rychlost, s jakou jsou tyto požadavky implementovány, je menší, než přísun nových požadavků. Tento backlog proto pomalým tempem neustále narůstá.

3.6.2.1 Plánování sprintu a backlog sprintu

Smyslem jednání, které se nazývá „plánování sprintu“, je odsouhlasení obsahu sprintového backlogu. Pokud je tento backlog následujícího sprintu odsouhlasen celým Scrum týmem (vývojový tým, Scrum master a product owner), vývojový tým se zavazuje dokončit všechny naplánované položky, během budoucího sprintu.

Tomuto jednání předchází příprava, kterou provádí v současné době Scrum master. Ten má na starosti jednak doplňování kompletního produktového backlogu v aplikaci Scrumwise (na základě požadavků zaregistrovaných v aplikaci ServiceNow), ale také přípravu sprintového backlogu.

Příprava sprintového backlogu probíhá v následujících krocích:

1. Nejprve je určena dostupná kapacita vývojového týmu v budoucím sprintu.
2. Druhým krokem je doplnění ostatních nezbytných aktivit do backlogu budoucího sprintu. Patří mezi ně například:
    - Plánovaná nedostupnost (dovolená)
    - Rezervace pro neplánovanou nedostupnost (nemoc)
    - Školení
    - Poskytování technické podpory
    - Schůze a jednání
3. Následuje doplnění položek s nejvyšší prioritou z produktového backlogu, do backlogu pro budoucí sprint tak, aby odpovídala kapacita dostupná a naplánovaná.

Setkání s názvem „plánování sprintu“ se koná zpravidla poslední den aktuálního sprintu (den před začátkem plánovaného sprintu). Naplánovaný backlog sprintu prezentuje Scrum master product ownerům, za účasti celého Scrum týmu – plánování sprintu. Product owneři mohou mít připomínky k naplánovanému backlogu sprintu a mohou požadovat úpravy. Výsledkem je odsouhlasený backlog příštího sprintu, kde jsou všechny úkoly rovnoměrně rozděleny mezi jednotlivé členy vývojového týmu.

Pokud je backlog nového sprintu odsouhlasen, Scrum master společně s vývojovým týmem rozdělí všechny jednotlivé úkoly, mezi členy vývojového týmu. Úkoly se rozdělují zpravidla na základě znalostí členů týmu. Všechny položky a úkoly do nového sprintu jsou tedy ještě před začátkem sprintu přiděleny jednotlivým členům vývojového týmu.

Plánování sprintu probíhá pro každý z vývojových týmů zvlášť (enginy a infrastruktura). Jedna tato schůzka trvá přibližně 2 hodiny, dohromady tedy 4 hodiny za měsíc.

Obr. 14 Plánování sprintu, zdroj: vlastní

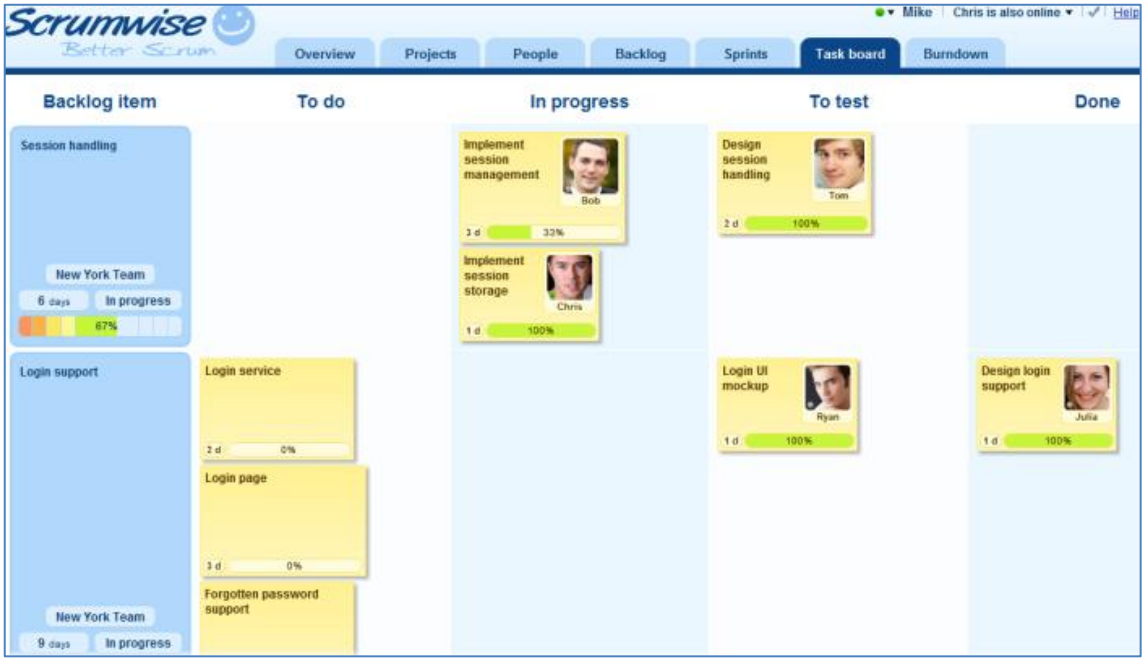
3.6.2.2 Sprint

Délka sprintu je momentálně nastavena na 30 dní (jeden měsíc), což odpovídá i doporučené délce sprintu ve většině literatury. Sprint začíná zpravidla první den v měsíci, což je také den po plánování sprintu. Konec sprintu je poslední den v měsíci.

3.6.2.3 Denní sprint setkání

Každý den ráno se koná „denní sprint setkání“. Toto denní setkání trvá přibližně 30 minut pro každý z vývojových týmů. Délka tohoto setkání je dána zejména vysokým počtem členů v týmu. Organizátorem těchto setkání je Scrum master, který svolává všechny členy svého vývojového týmu. Těchto setkání se neúčastní product owneři.

Účelem tohoto denního setkání je společně aktualizovat stav úkolů. Scrum master se postupně ptá jednotlivých členů na stav jejich úkolů a aktualizuje údaje vedené v aplikaci Scrumwise. Toto setkání tedy slouží ke sdílení základních informací o průběhu sprintu a k aktualizaci těchto údajů ve Scrumwise. K tomu má Scrum master v aplikaci Scrumwise k dispozici takzvaný „task board“, kde může jednoduše aktualizovat stav úkolů jednoduchým přetažením. Jde o elektronickou podobu tabule, nebo nástěnky, kde jsou znázorněny jednotlivé úkoly, jako nálepky. Viz Obr. 15.



Obr. 15 Scrum task board, zdroj: vlastní

3.6.2.4 Konec sprintu

Sprint končí posledním dnem v měsíci, kdy zároveň Scrum master připravuje backlog nového sprintu. Úkoly, které případně nebyly v současném sprintu dokončeny, jsou

posunuty do sprintu budoucího. Sprint je oficiálně ukončen následující den ráno, kdy se odstartuje sprint nový.

Po ukončení sprintu se v současné době nekonají žádné hodnotící setkání – přezkoumání, ani retrospektiva.

3.6.3 Ukazatele burnup a burndown

Scrum master a product owneři v současné době využívají pouze ukazatel „burndown[18]“. Jde o graf, který znázorňuje množství práce, které zbývá ještě vykonat do konce sprintu. Tento ukazatel slouží zejména pro Scrum mastera, který prostřednictvím něho sleduje průběh aktuálního sprintu a z grafu může vyvodit, jestli vývojový tým zvládá plnit naplánované úkoly. Viz Obr. 16.

Obr. 16 Ukazatel burndown, zdroj: vlastní

[18] **Burndown** – grafické znázornění zbývajícího množství úkolů, v průběhu sprintu.

Opak tohoto ukazatele je graf zvaný „burnup[19]“, který znázorňuje průběh již vykonané práce, v návaznosti na sprint, nebo technický release. Tento ukazatel momentálně využíván není.

3.6.4 Iterativní a inkrementální přístup

Iterativní přístup je u zkoumaného vývojového týmu uplatňován. Úkoly jsou plánovány a prováděny v měsíčních iteracích. Momentálně není využíván jiný způsob dlouhodobějšího plánování změnových požadavků. K plánování vývoje je využíván pouze Scrum.

Inkrementální přístup je zde však sporný. Období sprintu totiž neodpovídá období technických releasů[20]. To znamená, že na konci každého sprintu má sice vývojový tým dokončené úpravy systému, ale ty není možné vypustit na testovací prostředí, ani na produkční prostředí mimo technický release. Dodávání nových částí systému je z bezpečnostních důvodů možné pouze prostřednictvím releasů, které jsou předem naplánovány, přibližně na každé dva měsíce. Inkrementy produktu jsou tedy na konci každého sprintu dokončeny, ale jsou shromažďovány na straně IT, až do předem naplánovaného data release, kdy jsou znovu testovány a vypuštěny všechny naráz. O inkrementální přístup tedy jde – dodáváme produkt postupně v částech, ale tyto dodávky inkrementů neodpovídají spritům.

3.6.5 Změny v průběhu vývoje

Vzhledem k přeplněnému backlogu požadavků, je pro nové požadavky velmi dlouhá doba „time to market[21]“, někdy i více než jeden rok. Dlouhá doba doručení požadavků poměrně často způsobuje, že původní zadání požadavku už v době jeho implementace není aktuální. V těchto případech je nutné, aby product owner požadavky, které jsou plánovány do následujícího sprintu, znovu aktualizoval a prověřil jejich správnost a také potřebu. To se však v současné době děje jen zřídka a proto nastávají situace, kdy je naplánováno vykonání staršího požadavku bez kontroly, zda je tento požadavek stále

[19] **Burnup** – grafické znázornění hotového množství úkolů, v průběhu sprintu.
[20] **Release** – Termín, používaný v oblasti IT, k označení momentu vypuštění nové verze systému do produkčního prostředí.
[21] **Time to market** – doba od zadání požadavku až k jeho doručení

potřeba a zda jsou uvedené detaily aktuální. To může vést k situaci, kdy je v průběhu vývoje, testování, nebo dokonce až po nasazení na produkční prostředí zjištěno, že tento požadavek, definovaný například před rokem a půl, už neodpovídá současné situaci a je nutné jej opět změnit.

Změny v požadavcích, u kterých ještě nezačala fáze budování, jsou vývojovým týmem akceptovány a vývoj je přizpůsoben novým požadavkům. Toto je však ideální situace změny v požadavcích, kdy je dopad i navýšení nákladů minimální.

Změny v požadavku, který je již ve fázi budování, nebo testování, jsou stále přijímány a vývojový tým je ochotný vývoj přizpůsobit. V tomto případě je nutné přepracování části programu, která již byla vybudována podle nesprávných požadavků, což zvyšuje výsledné náklady na provedení změny. Přesto jsou však náklady výrazně nižší, než pokud by byly opravy prováděny až po vypuštění nežádoucí změny na produkční prostředí.

