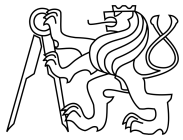
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Bimodal IT jako způsob zavádění agility v korporacích
Student:	Bc. Vladimír Šimon
Vedoucí:	Ing. Pavel Krejčí
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Analyzujte koncept Bimodal IT, agilní přístupy řízení projektů a tradiční způsoby řízení IT projektů. Formou rešerše diskutujte výhody a nevýhody spojené s těmito způsoby řízení a porovnejte jejich vhodnost pro velké organizace. Analyzujte způsob řízení IT projektů ve vybrané organizaci (po dohodě s vedoucím práce) a zhodnoťte vliv řízení projektů na spolupráci mezi IT a byznysem. Navrhněte způsob zavedení agility do vybrané organizace, a to včetně potřebných organizačních změn a pravidel řízení. Stanovte potřebné vlastnosti a kompetence členů projektových týmů, aby bylo možné plně využít přínosu agilních přístupů. Analyzujte benefity a rizika spojená se zavedením agility v dané organizaci.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 6. října 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
katedra softwarového inženýrství

Diplomová práce

Bimodal IT jako způsob zavádění agilního řízení do korporací

Bc. Vladimír Šimon

Vedoucí práce: Ing. Pavel Krejčí

9. ledna 2017

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Pavlovi Krejčímu za příkladné vedení a přínosné rady. Dále pak oponentovi Ing. Janovi Malému též za jeho konstruktivní kritiku a dobré připomínky k práci. Dále bych rád poděkoval Ivanovi Mouchovi za připomínkování DP. Všem pak za podporu na schůzkách praktické části DP.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. ledna 2017 .....

Odkaz na tuto práci

Šimon, Vladimír. *Bimodal IT jako způsob zavádění agilního řízení do korporací*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

V oblasti startupů je agilní řízení přirozeně rozšířené. Ve větších firmách a korporacích je situace jiná. Historicky zde fungují složité vodopádové metodiky spjaté s mnoha pravidly a omezeními danými buď regulátory nebo složitými, striktně definovanými procesy. Ne všechny projekty takové velké společnosti se dají řídit agilně. Práce je založena na konceptu Bimodal IT definovaném společností Gartner a čtenář se v ní dozví, jak skloubit nové agilní přístupy s klasickou vodopádovou metodikou řízení projektů.

Klíčová slova agilní metodiky, vodopádové metodiky, Scrum, Large Scalled Scrum, Scaled Agile framework,Bimodal IT, případové studie, firemní kultura, řízení změn v organizaci

Abstract

In the field of startups is agile management a natural part of them. In larger companies and corporations, the situation is different. Historically, they use complex waterfall methodology associated with many rules and constraints imposed by either regulators or the company itself. Not all areas thus managed large companies can be managed in agile way. The work is based on the concept

of Bimodal IT defined by Gartner and reader in it can learn how to combine new agile approaches with traditional waterfall project management structure.

Keywords Agile methodologies, Waterfall methodologies, Scrum, Large Scalled Scrum, Scaled Agile framework, Bimodal IT, Case study, Corporate culture,Change managment

Obsah

Úvod **1**

1 Teoretická část **3**
1.1 Základní pojmy 3
1.2 Agilní metodiky 4
1.3 Klasické vodopádové metodiky 20
1.4 Porovnání klasických vodopádových metodik a agilních metodik 23
1.5 Bimodal IT podle Gartner 25
1.6 Případové studie - zavádění agility do podobně velkých společností 27
1.7 Organizační změny a jejich zavádění 29

2 Případová studie a návrh zavedení agilních metodik do velké české banky **35**
2.1 Způsob práce na praktické části 35
2.2 Popis zkoumané organizace 35
2.3 Bimodal IT jako způsob zavádění agility 42
2.4 Předpoklady nutné k tomu, aby celý koncept fungoval 54
2.5 Akční plán a kroky nutné k zavedení Bimodal IT 59