Výjimečně také nastává situace, kdy dojde ke změně priorit požadavků v průběhu sprintu. V tom případě je po dohodě akceptována i změna v aktuálně běžícím sprintu. Může jít například o nahrazení naplánovaného požadavku ve sprintu za nový, s vysokou prioritou. Příkladem je například vyřešení nově objeveného incidentu, či defektu, které je nutné vyřešit přednostně.

Z pohledu přijímání změn v průběhu projektu, nebo vývoje jakýchkoliv změn, je tedy u zkoumaného vývojového týmu viditelný silně agilní přístup. Změny jsou přijímány dokonce i v rámci běžícího sprintu.

3.7 Dostupné nástroje pro metodiku Scrum

Proces vývoje metodikou Scrum je možné organizovat několika různými formami. Jde zejména o způsob, kterým se budou uchovávat a šířit informace o plánování, průběhu sprintu a aktuálním stavu vývoje produktu. Každý z možných způsobů má své klady i zápory a jde o to zvolit nejvhodnější nástroj pro daný tým a projekt.

Základní faktory, na které je potřeba se zaměřit při výběru vhodné formy a nástroje, jsou následující:

- **Finanční náklady na využívání určitého nástroje** – některé nástroje jsou freeware, tedy zdarma, což je zpravidla na úkor kvality a doplňkových funkcí. Placené nástroje jsou v drtivé většině lépe vybavené užitečnými funkcemi.
- **Geografické rozmístění Scrum týmu** – u mezinárodních firem je relativně běžné, že vývojový tým je složen z lidí, kteří nesídlí v jedné kanceláři, dokonce ani na stejném světadíle. V takovém případě je nutné brát v úvahu možnost online připojení přes internet, odkudkoliv.
- **Funkční vybavenost** – každý dostupný nástroj obsahuje určité doplňkové funkce (například burndown a burnup grafy průběhu vývoje) a nabízí množství různých nastavení (například způsob nastavení kapacit týmů, možnost přiřazení jednoho uživatele k více týmům, nastavení měrných jednotek a další).

Podpůrných nástrojů pro metodiku Scrum existuje nepřeberné množství. Některé jsou placené, některé jsou zdarma. Většina z nich obsahuje všechny nezbytné funkce, ale liší se vzhledem a doplňkovými funkcemi. Nejpoužívanější z nich budou představeny v následujících podkapitolách.

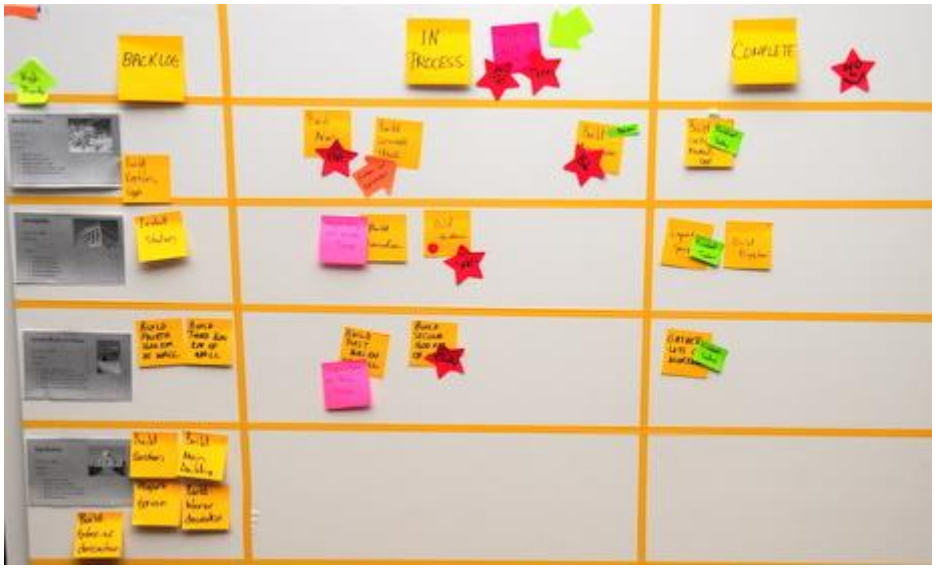
3.7.1 Whiteboard - nástěnka

Původním a nejjednodušším způsobem je tzv. Whiteboard[22]. V tomto případě nejde o softwarový nástroj, ale o fyzické znázornění plánovaných úkolů na nástěnce, například pomocí popsaných lístků. Nevýhodou této fyzické formy je nemožnost přímé účasti u geograficky rozmístěných týmů a absence veškerých automatizovaných doplňkových funkcí. Pro menší týmy a projekty, umístěné v jedné kanceláři, je však i tato forma dostatečná.

Tab. 3 Přehled nástroje Whiteboard, zdroj: vlastní

	Jednoduchost.
	Žádné automatizované doplňkové funkce, možnost účasti pouze fyzicky u nástěnky.
	Zdarma (neuvažujeme cenu běžných kancelářských potřeb).

Na následujícím obrázku je možné vidět příklad jednoduchého Scrum backlogu znázorněného na nástěnce – Whiteboardu.



Obr. 17 Náhled nástroje Whiteboard, zdroj: vlastní

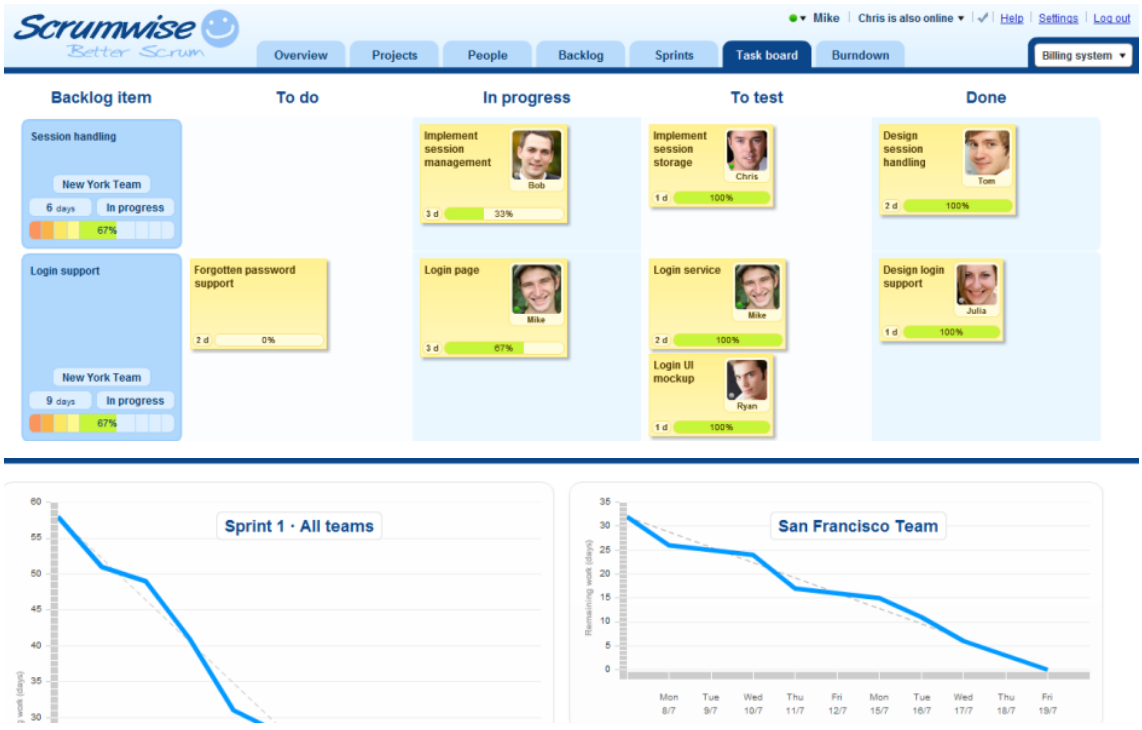
[22] **Whiteboard** – z anglického „tabule“, nebo „nástěnka“

3.7.2 Scrumwise

Scrumwise je Dánský softwarový Scrum nástroj. Jde o velmi jednoduchý a přehledný nástroj, který využívá funkci drag-n-drop[23] a obsahuje nejdůležitější doplňkové funkce. Scrumwise je využíván také některými velkými společnostmi, jako například Ikea, Electrolux, KBC, Siemens, Whirlpool apod.

Tab. 4 Přehled nástroje Scrumwise, zdroj: vlastní

	Uživatelsky velmi snadné a přehledné (využití funkce drag-n-drop), množství doplňkových funkcí: zobrazení Burnup a Burndown grafů, plánování relesů, sledování času – time tracking
	Placená aplikace (střední cena), chybí možnost vkládání příloh, nebo logování incidentů a defektů.
	Standardní cena: 9 EUR / uživatele / měsíc.



Obr. 18 Náhled nástroje Scrumwise, zdroj: vlastní

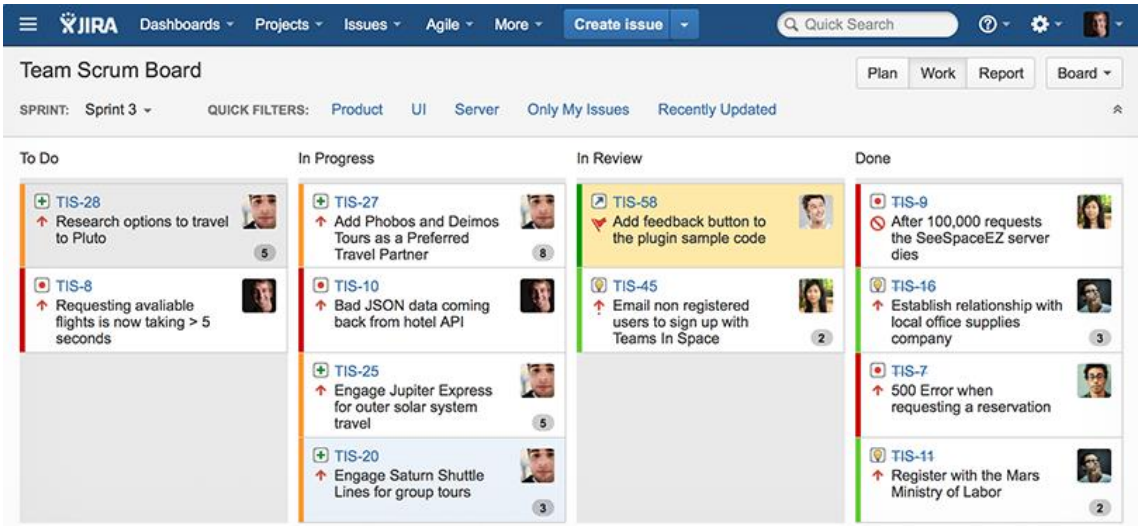
[23] **Drag-n-drop** – funkce, která umožňuje přemisťovat jednotlivé úkoly jednoduchým přetažením.

3.7.3 Jira - Confluence

Jira je produkt Australské firmy Atlassian. Jde o jeden z nejrozšířenějších nástrojů pro Scrum, který dnes poskytuje i doplňkovou aplikaci pro Apple iPhone/iPad. Poskytuje široké možnosti nastavení, včetně vlastního nastavení procesu vývoje. Aplikace se tím přizpůsobí systému práce ve firmě a nemusí se naopak firma přizpůsobovat aplikaci. Propracovaná je také možnost evidence defektů při testování, nebo zobrazení časových os, se všemi týmy a projekty.

Tab. 5 Přehled nástroje Jira, zdroj: vlastní

	Široké možnosti nastavení, včetně nastavení vlastního procesu vývoje, doplňková aplikace pro Apple iPhone/iPad, množství doplňkových funkcí: drag-n-drop, široké možnosti reportování, včetně Burndown a Burnup, možnost prioritizace úkolů, logování defektů, online chat, vkládání příloh.
	Placená aplikace, uživatelsky složitější.
	Do 25 uživatelů: 1.200 EUR / rok Do 50 uživatelů: 2.200 EUR / rok Do 500 uživatelů: 12.000 EUR / rok



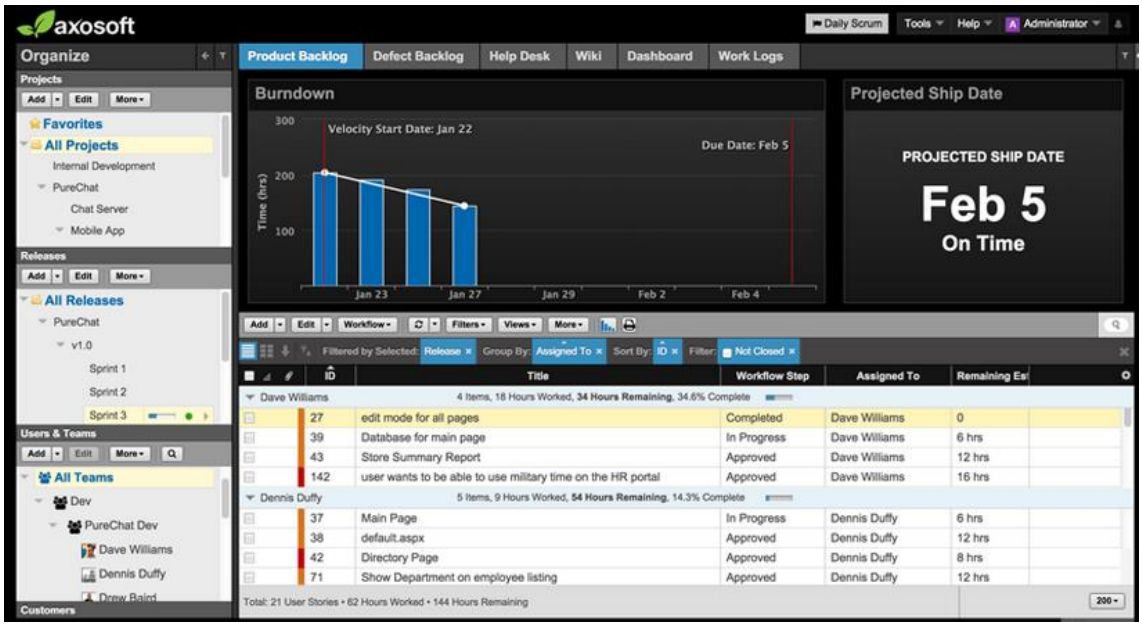
Obr. 19 Náhled nástroje Jira, zdroj: vlastní

3.7.4 Axosoft

Axosoft je další z placených aplikací pro Scrum. Na rozdíl od Scrumwise a Jira, není tato aplikace momentálně v KBC využívána. Obsahuje velké množství doplňkových funkcí, jako například logování incidentů a defektů, grafické zobrazení průběhu prostřednictvím burndown a burnup, možnost přiřazení jednotlivých úkolů ke konkrétním softwarovým releasům.

Tab. 6 Přehled nástroje Axosoft, zdroj: vlastní

	Možnost připojení online, široké možnosti nastavení a doplňkových funkcí: možnost logování defektů a incidentů, zobrazení Burnup a Burndown grafů, plánování releasů, sledování času – time tracking.
	Vyšší cena, uživatelsky složitější.
	Tým do 10 uživatelů: zdarma Tým více než 10 uživatelů: 25 EUR / uživatele / měsíc.



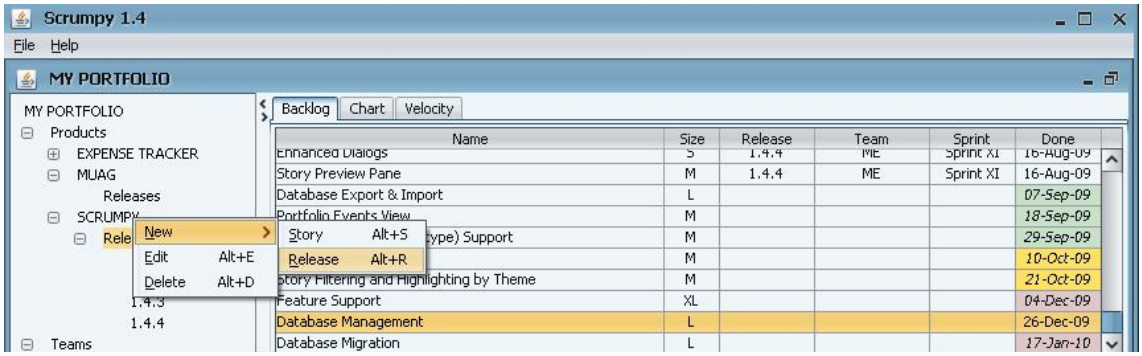
Obr. 20 Náhled nástroje Axosoft, zdroj: vlastní

3.7.5 Scrumpy

Pro srovnání je dobré podívat se i na některé nástroje, které jsou dostupné zcela zdarma, například Scrumpy. Nabízí všechny nutné funkce k organizaci Scrumu, ale na rozdíl od placených aplikací je méně vzhledově a uživatelsky přívětivý. Scrumpy je zaměřen zejména na využití product ownerem, vzhledem k rozšířeným možnostem dlouhodobého plánování.

Tab. 7 Přehled nástroje Scrumpy, zdroj: vlastní

	Možnost připojení online, potřebné doplňkové funkce: zobrazení Burnup a Burndown grafů, dlouhodobé plánování produktu (průběh zbývajících úkolů v produktovém backlogu), funkce drag-n-drop.
	chybí sledování času – time tracking, evidence defektů a oncidentů, méně vzhledově a uživatelsky přívětivý.
	Zdarma



Obr. 21 Náhled nástroje Scrumpy, zdroj: vlastní

3.7.6 Další dsotupné nástroje

Nástrojů pro Scrum je k dispozici velké množství. Mezi další známé zástupce placených aplikací pro Scrum patří například tyto:

- Mingle (více na http://www.thoughtworks.com/mingle/)
- Acunote (více na http://www.acunote.com/)
- Agile Fant (více na http://agilefant.com/)

- Agile Zen (více na http://www.agilezen.com/)
- Banana Scrum (více na http://www.bananascrum.com/)
- Eylean (více na http://www.eylean.com/)
- GravityDev (více na https://www.gravitydev.com/)
- QuickScrum (více na http://quickscrum.com/)
- 21Scrum (více na http://www.21scrum.com/)

Existuje také mnoho neplacených freeware aplikací, které však zpravidla nejsou tak propracované a nenabízí tolik doplňkových funkcí, jako aplikace placené. Patří mezi ně například tyto:

- Fulcrum (více na http://wholemeal.co.nz/projects/fulcrum.html)
- Ice Scrum (více na http://www.icescrum.org/)
- Pango Scrum (více na http://pangoscrum.com/)
- Scrum Factory (více na http://www.scrum-factory.com/)

3.7.7 Souhrn přehledu podpůrných nástrojů pro metodiku Scrum

Podpůrných aplikací pro Scrum existují desítky, ne-li stovky. Představeny byly nejznámější placené nástroje (Axosoft, Scrumwise a Jira), freeware aplikace (Scrumpy) a také jednoduchý způsob formou nástěnky.

Z uvedeného přehledu vyplývá, že placené aplikace jsou propracovanější a nabízejí větší množství doplňkových funkcí a možností nastavení. Pro větší projekty a velké vývojové týmy je tedy vhodné využít některou z těchto aplikací. Je nutné počítat s finančními náklady na licence těchto placených nástrojů, ale rozšířené možnosti plánování a reportování mohou mít velkou přidanou hodnotu při projektovém vývoji.

Pro malé vývojové týmy a krátkodobé projekty by bylo dostačující využít neplacené nástroje plánování, z důvodu úspory finančních nákladů na licence. Některé placené aplikace však nabízí omezené verze, nebo dokonce plné verze pro týmy do 10ti členů zdarma. Pro tým do 10ti členů se pak jeví nejvhodnější využít profesionální nástroj, který je poskytovaný zdarma, nebo variantu nástěnky – whiteboard.

Z uvedených profesionálních nástrojů (Axosoft, Scrumwise a Jira) nabízí Jira a Axosoft nejširší možnosti doplňkových funkcí a vlastních nastavení, včetně logování

problémů a reportování. Licence pro aplikaci Axasoft je ale v případě týmů nad 10 členů jednoznačně nejdražší. Axasoft je vhodné využít u malých týmů, jelikož je v tomto případě zdarma. Jak Jira, tak Axasoft se však jeví uživatelsky relativně komplikované. Oproti tomu nástroj Scrumwise je zaměřený na jednoduchost a přehlednost. Vzhledem k tomu, že tento nástroj a jeho data mohou být využívány nejen vývojovým týmem a Scrum masterem, ale také product ownerem, případně dalšími zainteresovanými stranami, je přehlednost a intuitivní funkčnost důležitým faktorem.

Pro velké projekty a vývojové týmy se tedy jeví nejvhodnější Scrumwise a Jira, z důvodu množství doplňkových funkcí, jednoduchosti a přehlednosti. Pro malé týmy je vhodný nástroj Axasoft, který je pro týmy do 10 členů nabízen zdarma.