Závěr **65**

Literatura **67**

A Seznam použitých zkratek **69**

B Obsah přiloženého CD **71**

C Popis programu manažerské kalkulačky pro výpočet nákladů na projekt **73**

D Dotazník o stavu agility **75**

Seznam obrázků

0.1 Struktura diplomové práce graficky. 2

1.1 Přehled metodiky Scrum. [1] [str. 18 obr. 2.1] 7
1.2 Vzhled Scrum boardu. 8
1.3 Burndown chart. Graf používaný ke kontrole rozpracovanosti úkolů ve sprintu. 9
1.4 Pohled na životní cyklus Scrumu podle [20] 11
1.5 Rozdíly metodiky Scrum a metodiky XP. Založené na [2] 15
1.6 Přehled metodiky Scalled agile framework [3] 17
1.7 Přehled metodiky Large scalled scrum [4] 18
1.8 Přehled pracnosti v jednotlivých fázích a iteracích UP a RUP. Převzato z [5] 24
1.9 Přehled celé maticové struktury Spotify. Vychází z obrázku v [6] . 28
1.10 Křivka fází přijímání změny. 30
1.11 Proces realizace změny 31
1.12 Jak často jsou využívány funkce software. Obrázek převzat z [7] . 32
1.13 Co znamená hodnota v různých oblastech softwarového vývoje [8][str. 76] 33

2.1 Organizační struktura banky 37
2.2 Hierarchie a vazby mezi základními prvky ve vývoji nového software. 38
2.3 Vrstvy řízení projektů v bance 39
2.4 Základní struktura konceptu Bimodal IT. 42
2.5 Agilní řízení pro fáze vodopádového projektu versus plná agilita . . 43
2.6 Řízení projektu kombinací módů 46
2.7 Parametry, podle kterých se bude posuzovat, jakým způsobem projekt řídit 47
2.8 Přehled konceptu Nástrojů pro kontinuální integraci. Zdroje obrázků pro konkrétní technologie: [9, 10, 11, 12, 13, 14, 15] 56
2.9 Popis změn ve zkoumané bance rozdělený podle [16]. 60

2.10 Zapojení byznysu do groomingu 63

C.1 Aktuální vzhled manažerské kalkulačky na výpočet nákladů na projekt 74

D.1 Agilní dotazník: Otázka 1 75
D.2 Agilní dotazník: Otázka 2 76
D.3 Agilní dotazník: Otázka 3 76
D.4 Agilní dotazník: Otázka 4 77
D.5 Agilní dotazník: Otázka 5 77
D.6 Agilní dotazník: Otázka 6 78
D.7 Agilní dotazník: Otázka 7 78
D.8 Agilní dotazník: Otázka 8 79
D.9 Agilní dotazník: Otázka 9 79

Seznam tabulek

1.1 Porovnání mode 1 a 2 26

2.1 Míra pravděpodobnosti rizik 51
2.2 Dopady rizik na koncept 52
2.3 Výpočet úrovně rizik 52
2.4 Vypočtená úroveň rizik 53

Úvod

Cílem práce je definovat nový způsob řízení projektů v korporaci založený na agilním řízení. Proč zavádět agilní řízení do firmy? Jaká jsou jeho specifika? Jaké problémy mohou nastat při komunikaci mezi rozdílně řízenými projekty? Jak jednotlivé agilní metodiky fungují a proč? A co by mohlo přinést jejich propojení založené na hluboké znalosti fungování korporátního prostředí? Tyto otázky hráli hlavní roli při výběru tématu a diplomová práce na ně bude hledat odpovědi.