3.8 Souhrn analýzy současného stavu

V předcházejících kapitolách byla podrobně rozepsána současná aplikace metodiky Scrum, v jednom z vývojových týmů společnosti KBC.

Geografické rozmístění celého Scrum týmu podstatně znesnadňuje vzájemnou spolupráci a komunikaci. Pomocí moderních elektronických aplikací, jako jsou videohovory, nebo i aplikace pro organizaci Scumu, je tato překážka téměř eliminována. Absence osobního kontaktu a mírná jazyková bariéra však stále zůstávají.

Jedním z problémů je nejasná role product ownera. Přidávat položky do produktového backlogu mohou totiž nejen product owneři z týmu řízení aplikací, ale také projektoví manažeři. Tím product owneři ztrácí přehled o vlastním backlogu, což ztěžuje jejich pozici při jednáních o plánování následujících sprintů. Samotné plánování pak připravuje Scrum master na základě backlogu. Výsledný plán spritu pak na společném setkání odsouhlasí všechny strany Scrum týmu. Roli Scrum mastera zde vykonává manažer vývojového týmu.

Momentálně jsou využívány pro každý účel jiné softwarové aplikace. Defekty jsou evidovány v aplikaci QualityCenter, změnové požadavky s podrobnými detaily v aplikaci ServiceNow a plánování metodikou Scrum v plikaci Scrumwise. To nejen zvyšuje administrativní náročnost, ale také vzhledem k nutné duplicitě některých informací zvyšuje riziko chyb.

Denního setkání sprintu se účastní vývojový tým a Scrum master, což je dohromady dva krát patnáct účastníků. Vzhledem k tomuto počtu členů každého týmu je toto denní setkání poměrně dlouhé a trvá přibližně půl hodiny, místo doporučovaných deseti minut.

Iterativní přístup je u zkoumaného týmu viditelný. Inkrementální přístup je také viditelný, ale ne přímo. Inkrementy produktu, dodávané na produkční prostředí, jsou totiž publikovány v takzvaných releasech. Načasování těchto releasů je přibližně jednou za dva měsíce, což neodpovídá plánování sprintů. Proto není dodáván inkrement produktu vždy po konci sprintu. Průběh vykonaných úkolů v rámci sprintu je sledován pomocí ukazatele burndown. Průběh vykonaných úkolů pro konkrétní release inkrementu produktu však sledován není. Stejně tak není využíván ukazatel burnup, který většina podpůrných nástrojů nabízí.

Po ukončení každého sprintu se nekoná přezkoumání sprintu, ani retrospektivní setkání. Tyto zpětné pohledy by měly sloužit k přehodnocení nastaveného procesu vývoje a dodaných inkrementů.

Změny v průběhu vývoje jsou akceptovány. Scrum tým je schopný pružně reagovat na změny v požadavcích i v průběhu vývoje.

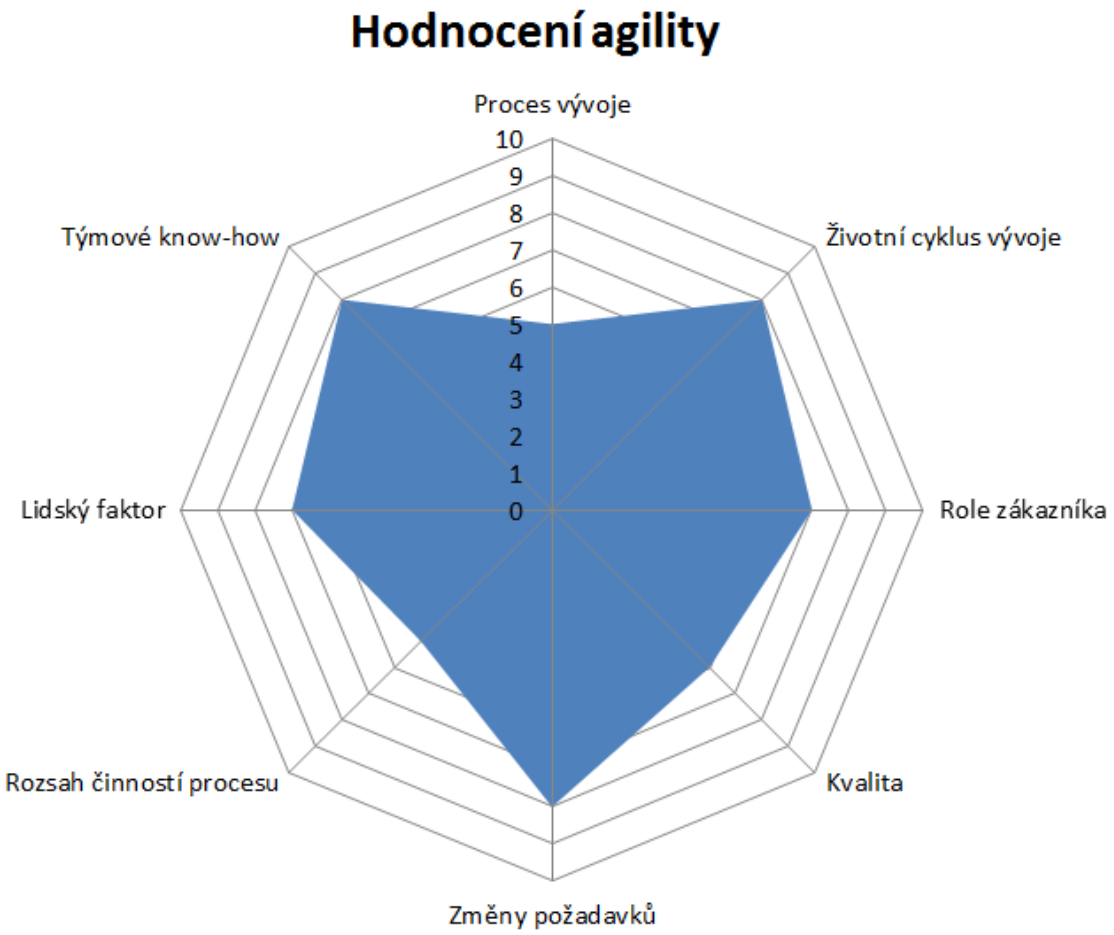
Na trhu je dostupné velké množství podpůrných aplikací, ať už placených, tak zdarma. Placené aplikace jsou zpravidla více propracované a nabízí širší množství podpůrných funkcí. Zkoumaný tým v současné době používá jednoduchý, avšak přehledný softwarový nástroj Scrumwise, který však neposkytuje tak široké možnosti funkcí, jako jiné představené produkty.

Následující tabulka znázorňuje hodnocení agility současného přístupu u zkoumaného vývojového týmu.

Tab. 8 Hodnocení agility současného přístupu, zdroj: vlastní

Oblast	Popis situace	Hodnocení agility (1 – 10)
Proces vývoje	Proces spíše definovaný a opakovatelný. V případě nutnosti však možnost změn v průběhu vývoje.	5
Životní cyklus vývoje	Plánování v iteracích. Inkrementy produktu však nejsou dodávány po sprintu.	8
Role zákazníka	Product owner spravuje celý backlog a zasahuje do plánování sprintu. V průběhu sprintu se však přehled ztrácí.	7
Kvalita	Ze strany vývojového týmu je stále tlak na dodržování standardního procesu, místo pružnosti pro vyšší užitek pro zákazníka.	6
Změny požadavků	Změny jsou většinou přijímány, a to bez nutnosti řízení změn.	8
Rozsah činností procesu	Současný proces je poměrně komplikovaný, s množstvím administrativy. Denní setkání jsou velmi dlouhé.	5
Lidský faktor	Důraz na dokumentaci. Lidský faktor a zkušenosti jedinců jsou však využívány.	7
Týmové know-how	Složitější problémy jsou řešeny kolektivně, informace jsou v týmu šířeny.	8

Toto hodnocení je přehledně zobrazeno následujícím obrázkem Obr. 22. Čím větší je výsledná plocha, tím agilnější je přístup týmu.



Obr. 22 Graf hodnocení agility současného přístupu, zdroj: vlastní

4 NÁVRH ŘEŠENÍ A PŘÍNOS NÁVRHŮ ŘEŠENÍ

Zkoumaný vývojový tým má snahu řídit vlastní vývoj agilním způsobem, konkrétně metodikou Scrum development process. Z předchozích kapitol vyplývá, že je v některých oblastech zřejmý agilní přístup, v některých oblastech jsou však viditelné pozůstatky tradičního řízení. Také metodika Scrum není využívána zcela podle doporučených praktik, popsaných ve většině literatury.

4.1 Návrhy řešení

Tato část je zaměřena na změny v oblastech, kde byly objeveny nedostatky.

4.1.1 Struktura a geografické rozložení týmů

Vzhledem ke geografickému rozmístění členů vývojového týmu a týmu řízení aplikací, je nemožný přímý osobní kontakt a veškeré jednání musí probíhat prostřednictvím elektronických kanálů. Někteří členové osobně potvrdili, že je řešení některých otázek bez osobního kontaktu obtížnější a problémy nejsou prodiskutovány tak podrobně, jako by mohly být při osobním jednání.

Negativní dopad má i mírná jazyková bariéra. Komunikace mezi třemi národnostmi sice probíhá zásadně v anglickém jazyce, ale pro žádného člena nejde o rodný jazyk.

Tým řízení aplikací byl přesunut do Brněnského centra za účelem snížení nákladů. Výsledkem jsou v současné době poloviční náklady, které jsou dokonce spojeny s nárůstem produktivity, oproti původnímu týmu, který dříve fungoval v Belgii. Tento fakt nabízí otázku: Proč nepřesunout i Belgický vývojový tým do Brna, případně do Indie, kde už v současnosti působí 2 členové tohoto týmu? Odpovědí je obava managementu z kvality vývoje novým týmem, který nemá tolik zkušeností. Tato obava by však mohla být vyvrácena dosavadní zkušeností s týmem řízení aplikací.

Řešení problému

Nejvhodnějším řešením se jeví přesun vývojového týmu do Brněnského centra. Tato varianta by ve výsledku znamenala pokles nákladů na vývojový tým přibližně na polovinu. Vzhledem k dobrým výsledkům jiných přesunů do Brna je pravděpodobné, že

kvalita vývoje by zůstala minimálně na stejné úrovni. Díky počtu technicky zaměřených vysokých škol v Brně je na lokálním trhu práce dostatek IT pracovníků, takže v Brně nehrozí ani problém s hledáním nových IT pracovníků.

Současní dva členové vývojového týmu v Indii plní jakousi roli rezervy, při nedostatku kapacity ostatních členů vývojového týmu. Vzhledem k nákladnosti těchto členů, která je na úrovni jedné čtvrtiny původních Belgických zaměstnanců, je vhodné tyto Indické pracovníky ponechat i nadále pro jednodušší úkoly a pro případ nedostatku kapacity.

4.1.2 Role Scrum týmu

Scrum master

Podle dostupných zdrojů, by měl být v roli Scrum mastera manažer, analytik, nebo člověk speciálně pro roli Scrum mastera. Je však doporučované, aby Scrum masterem nebyl jeden z programátorů, ani Scrum master, kvůli možnému střetu zájmů a v případě programátora i omezenému nadhledu na celý problém. Scrum master by měl podporovat vývojový tým, starat se o to, aby měli k dispozici vše potřebné ke svojí práci, jedná s product ownerem řeší jakékoliv problémy a v případě potřeby i upravuje nastavené procesy tak, aby byly výsledky vývojového týmu co nejlepší.

Roli Scrum mastera vykonává manažer každého z vývojových týmů, což odpovídá doporučením a všem požadavkům na tuto roli.

Product owner

Role product ownera je poměrně nejasná. Je sice definováno, kdo je v roli product ownera, ale přitom mají k backlogu požadavků přístup i další strany, které do něj mohou přidávat vlastní požadavky. Jde zejména o projektové manažery a okolní IT týmy. Product owner tak ztrácí přehled o vlastním backlogu. Nejen že nemá neustále seřazené požadavky podle priorit, ale některé požadavky ve vlastním backlogu ani nezná a neví, jak jsou důležité. Při plánování následujícího sprintu tak může dojít k zařazení položky s nízkou prioritou do sprintového backlogu, přesto že jiné s vyšší hodnotou pro zákazníka zůstávají stále v backlogu.

Řešení problému

Product owner by měl být jediný, kdo má pravomoc jakkoliv měnit vlastní produktový backlog. Veškeré požadavky od okolních stran, například od projektových manažerů, nebo jiných IT týmů, by měly být podány a vysvětleny product ownerovi, který má na starosti danou aplikaci. Sám product owner pak tyto požadavky doplní do svého backlogu.

4.1.3 Řízení produktového backlogu

První změna v řízení produktového backlogu byla popsána v předchozím odstavci, zabývajícím se rolí product ownera.

Podstatný dopad na řízení produktového backlogu mají také aplikace, které jsou využívány. Doporučení týkající se výběru využívaných aplikací pro organizaci vývoje, je uvedeno v kapitole 4.1.8 níže.

Při pohledu na řízení produktového backlogu byl identifikován další problém. Jde o obrovské množství požadavků v backlogu, které se navíc zvyšuje rychleji, než je vývojový tým schopný dodávat. Důsledkem toho je nárůst doby implementace nových požadavků, čímž vývoj přestává být dostatečně agilní. Současný backlog obsahuje přibližně 200 nových požadavků, z čehož některé čekají ve frontě i více než 2 roky.

Řešení problému

Zvyšující se tlak na množství vykonaných požadavků pak také negativně působí na výslednou kvalitu. Zkracování doby vývoje každé změny není způsobeno zkrácením doby vývoje, ale doby testování a aktualizace technické dokumentace, což vede k nárůstu produkčních incidentů. Čas věnovaný opravám produkčních chyb je pak zpravidla několikanásobně vyšší, než pokud by byla větší pozornost věnována testování. Tlak na zkracování doby vývoje jednotlivých položek backlogu tedy řešením není.

Správným řešením je zde zásah product ownera, který by měl pravidelně procházet celý backlog a odstraňovat z něj požadavky, u kterých je přínos hodnoty zákazníkovi menší, než náklady na jeho vybudování. Stejně tak je nutné přehodnocovat veškeré nové požadavky. Nové požadavky se zápornou přidanou hodnotou by se do backlogu měly přidávat pouze v případě nutnosti, což může být požadavek auditu, oddělení risku,

nebo kvůli dodržení právních požadavků. Současně je nutné všechny požadavky neustále řadit podle priorit a přidané hodnoty, aby byly nejdříve implementovány ty nejhodnotnější.

Pokud by se ani tak nepodařilo redukovat počet položek v backlogu a zastavit tempo nárůstu, druhá a výrazně nákladnější varianta je pak rozšíření vývojového týmu, s čímž je spojeno i rozšíření týmu řízení aplikací.

4.1.4 Plánování sprintu

Aktuálně je proces nastaven tak, že plánování připravuje Scrum master, na základě produktového backlogu. Nejdříve definuje dostupnou kapacitu na následující období, pak vytvoří rezervace na standardní ostatní nezbytné úkoly (například podpora, nedostupnost, školení apod.), zbylá volná kapacita je pak naplněna položkami z backlogu, které jsou umístěny nejvýše. Plánováno je tolik úkolů, aby byla přesně naplněna dostupná kapacita.

Takto připravený backlog sprintu je pak odsouhlasován na společném jednání, kterého se účastní celý Scrum tým. Toto jednání trvá přibližně hodinu a půl, jelikož se postupně prochází položka po položce. Po odsouhlasení backlogu jsou všechny úkoly rovnoměrně přiřazeny jednotlivým členům a každý si pak hlídá své vlastní úkoly. Tento postup není vyloženě špatný, ale jsou zde dva body, na které by bylo dobré se zaměřit.

Řešení problému

Prvním bodem je účast celého Scrum týmu na plánování sprintu. Vzhledem k tomu, že diskuze zde probíhá zejména mezi Scrum masterem, který připravuje plán na příští sprint, a mezi product ownerem, který tento plán odsouhlasí, není nutné, aby se tohoto jednání účastnil i celý vývojový tým. Každá z položek má již uveden odhad nákladů (odhad byl vyplněn v předchozích sprintech, prostřednictvím položek „analýza“, viz kapitola 3.6.2). U každé položky je tedy jasný přínos i náklady. Jednání o plánování sprintu může probíhat jen mezi product ownerem a Scrum masterem. Tím je možné ušetřit podstatné množství času pro celý vývojový tým.

Druhým bodem je rozdělování úkolů, které se momentálně děje ihned po odsouhlasení backlogu sprintu, a to pro všechny naplánované položky. Podle

doporučení v literatuře, je možné zvýšit agilitu týmu tím, že se jednotlivé úkoly přiřadí jednotlivým členům až v daný den ráno, na denním setkání sprintu. Doporučený způsob rozdělování úkolů je popsán podrobněji v následující kapitole. Při plánování sprintu by neměly být hned přiřazeny všechny úkoly jednotlivým členům. Snižuje se pružnost při vývoji, jelikož úkoly pak nejsou předávány z jednoho člena na druhého, podle aktuální situace.

4.1.5 Denní sprint setkání

Denního setkání sprintu se účastní vývojový tým a Scrum master, což je dohromady dva krát patnáct účastníků. Vzhledem k tomuto počtu členů každého týmu je toto denní setkání poměrně dlouhé a trvá přibližně půl hodiny, místo doporučovaných pěti minut. Při tomto setkání se aktualizuje průběh úkolů a diskutují se libovolná témata. Product owner se tohoto jednání neúčastní a nemá tak přehled detailní přehled o průběhu vývoje v rámci sprintu.

Přiřazování jednotlivých úkolů členům vývojového týmu se na denním setkání neřeší, jelikož jsou všechny rozděleny už od samotného naplánování sprintu.

Řešení problému

Účelem tohoto setkání by mělo být nejen aktualizovat stav úkolů v aplikaci pro Scrum (Scrumwise), ale hlavně podělit se vzájemně o nové poznatky a aktuální problémy při vývoji. To podporuje vzájemnou informovanost jednotlivých členů vývojového týmu a zvyšuje efektivitu vývoje. Každý člen vývojového týmu by měl odpovědět na tyto čtyři otázky:

- „Co jsem udělal, nebo dokončil od posledního setkání?“
- „Na čem budu pracovat teď a kolik předpokládám, že zvládnu dokončit, do dalšího setkání?“
- „Podařilo se mi vyřešit nějaké problémy a jak?“
- „Brání mi momentálně nějaké problémy k dokončení úkolů?“

Denní setkání by mělo trvat maximálně 15 minut, aby nezabíral zbytečnou kapacitu týmu. Scrum master by se měl více soustředit na moderování tohoto setkání, udržovat

diskuzi v potřebném rytmu a nenechat účastníky zacházet příliš do detailu jednoho problému. Pokud je potřeba nějaký problém řešit více do hloubky, je lepší řešení tohoto problému probrat na zvláštním setkání, kterého se nemusí účastnit celý tým.

Jak bylo zmíněno v předchozí kapitole o plánování sprintu, pro dosažení vyšší agility je doporučováno jednotlivé úkoly v backlogu sprintu přiřazovat jednotlivým členům až přímo na denním setkání. Na denním setkání si jednotliví členové rozebírají jednotlivé úkoly, podle priority převzatých z backlogu, při zodpovídání této otázky: „Na čem budu pracovat teď."

Dalším doporučením je zpřístupnění denních setkání i product ownerovi, který by se mohl účastnit jako posluchač, případně ihned zodpovídat rychlé otázky a aktualizovat prioritu.

4.1.6 Konec sprintu

Konec sprintu není následován přezkoumáním sprintu, ani retrospektivou. Tyto zpětné pohledy by měly sloužit k přehodnocení nastaveného procesu vývoje a dodaných inkrementů.

Řešení problému

Doporučené je obě tyto setkání organizovat po každém sprintu, je možné i dohromady při jednom setkání.

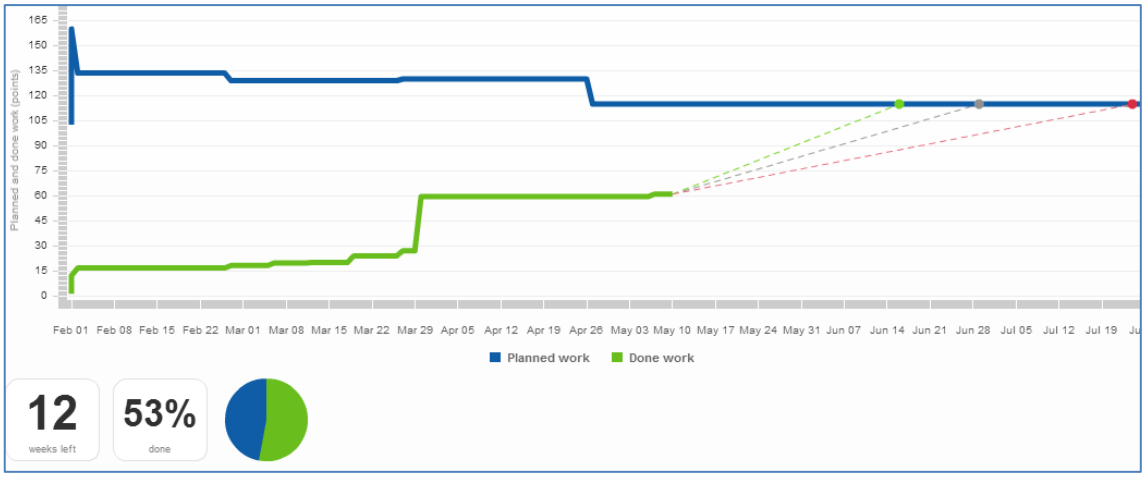
4.1.7 Sledování průběhu vývoje

Průběh vykonaných úkolů v rámci sprintu je sledován pomocí ukazatele burndown. Průběh vývoje položek, pro konkrétní release inkrementu produktu, však sledován není. Product owner tak má přehled o průběhu sprintu, ale nemá přehlednou představu o průběhu položek pro konkrétní release inkrementu produktu. Pro tento přehled musí kontrolovat všechny položky jednotlivě, protože mohou být rozplánovány v několika různých sprintech. Také není využíván ukazatel burnup, který většina podpůrných nástrojů nabízí, využíván je pouze burndown.