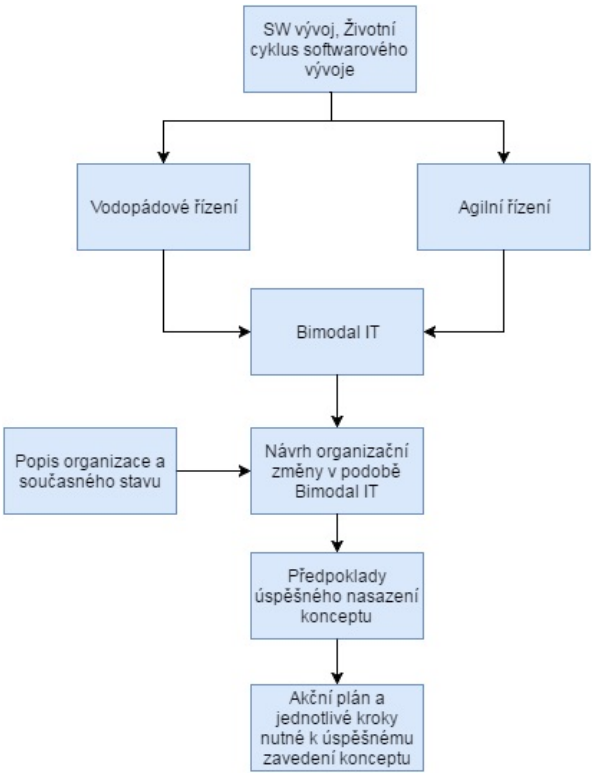
Klíčovými výhodami typickými pro agilní řízení jsou zvýšení kvality výsledného softwaru, redukce počtu chyb, snížení nákladů a urychlení dodávek systémových komponent na trh. Vzhledem ke komplexnosti velkých projektů v bance ale nelze všechny projekty řídit agilně. Proto přišla velká poradenská společnost Gartner s řešením jménem Bimodal IT, které bude v práci detailně rozebráno a navrženo reálné řešení pro vybranou společnost.

Základem konceptu Bimodal IT je rozdělení projektů dle jasně definovaných pravidel, na projekty řízené agilně a projekty řízené tradiční vodopádovou metodikou. Na oba typy řízení se hodí jinak zaměření pracovníci. Jelikož oba způsoby vyžadují odlišný přístup, je nutné zajistit, aby nenastaly konflikty, proto zřizujeme kancelář CIO nadřízenou oběma typům projektů. Nastane-li konflikt, závisí rozhodnutí na ní.

Teoretická část mapuje současné pole agilních i vodopádových metodik, upozorňuje na jejich klady a zápory a zkoumá, co mají obě metodiky společného. Dále porovnává oba typy metodik ve více ohledech, pozoruje na případových studiích způsob zavádění agilního řízení do korporací a definuje obecné koncepty použitelné při organizačních změnách, jakou je Bimodal IT. V závěru definuje způsob jak oba typy metodik skloubit v konceptu Bimodal IT podle doporučení Gartner. Teoretická část je zásadní pro pochopení fungování agil-

ních i tradičních metodik. Většina jejich charakteristik vychází z jejich popisu a způsobu práce. Agilní metodiky ale často nejsou o konkrétním postupu, ale o změně přemýšlení nad softwarovým vývojem.

Praktická část Po více než ročních konzultacích popisuje možnost reálně zavést koncept Bimodal IT do zkoumané organizace. Nejdříve čtenáře seznamuje se zkoumanou organizací pro představu o rozměrech změny, pokud by se měla zavést do celé společnosti. Dále pak určuje nutné předpoklady, aby celý koncept mohl existovat. Následně dává na misky vah přínosy konceptu po zavedení a rizika. V poslední části je popsán již započatý plán zavedení konceptu do prostředí korporace.



Obrázek 0.1: Struktura diplomové práce graficky.

Kapitola 1

Teoretická část

1.1 Základní pojmy

1.1.1 Byznys

Často se v práci budeme setkávat s pojmem byznys, aby nevznikal nesoulad v použitých definicích pro potřeby práce se určí podle. [17]. „Byznysem se pro potřeby práce rozumí jakákoliv podniková struktura mimo IT oddělení. Tento pojem se na úrovni jiných částí podniku pro pojmenování ostatních příliš nepoužívá, nicméně ve světě podnikového IT se jedná o ustálený výraz.“[17]

1.1.2 Životní cyklus softwarového vývoje

Pro vývoj každého softwarového díla se ustálil shodný postup, který musí mít nejméně tyto 4 fáze:

1. Definování prvotního konceptu
2. Návrh/design aplikace
3. Implementaci
4. Testování

Každá z těchto fází musí alespoň jednou proběhnout. Pokud je délka fází v řádů měsíců a realizují se striktně sekvenčně, říkáme o takovém vývoji, že je lineární nebo také vodopádový. Pokud jsou fáze kratší a cyklus se vícekrát opakuje, mluvíme o iterativním, respektive inkrementálním přístupu.

Definování prvotního konceptu Na začátku vývoje se definuje k čemu bude finální produkt sloužit. V krajních případech se může vývojář rozhodnout neimplementovat systém, který bude ovládat jadernou pumu nebo sloužit ruské kontrarozvědce.

Návrh/design aplikace Před samotným psaním kódu se navrhne architektura řešení a určí se základní stavební prvky. V této fázi se při prvním průběhu určí technologie používaná k implementaci.

Implementace Když je jasné proč se systém vyvíjí, jaké budou použité technologie a návrh řešení, přistoupí se k samotnému psaní kódu.

Testování Psaní kódu je komplexní problém. Na první pohled se může zdát, že je kód funkční, ale nemusí fungovat ve všech případech. Jeho funkčnost je vhodné v nejvyšší míře pokrýt testy. V současnosti se nejčastěji používají automatické testy, ty nemusí spouštět vývojář, ale spouští je stroj při každém zásadnějším přírůstku kódu.

1.1.3 Projekt a projektová metodika

V předchozí kapitole byly definovány kroky nutné k vývoji softwarového produktu. To ale není vše, chybí určit, kdo stanoví zadání, kdo bude tým řídit, jak se budou řešit konflikty s ostatními vývojáři v jedné společnosti a kdo bude dohlížet na bezproblémový provoz po implementaci. S rozvojem IT a současně s růstem počtu aplikací a složitosti ekosystému IT uvnitř velkých společností, vznikla potřeba dát jasné odpovědi na výše položené otázky. Liniový management nestačil implementovat příchozí změny. Bylo nutné využít nově příchozích konceptů, konkrétně projektového řízení. Organizace se změnily na projektově řízené a to dalo vzniknout projektům a projektovým manažerům, kteří je řídí. Aby byla jasná pravidla řízení projektu, aby měl co nejnižší náklady a aby se dosáhlo vysoce kvalitního řešení v nejkratším čase, definovala se projektová metodika určující kompetence všech účastníků projektu, konkrétní kroky realizované v každé fázi projektu a dokumenty, které musí popisovat průběh práce na projektu. Projektové metodiky mohou být dvojího typu. Rozdělení je podle toho, který přístup ve vývoji software se používá. Pokud se používá iterativní přístup a metodiky splňují další podmínky definované v 1.2.1 nazýváme je agilní. Pokud se využívá lineární přístup, mluvíme o tradičních nebo také vodopádových metodikách.

1.2 Agilní metodiky

Agilní řízení představuje množinu postupů a metodik, které pomáhají týmu přemýšlet a pracovat efektivněji, dělat lepší rozhodnutí. Tyto postupy a metodiky zasahují do všech oblastí klasického vývoje software včetně projektového managementu, návrhu software, architektury i zlepšování procesů. Spolu s postupy jsou optimalizovány i způsoby pro rychlou adopci agilních principů. Agilita je hlavně o nastavení v hlavách jejích uživatelů. Dobré nastavení zásadně napomáhá úspěchu agilních metodik. Vývojáři musí fungovat jako jeden tým,

každý musí spolurozhodovat o krocích provedených v týmu. [2] Vzory agilních principů jsou definovány v Agilním manifestu 1.2.1.4.