Řešení problému

Nadále využívat ukazatel burndown ke sledování průběhu vývoje v rámci sprintu. Tento ukazatel slouží zejména pro Scrum mastera, který hlídá průběh na straně vývojového týmu.

Pro sledování průběhu jednotlivých releasů, je možné určité položky v produktovém backlogu přiřadit jednotlivým releasům. Takto přiřazené položky pak tvoří inkrement produktu, který bude doručen s konkrétním releasem. Pro sledování vývoje právě těchto položek je vhodné využít ukazatel burnup. Tento ukazatel pak slouží zejména pro product ownera.



Obr. 23 Ukazatel burnup, zdroj: vlastní

4.1.8 Využití softwarových nástrojů

Momentálně jsou využívány pro každý účel jiné softwarové aplikace. Defekty jsou evidovány v aplikaci QualityCenter, změnové požadavky s podrobnými detaily v aplikaci ServiceNow a plánování metodikou Scrum v plikaci Scrumwise. Toto množství aplikací znamená nejen vysoké náklady na softwarové licence, ale také nadbytečné množství administrativy. Duplicita některých informací také zvyšuje riziko chyb.

Řešení problému

Aplikace Scrumwise je sice velmi přehledná a uživatelsky snadná, ale nabízí funkce pouze k plánování vývoje metodikou Scrum. Oproti tomu profesionální aplikace k plánování vývoje, jako například Jira, nebo Axasoft, nabízí široké možnosti, jako

například evidence defektů a incidentů. V případě, kdy jsou všechny tyto informace vedeny v jedné aplikaci, nabízí se další možnosti reportování a automatizace. Může jít například o počet defektů a jejich stav, k jednotlivým položkám backlogu, nebo automatické vytvoření položky v backlogu na opravu incidentu. S možností připojení příloh a nastavení workflow jednotlivých položek je pak možné vést přímo v jedné aplikaci i samotnou evidenci požadavků s kompletní dokumentací a schvalováním.

Vzhledem k výše uvedeným možnostem se jeví nejvhodnější aplikace Jira s doplňkem Confluence, která stejně jako Axasoft nabízí možnost evidence incidentů a defektů, ale navíc umožňuje vlastní nastavení workflow procesu jednotlivých funkcí.

Podstatným faktorem je také fakt, že společnost KBC již využívá aplikaci Jira a vlastní licence pro velké množství uživatelů. To znamená dostupnou podporu zkušenějších uživatelů v rámci firmy a nižší náklady na licence.

Všechna uvedená fakta tedy vedou k doporučení nahradit aplikací Jira nejen aplikaci Scrumwise pro Scrum, ale také QualityCenter pro evidenci defektů a ServiceNow pro evidenci požadavků a incidentů. Po správném nastavení umožní aplikace Jira všechny potřebné funkce těchto aplikací a navíc je možné automatizovat vzájemné interakce mezi těmito funkcemi.

4.2 Přínos navržených řešení

Tato část vysvětluje přínos všech navržených změn. Popsány budou náklady nutné k provedení těchto změn, stejně jako očekávaný přínos z každého doporučení.

4.2.1 Přesun vývojového týmu

Přesun vývojového týmu do Brněnského centra znamená investiční náklad, který v sobě zahrnuje nábornových zaměstnanců a jejich školení, kancelářské prostory, vybavení a další. Pro přesnou kalkulaci těchto investičních nákladů je vhodné vytvořit projekt, v rámci kterého budou přezkoumána všechna rizika a nutná investice. Vzhledem k předchozím zkušenostem v rámci Brněnského centra však můžeme investiční náklad na vytvoření nového oddělení odhadnout na 2.600 EUR / zaměstnance.

Z dlouhodbého hlediska jsou podstatné náklady na jednoho zaměstnance v rámci Brněnského centra, které v případě IT dosahují 385 EUR / FTE[24] / den. Tato částka v sobě zahrnuje veškeré náklady spojené s jedním IT zaměstnancem, včetně vybavení, kancelářských prostor, nebo softwarových licencí. Průměrný počet pracovních dní v měsíci budeme uvažovat 22 dní.

Finanční přínos z dlouhodobého hlediska jsou zejména úspory za zaměstnance z bývalého IT oddělení v Belgii. Průměrné celkové náklady na IT zaměstnance v Belgii jsou 670 EUR / FTE / den.

Hlavním přínosem, který je sledovaný touto prací, je však zlepšení vzájemné komunikace v rámci Scrum týmu, odstraněním jazykové bariéry a umožněním osobního kontaktu. Zlepšení vzájemné komunikace může dle odhadu vést ke snížení chybovosti při vývoji až o 15% a zvýšení produktivity o 5%. Snadnost a efektivita komunikace je pro agilní řízení zásadním faktorem.

Tab. 9 Přínos z přesunu vývojového týmu, zdroj: vlastní

Ukazatel	**Výsledná hodnota**	**Poznámka**
Jednorázová finanční investice	78.000 EUR	2.600 EUR * 30 zaměstnanců
Finanční náklady na zaměstnance	254.100 EUR / měsíc	385 EUR * 30 zaměstnanců * 22 dní
Finanční úspory na zaměstnance	442.200 EUR / měsíc	670 EUR * 30 zaměstnanců * 22 dní
Dlouhodobý finanční dopad	188.100 EUR / měsíc	Finanční úspory – finanční náklady
Nefinanční přínos	Produktivita vývojového týmu: + 5% Kvalita - chyby při vývoji: - 10% (snížení počtu defektů)	

[24] **FTE** – Zkratka užívaná pro označení doby práce jednoho člověka, anglicky „**Full-time equivalent**“

Převedením vývojového týmu tedy může společnost (po počáteční investici ve výši 78.000 EUR) nejen spořit 188.100 EUR měsíčně, ale také mírně zvýšit produktivitu vývojového týmu a snížit počet chyb při vývoji.

4.2.2 Jediným správcem backlogu je product owner

Product owner by měl být jediný, kdo má pravomoc jakkoliv měnit vlastní produktový backlog. Zde jde pouze o změnu procesu, kdy projektoví manažeři a okolní týmy budou vlastní požadavky předkládat product ownerovi dané aplikace, který tyto požadavky musí nastudovat a pochopit a vytvoří je ve vlastním backlogu.

Nákladem je zde navýšení pracovního vytížení product ownera, který ude spravovat více položek, než doposud. Správa backlogu je pro product ownera přibližně 0,25 FTE. Z celkového počtu 213 požadavků, je 20% požadavků od okolních týmů. Pro product ownera tato změna znamená nárůst pracovního vytížení správou backlogu na 0,3 FTE. Náklady na business aplikací, jsou v Brněnském centru 255 EUR / FTE / den.

Přínosem této změny je nárůst efektivity plánování a dodávání inkrementů produktu, obsahujích změny, které mají pro zákazníky momentálně nejvyšší přínos. Vyčíslení tohoto přínosu je velmi obtížné. Sledovat však můžeme dopad na snížení změn backlogu sprintu v průběhu vývoje, nebo nárůst spokojenosti zákazníků.

Tab. 10 Přínos ze změny přístupu k produktovému backlogu, zdroj: vlastní

Ukazatel	Výsledná hodnota	Poznámka
Náklady: čas product ownera	247,5 EUR / měsíc	0,05 FTE * 225 EUR * 22 dní
Nefinanční přínos	Změny v průběhu vývoje: - 10% Spokojenost zákazníků: + 5%	

4.2.3 Aktivní třídění požadavků v produktovém backlogu

Aktivně odstraňovat backlogové požadavky, u kterých je přínos hodnoty zákazníkovi menší, než náklady na jeho vybudování. Stejně tak je nutné přehodnocovat veškeré nové požadavky.

Náklady je zde opět možné identifikovat navýšením pracovního vytížení product ownera, konkrétně o 0,1 FTE. Přínosy jsou zde zřejmé ve zkrácení celkové doby dodávání jednotlivých požadavků a tím i zvýšení spokojenosti zákazníků. Průměrná doba dodání jednoho požadavku může být snížena z 10 měsíců, na 8 měsíců.

Tab. 11 Přínos z aktivního třídění produktového backlogu, zdroj: vlastní

Ukazatel	**Výsledná hodnota**	**Poznámka**
Náklady: čas product ownera	495 EUR / měsíc	0,1 FTE * 225 EUR * 22 dní
Nefinanční přínos	Doba time to market: - 20% Spokojenost zákazníků: + 5%	2 měsíce = 20% z 10

4.2.4 Změna organizace plánování sprintu

V případě, že se jednání o plánování sprintu účastní pouze product owner a Scrum master, zbytek vývojového týmu se může těchto 90 minut věnovat vlastní práci. Když vezmeme v úvahu celý vývojový tým, což je 2 x 14 vývojářů, může tím být ušetřeno několik hodin práce. Pokud budeme uvažovat, že je vývojový tým přesunut do Brněnského centra, vzhledem k nižším nákladům bude i nižší úspora.

Tab. 12 Přínos ze změny organizace plánování sprintu, zdroj: vlastní

Ukazatel	**Výsledná hodnota**	**Poznámka**
Náklady	0	Nebyly identifikovány
Přínos: úspora času (vývojový tým Belgie)	1.940 EUR / měsíc (3.377 EUR / měsíc)	0,18 FTE * 385 EUR (670 EUR) * 28 zaměstnanců * 1 den

4.2.5 Změna organizace denních setkání sprintu

Změna organizace a moderování denních setkání zlepší informovanost vývojového týmu o problémech a jejich řešení a navíc zkrátí dobu těchto setkání. Při zkrácení těchto setkání na polovinu je možné ušetřit 15 minut třiceti lidí každý den, což je 0,03 FTE pro jednoho.