1.2.1 Manifest Agilního vývoje software

Agilní manifest je dostupný online.[18] Jeho znění je:

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

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

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

Každá metodika, která splňuje tyto obecné body se dá považovat za agilní. Pojďme si teď blíže rozebrat body Agilního manifestu, abychom jasněji pochopili, co koncepty obecně popisují a jak bychom si to v praxi mohli představit.(Inspirováno [19])

1.2.1.1 Jednotlivci a interakce před procesy a nástroji

Nástroje podporující SW vývoj nejsou samospásné. „Pojďme se podívat na dva odlišné týmy. V prvním máme 10 profesionálů, kteří si vytvořili vlastní nástroje a pracují v silně kolaborativním prostředí, ve druhém máme 10 vývojářů, kteří pracují v přesně definovaném procesním prostředí s nejlepšími nakoupenými nástroji, ale bez interakcí a spolupráce.“[19]. Ve kterém týmu byste radši pracovali? Který tým bude efektivněji spolupracovat na zadaném úkolu? Je zřejmé, že to bude první tým. Práce něm je srovnatelná s spěšnými startupy, kde probíhá komunikace přímočaře. Procesy jsou dobrá věc, ale musí být nastaveny tak, aby dávaly prostor dostatečné míře autonomie týmu.

1.2.1.2 Fungující software před vyčerpávající dokumentací

Když jste firma a zadali jste si do výroby nový software, na prvním místě vás zajímá, jakou má dokumnetaci nebo byste raději chtěli vidět demo hotového produktu? Nenechte se zmást, mezi agilní principy rozhodně nepatří nepsat dokumentaci, ale pouze nejdříve psát samotný software a poté dodat i dokumentaci. Často se totiž stává, že dodavatel software se snaží udělat dojem detailním popisem všech funkcí systému, které ale nakonec vůbec nedodá. Odpověď na původní otázku je jasná, nejdříve výsledky, průběžně pak dodávat základní dokumentaci.

1.2.1.3 Spolupráce se zákazníkem před vyjednáváním o smlouvě

Motivací vývojáře k vytváření produktu je následný prodej zákazníkovi. Pokud chceme vyvíjet software, který má zákazník rád a využívá maximum jeho funkcí, musíme s ním komunikovat. Problém je, že „zákazník mnohdy neví, co vlastně chce, a často mění názor, ale dlouhodobou spoluprací můžeme zákazníka „vychovat“ a společně vytvořit spolupracující tým, který zajistí jak úspěch náš, tak i úspěch zákazníka.“[19] Příkladem nekvalitně komunikujícího dodavatele je takový, který se vždy odkazuje na podepsanou smlouvu místo toho, aby poslouchal zákazníka a reagoval na jeho požadavky. Je jasné, že se požadavky na systém v čase mění a při dodržení úzké spolupráce se zákazníkem je možné na změny aktivně reagovat. „Spokojený zákazník, který dostal to, co potřebuje, vás doporučí dalším firmám, zatímco zákazník, který sice má to, co si objednal a ve smlouvě podepsal, ale nijak mu to v jeho misi nepomohlo, se příště podívá po jiném dodavateli.“ [19]

1.2.1.4 Reagování na změny před dodržováním plánu

Žijeme v době, kdy se situace mění z hodiny na hodinu. Konkurence se zvyšuje a objem práce, který jsme stihli udělat za dnešek může být zítra dvojnásobný. Neustále se zjednodušuje zavádění změn do softwarových produktů a musíme být schopni zavádět změny i na konci projektu. „ Představte si, že zákazník přijde se změnou v posledních fázích projektu a ta změna bude důležitá pro jeho přežití na trhu.“ [19] Jaký vliv asi bude mít na zákazníka, pokud mu řekneme, že jsme si to nedomluvili a že jestli má zájem, dodáme mu požadované funkce po dokončení původní domluvy. Pravděpodobně si u něj neuděláme dobré jméno, jelikož nebudou uspokojeny jeho požadavky a celé dílo pro něj začne ztrácet na významu.

1.2.2 Scrum

Scrum je nejznámější a v současnosti nejpoužívanější agilní metodika, která je nejvíce popsána a pokrývá nejširší oblast s ohledem na projektový management. Jako jedna z mála by se tak dala využít k řízení celého projektu bez nutnosti stanovení dalších pravidel a hlavně rolí mimo ni. V dalších sekcích budou popsány nejdříve role a artefakty(tabule, grafy,...) používané ve Scrumu a poté bude osvětlen samotný průběh řízení softwarového vývoje pomocí Scrumu. Následující popis je založen na knize Essential Scrum. [1]

Původ Scrumu První zmínka o Scrumu se objevila v roce 1986 v Harvard Business Review v článku "The New New Product Development Game"(Takeuchi a Nonaka 1986). Tento článek popisuje, jak společnosti jako Honda, Canon, Fuji-Xerox produkovaly vysoce efektivní výsledky pomocí škálovatelného přístupu, založeném na týmovém „dělej-vše-jen-jednou“ produktovém rozvoji.

Obrázek 1.1: Přehled metodiky Scrum. [1] [str. 18 obr. 2.1]

To také podtrhuje význam, který je kladen na samoorganizující se týmy, a nastiňuje úlohu managementu v procesu vývoje. Článek byl zajímavý ve spojení s množstvím konceptů, které daly vzniknout tomu, co dnes nazýváme Scrum. Scrum není zkratka, ale spíše termín využívaný ve sportu rugby, který se odkazuje na způsob restartování hry po náhodném přerušení, nebo v případě, že míč šel do autu. Dokonce i když nejste ragbyový fanoušek, pravděpodobně jste viděli skrumáž, kde tyto dvě sady forwardů uskupeni do tvaru koule s uzamčenými rameny a s hlavami dolů, bojují o získání míče. Takeuchi a Nonaka používají metafory z rugby na popsání vývoje produktů. Ačkoli se Scrum nejčastěji používá k rozvoji softwarových produktů, jádro hodnoty a principy Scrumu mohou a jsou používány k rozvoji různých typů výrobků nebo organizaci směru různých druhů práce, například vývoji hardwaru, marketingových programů a sales.

1.2.2.1 Role ve Scrumu

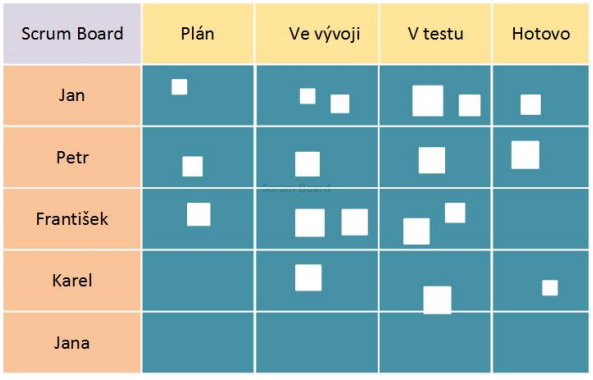
Scrum Master pomáhá všem zapojeným do softwarového vývoje pochopit a osvojit si hodnoty Scrumu, principy a osvědčené postupy. Funguje jako mentor, poskytuje procesní řízení a pomáhá týmu i organizaci provozovat jejich vlastní vysoce výkonný Scrum užívaný ve shodě s principy organizace. Na druhé straně pomáhá i organizaci projít změnami, které mohou nastat při přechodu na Scrum. Scrum Master z dálky pomáhá týmu vyřešit problémy a kontinuálně zlepšovat svou práci. Funguje jako odpovědná osoba za komunikaci s okolním prostředím v organizaci i mimo ni, dále pak odstraňuje požadavky na tým, které by mohly být problémem pro jeho výkon. Má tým motivovat, směrovat, diskutovat s ním, ale není jeho účelem ho direktivně řídit a rozdělovat práci. Není to manažer, je to leader.