Product owner by díky občasné účasti na denním setkání měl detailní přehled o průběhu a problémech při vývoji. Zlepší se tedy informovanost product ownera o průběhu vývoje, podle předpokladu o 10%.

Tab. 13 Přínos ze změny organizace denních setkání sprintu, zdroj: vlastní

Ukazatel	**Výsledná hodnota**	**Poznámka**
Náklady: čas product ownera	148,5 EUR / měsíc	0,03 FTE * 225 EUR * 22 dní
Přínos: úspora času (vývojový tým Belgie)	7.623 EUR / měsíc (16.266 EUR / měsíc)	0,03 FTE * 385 EUR (670 EUR) * 30 zaměstnanců * 22 dnů
Nefinanční přínos	Šíření know-how vývojářů: +5% Informovanost prod. ownera: +10%	

4.2.6 Hodnocení průběhu sprintu

Doporučení je po každém sprintu organizovat setkání, které může trvat 45 minut (0,95 FTE / zaměstnance / den) a tématem bude retrospektiva a přezkoumání sprintu. Náklady jsou tedy čas Scrum týmu, jednou za měsíc.

Na základě těchto setkání by měly být odstraňovány nedostatky a zbytečné části současného procesu, k zefektivnění vývoje. Zefektivnění procesu vede ke zvýšení produktivity a také kvality dodávaných inkrementů.

Tab. 14 Přínos ze zavedení hodnocení sprintu, zdroj: vlastní

Ukazatel	**Výsledná hodnota**	**Poznámka**
Náklady: čas product ownera	213,75 EUR / měsíc	0,95 FTE * 225 EUR
Náklady: čas vývojářů (vývojový tým Belgie)	10.972,5 EUR / měsíc (19.095 EUR / měsíc)	0,95 FTE * 385 EUR (670 EUR) * 30 zaměstnanců
Nefinanční přínos	Produktivita vývoje: +4% Kvalita - chyby při vývoji: - 3%	

4.2.7 Sledování průběhu vývoje inkrementu

Přiřazování jednotlivých položek backlogu jednotlivým releasům, umožňuje tvořit skupiny požadavků, které budou implementovány jedním inkrementem produktu, neboli v jednom releasu. U těchto inkrementů pak může product owner přehledně sledovat průběh vývoje.

Náklady jsou zde zřejmé v mírném nárůstu administrativní náročnosti práce product ownera (0,02 FTE / den). Přínosem však je zvýšení přehledu o průběhu vývoje inkrementu a možnost reportování ukazatelem burnup.

Tab. 15 Přínos ze sledování průběhu vývoje inkrementu, zdroj: vlastní

Ukazatel	**Výsledná hodnota**	**Poznámka**
Náklady: čas product ownera	99 EUR / měsíc	0,02 FTE * 225 EUR * 22 dní
Nefinanční přínos	Možnost reportování průběhu vývoje inkrementu produktu -> burnup Informovanost prod. ownera: + 10%	

4.2.8 Přechod na aplikaci Jira

Přechod na aplikaci Jira Confluence umožní odstranění tří jednotlivých aplikací: Scrumwise, ServiceNow a QualityCenter. Díky tomu je možná úspora nákladů na softwarové licence těchto aplikací, pro všechny členy Scrum týmu.

Z pohledu agilního vývoje je však důležitým přínosem této změny snížení administrativní náročnosti, eliminace duplicity informací, automatizace vzájemných interakcí mezi jednotlivými funkcemi a podstatné rozšíření možností reportování.

Vzhledem k tomu, že společnost KBC již vlastní skupinovou licenci aplikace Jira, využívání této aplikace dalším vývojovým týmem nijak nezvýší náklady na její licence. Znatelným nákladem však bude čas, vývojářů, věnovaný nastavení nové aplikace podle vlastních potřeb a její otestování (5 FTE / den) a také migrace současných dat z ostatních tří aplikací (10 FTE / den).

Tab. 16 Přínos z přechodu na aplikaci Jira, zdroj: vlastní

Ukazatel	Výsledná hodnota	Poznámka
Náklady: licence Jira	0	KBC již vlastní skupinovou licenci
Jednorázový náklad: nastavení aplikace a migrace dat (týmem v Belgii)	5.775 EUR (10.050 EUR)	15 FTE * 385 EUR (670 EUR)
Přínos: licence Scrumwise	360 EUR / měsíc	9 EUR * 40 uživatelů
Přínos: licence QualityCenter	0	KBC vlastní skupinovou licenci
Přínos: licence ServiceNow	517 EUR / měsíc	155 EUR / 12 měsíců * 40 uživatelů
Nefinanční přínos	Produktivita vývoje: + 3% Eliminace duplicity dat požadavků Rozšíření možností reportování	(Pokles administrativní náročnosti)

4.2.9 Souhrn

V této kapitole byl zhodnocen ekonomický dopad jednotlivých návrhů ke změnám.

Zásadní změna byla doporučena v oblasti geografického rozložení týmů, kde se dlouhodobě nejvhodnějším řešením jeví přesunu Belgického vývojového týmu do Brněnského centra. Současné nastavení rolí ve Scrum týmu se jeví efektivní. Změna byla však doporučena v oblasti řízení backlogu, kde product owner by měl být jediným oprávněným k provádění změn v produktovém backlogu. Je zde také nutnost redukce množství položek v současném backlogu. Plánování sprintu může být organizováno bez účasti celého vývojového týmu, čímž může být ušetřeno celkem 5 dnů lidské práce každý měsíc. V oblasti denních setkání je hlavní doporučení zaměřené na jejich moderování. Zkrácení těchto setkání, z 30 minut na 15 minut, ušetří až 20 dnů lidské práce za měsíc. Denní setkání by mohly být zpřístupněny i product ownerovi, který může přispívat aktuálními informacemi a získat lepší přehled o problémech při vývoji.

Konec sprintu by měl být doprovázen přezkoumáním a retrospektivou. Účelem těchto jednání je upravovat proces vývoje, předcházet problémům při vývoji a odstraňovat vše zbytečné, což jsou vše ukazatele agility vývoje. Průběhu sprintu je sledován ukazatelem burndown. Vzhledem k rozdílnému načasování sprintů a releasů inkrementů produktu je vhodné zavést sledování vývoje celého inkrementu. Pro přehlednost byl doporučen ukazatel burnup.

Důležité doporučení je v oblasti využívaných softwarových aplikací. Využitím aplikace Jira vhodným způsobem je možné nahradit nejen aplikaci Scrumwise, ale i QualityCenter pro vedení defektů a ServiceNow pro vedení indicentů a evidenci požadavků.

Souhrn celkového efektu z provedených změn je znázorněn v následující tabulce. Předpokládáme situaci, kdy jsou zavedeny všechny doporučené změny.

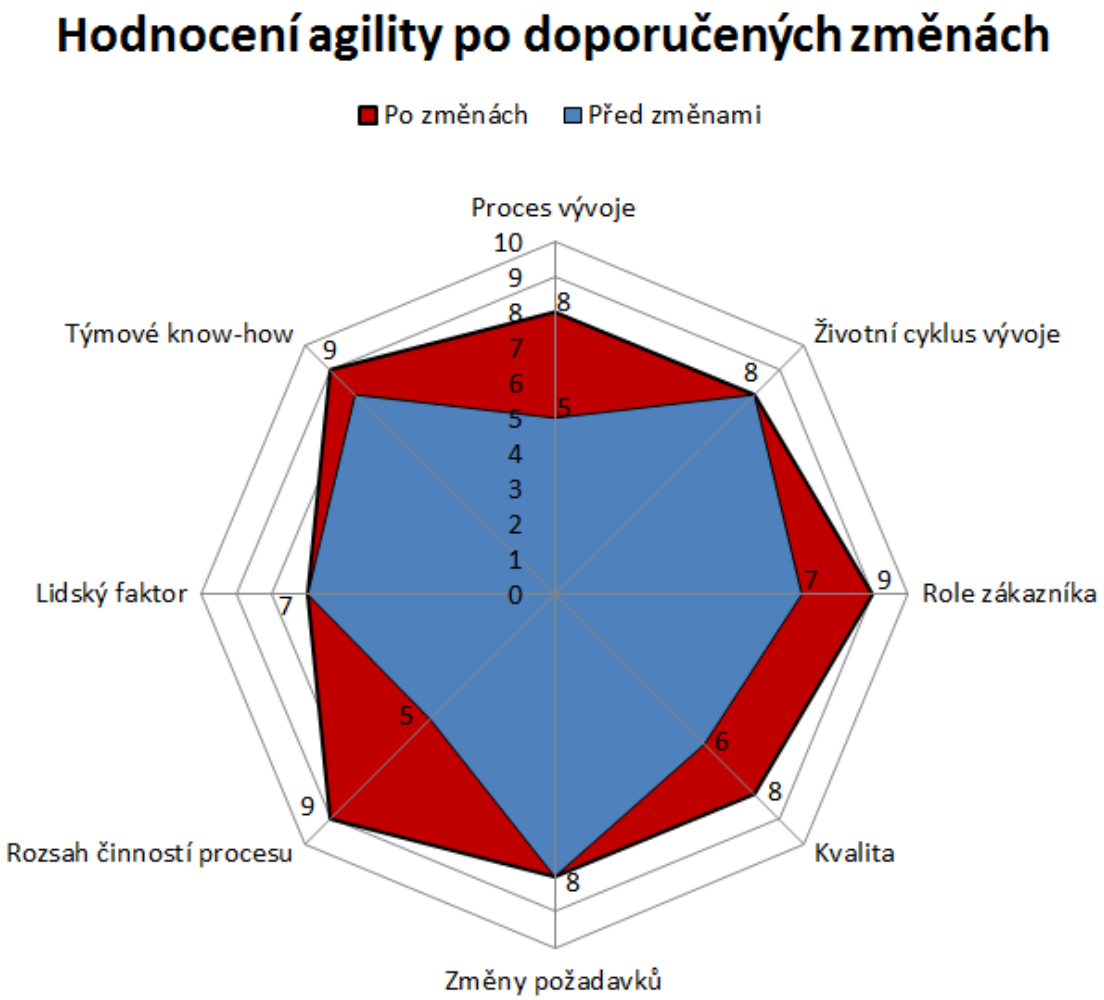
Tab. 17 Celkový předpokládaný dopad při zavedení všech změn, zdroj: vlastní

Ukazatel	**Hodnota**
Investiční finanční náklady	83.775 EUR
Provozní náklady	266.276,25 EUR / měsíc
Provozní přínosy	450.640,00 EUR / měsíc
Celkový provozní dopad	**úspora ve výši 186.363,75 EUR / měsíc**
Nefinanční přínosy	Produktivita vývoje: + 12% Doba time to market: - 20% Spokojenost zákazníků: + 10% Změny v průběhu vývoje: - 10% Kvalita - chyby při vývoji: - 13% Šíření know-how vývojářů: +5% Informovanost prod. ownera: + 20% Eliminace duplicity dat požadavků Rozšíření možností reportování

Následující tabulka a graf uvádí předpokládaný dopad na agilitu vývoje, v případě zavedení všech doporučených změn.

Tab. 18 Hodnocení očekávané agility po zavedení změn, zdroj: vlastní

Oblast	**Popis situace**	**Hodnocení agility před změnami (1 – 10)**	**Hodnocení agility po změnách (1 – 10)**
Proces vývoje	Proces vývoje je neustále upravován podle aktuálních potřeb.	5	8
Životní cyklus vývoje	Plánování v iteracích. Inkrementy produktu však nejsou dodávány po sprintu.	8	8
Role zákazníka	Product owner má úplnou kontrolu nad backlogem a má detailní přehled o průběhu sprintu i inkrementu pro release.	7	9
Kvalita	Vývojový tým pružně reaguje na aktuální situaci i v průběhu sprintu, čímž je maximalizován přínos pro zákazníka.	6	8
Změny požadavků	Změny jsou většinou přijímány, a to bez nutnosti řízení změn.	8	8
Rozsah činností procesu	Všechny možné činnosti jsou automatizovány pomocí vhodně nastavené aplikace. Denní setkání jsou vedené efektivně.	5	9
Lidský faktor	Důraz na dokumentaci. Lidský faktor a zkušenosti jedinců jsou využívány.	7	7
Týmové know-how	Složitější problémy jsou řešeny kolektivně, informace jsou v týmu aktivně šířeny.	8	9



Obr. 24 Graf hodnocení očekávané agility po zavedení změn, zdroj: vlastní

5 ZÁVĚR

Záměrem této práce bylo vytvořit teoretický souhrn podstatných informací o tradičních a agilních metodikách řízení projektů a vývoje, dále analyzovat současný způsob využití metodiky Scrum u zkoumaného vývojového týmu a v neposlední ředě také uvést doporučení ke změnám, které by měly vést ke zlepšení efektivity vývoje. S ohledem na zaměření této práce je největší část teoretické části věnována podrobnému popisu metodiky Scrum development process. Část práce je také zaměřena na podpůrné aplikace, které jsou využívány pro plánování a řízení vývoje Scrumem. Proto byl v práci uveden i stručný přehled vybraných podpůrných nástrojů, ať už ve formě fyzické nástěnky, nebo prostřednictvím softwarových aplikací, placených i freeware.

Analýzou současného stavu byly identifikovány problematické oblasti, současného způsobu fungování metodikou Scrum. K identifikaci těchto problémových oblastí byly využity teoretické poznatky, zmíněné v teoretické části práce. Pro tyto nedostatky byly v následující části doporučeny vhodné změny, ať už v oblasti organizace, nastavení procesu, nebo využití podpůrných softwarových nástrojů. Poslední kapitola pak rozebírá ekonomický dopad každého navrženého řešení, kde jsou identifikovány změny finančních i nefinančních ukazatelů, způsobené zavedením jednotlivých doporučení.

Geografické rozmístění Scrum týmu v současné době znesnadňuje vzájemnou komunikaci, zejména absencí osobního kontaktu a mírnou jazykovou bariérou. Dlouhodobě nejvhodnějším řešením se zde jeví přesunu vývojového týmu z Belgie do Brněnského centra, kde momentálně sídlí tým product ownerů pro dané aplikace. Přínos je zde nejen oblasti nákladů, ale také ve zlepšení komunikace v rámci Scrum týmu.

Současná funkce rolí Scrum týmu se jeví efektivní, změna byla však doporučena v oblasti řízení produktového backlogu, kde byl identifikován problém v tom, že tento backlog mohl být upravován i jinými týmy, než jen product ownerem. Doporučení je posílení role product ownera, který by měl být jediným oprávněným k provádění změn v produktovém backlogu. Jedině tak je možné produktový backlog efektivně řídit, aby bylo možné na jeho základě plánovat vývoj v následujících sprintech.

Problém byl identifikován také v oblasti zvolených softwarových aplikací, které jsou využívány. Zkoumané týmy totiž využívají několik různých aplikací s omezenými

funkcemi. V některých jsou data dokonce vedena duplicitně, což zvyšuje množství administrativy a riziko chyb. Zde bylo na základě analýzy aplikací pro Scrum doporučeno nahradit tři využívané aplikace softwarem zvaným Jira Confluence, který umožňuje jak řízení metodikou scrum, tak vedení evidence defektů, incidentů a požadavků. Tím je možno vést všechna data v jedné aplikaci, což umožní rozšíření možností reportování a také snížení množství administrativy.

V oblasti denního setkání sprintu hlavní problém v oblasti příliš volného průběhu tohoto setkání, které pak trvá nepřiměřenou dobu. Důraz je kladen na moderování, což zefektivní využití času a také šíření využitelných informací v rámci týmu. Občasná účast product ownera na denních setkáních sprintu může urychlit rozhodování a poskytnout aktuální informace oběma stranám. Průběh vývoje v rámci sprintu je sledován ukazatelem burndown. Rozdíl v načasování sprintů a releasů inkrementů produktu však identifikuje potřebu k využití dalšího nástroje pro sledování vývoje celého inkrementu. Doporučen byl ukazatel burnup. Konec každého sprintu by měl být následován přezkoumáním sprintu a retrospektivou. Tyto zpětné pohledy slouží k přehodnocení nastaveného procesu vývoje a dodaných inkrementů. Zavedení těchto setkání je důležité k neustálému zdokonalování nastaveného procesu, odstranění nepotřebných činností a předcházení chybám při vývoji.

Uvedené tabulky a grafy hodnotí současný stav agilního přístupu v rámci zkoumaného vývojového týmu a také očekávaný stav v případě, že budou zavedeny všechny změny, doporučené v této práci.

Doporučení plynoucí z této práce tedy je, zavést všechny navržené řešení, jelikož u všech byl identifikován kladný ekonomický dopad. Zavedení těchto změn má však dopad nejen ekonomický, ale zvýší celkovou agilitu vývojového týmu, která povede ke zvýšení kvality a flexibility vývoje. Tyto ukazatele se promítnou také do spokojenosti zákazníků, což je základním faktorem úspěchu podnikání.

6 SEZNAM POUŽITÝCH ZDROJŮ

1. 10006, ISO. Quality management systems: Guidelines for quality management in projects. Ženeva, Švýcarsko : International Organization for Standardization, 2003.

2. *Manifest Agilního vývoje software.* [Online] 2011. [Citace: 11. Leden 2015.] http://www.agilemanifesto.org/iso/cs/.

3. Sommervile, Ian. *Software engineering.* 8th ed. New York : Addison-Wesley, 2007. 978-0-321-31379-9.

4. Vývojové modely. *Diagnostika a testování elektronických systémů.* [Online] Umel, VUT Brno, 2012. [Citace: 12. Leden 2015.] http://www.umel.feec.vutbr.cz/bdts/index.php/embedded-systemy/vyvojove-modely.

5. Vondrák, Ivo. *Úvod do softwarového inženýrství.* Ostrava : VŠB – Technická univerzita, 2002.

6. Principy stojící za Agilním Manifestem. *Manifest Agilního vývoje software.* [Online] 2011. [Citace: 11. Leden 2015.] http://agilemanifesto.org/iso/cs/principles.html.

7. Rubin, Kenneth S. *Essential Scrum: a practical guide to the most popular agile process.* Upper Saddle River, NJ : Addison-Wesley, 2012. 978-0-13-704329-3.

8. What is extreme programming. *XProgramming.* [Online] 2015. [Citace: 15. Leden 2015.] http://xprogramming.com/what-is-extreme-programming/.

9. The Agile Umbrella. *Feature Driven Development.* [Online] 3. Červen 2003. [Citace: 15. Leden 2015.] http://www.featuredrivendevelopment.com/node/531.

10. Introduction to Test Driven Development. *AgileData.* [Online] 2013. [Citace: 15. Leden 2015.] http://agiledata.org/essays/tdd.html.

11. *Scrum Guides.* [Online] 2014. [Citace: 15. Leden 2015.] http://www.scrumguides.org/scrum-guide.html.

12. Don't Control Agile Projects. *Jim Highsmith.* [Online] 20. Březen 2014. [Citace: 15. Leden 2015.] http://jimhighsmith.com/dont-try-control-agile-projects/.

13. Crystal methodologies. *Alistair Cockburn.* [Online] 19. Červen 2008. [Citace: 15. Leden 2015.] http://alistair.cockburn.us/Crystal+methodologies.

14. Lean-Agile Roadmap for Achieving Enterprise Agility. *NetObjectives.* [Online] 2013. [Citace: 15. Leden 2015.] http://www.netobjectives.com/resources/lean-agile-roadmap.

15. How Startups Use Hard Stops and Agile Management. *Entrepreneur Ideas.* [Online] 2015. [Citace: 13. Leden 2015.] http://www.entrepreneur-ideas.org/startups-use-hard-stops-agile-management/.

16. Buchalcevová, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky.* Praha : Grada, 2005. 80-247-1075-7.

17. Vizdos, Michael. The Classic Story of the Pig and Chicken. *Implementing Scrum.* [Online] 11. Září 2006. [Citace: 7. Březen 2015.] http://www.implementingscrum.com/2006/09/11/the-classic-story-of-the-pig-and-chicken/.

18. Informace o skupině KBC. *ČSOB.* [Online] duben 2015. [Citace: 18. duben 2015.] http://www.csob.cz/cz/csob/Servis-pro-media/Stranky/O-Skupine-KBC.aspx.

19. Pham, Andrew and Pham, Phuong-Van. *Scrum in action: Agile software project management and development.* Boston, MA : Course Technology, 2012. 978-1-4354-5913-7.

20. Doležal, Jan, Máchal, Pavel a Lacko, Branislav. *Projektový management podle IPMA.* 2., aktualiz. a dopl. vyd. Praha : Grada, 2012. 978-80-247-4275-5.

21. Kadlec, Václav. *Agilní programování: metodiky efektivního vývoje softwaru.* Brno : Computer Press, 2004. 9788025103425.

22. Procházka, Jaroslav a Klimeš, Cyril. *Provozujte IT jinak: agilní a štíhlý provoz, podpora a údržba informačních systémů a IT služeb.* Praha : Grada, 2011. 978-80-247-4137-6.

7 SEZNAM OBRÁZKŮ

8 SEZNAM TABULEK