Product Owner je středobodem vývoje produktu. Je jedinou autoritou rozhodující o tom, které řásti nebo funkcionality se budou implementovat a v jakém pořadí. Product Owner musí často a pravidelně komunikovat s vývojovým týmem a musí mu poskytnout jasnou vizi čeho má tým dosáhnout. Je zodpovědný za obecný úspěch řešení, které je vytvářeno a bude provozováno. Je jedno, zda se jedná o externí produkt nebo interně vyvíjený systém, PO musí mít neustále jistotu, že se pracuje na nejhodnotnějších částech produktu jak pro společnost, tak pro Scrum tým.

Členové týmu Tradiční vývoj software obsahuje nespočet rolí a pracovních pozic. Scrum toto rozdělení nemá, všichni členové týmu jsou univerzálně schopní vyvíjet, testovat, navrhovat a provozovat vytvářený produkt. Tým se sám organizuje ke zjištění nejlepší cesty ke splnění cíle, který nastavuje Product Owner. Tým by se měl skládat z pěti až devíti členů, pokud je potřeba více týmů řízených společně při práci na větším produktu, je možné využít škálování agility, které si popíšeme v dalších částech této práce.

1.2.2.2 Artefakty Scrumu

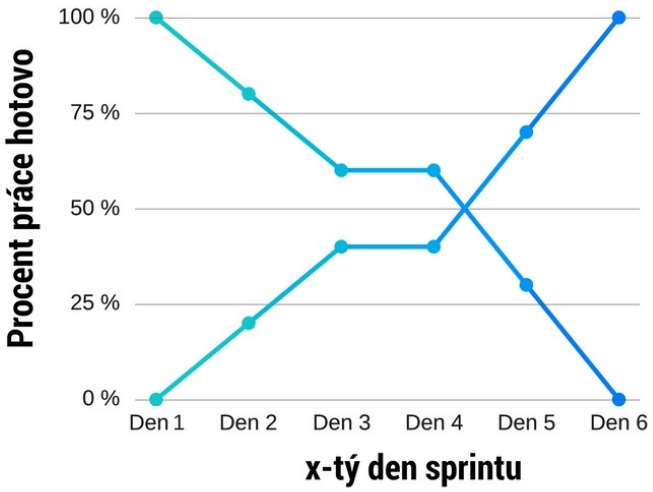
Scrum board Je to velká tabule v místnosti, kde pracuje tým. Představuje jednoduchý ale účelný nástroj, ke komunikaci průběhu prací na sprintu bez zbytečných dotazů. Na Scrum Board se přilepují papírky. Na jednom papírku je jeden úkol a jeho pracnost a postupně se na něj dělají čárky a tím je vidět, kolik času na něm vývojář strávil. Všechny úkoly začínají svou cestu v kolonce plánováno. Když se na úkolu začne pracovat, přesune se do kolonky v průběhu, a když se dokončí, přesune se k testování a následně do části hotovo.



Obrázek 1.2: Vzhled Scrum boardu.

Burndown chart je graf. Každý den během sprintu členové týmu upravují kolik práce jim zůstává na úkolech, které si do něj naplánovali. Burndown chart pak zobrazuje, kolik z celkové práce zbývá v daném momentu udělat. Na

začátku sprintu je sto procent práce nekompletních, postupně podíl nedokončené práce klesá. Na grafu jsou vidět historická data, kolik práce se ve který den udělalo a dá se z něj okamžitě zjistit, pokud například v jeden den ubyla polovina práce, že se zákonitě muselo něco stát, např. byla zrušena práce na velkém projektu.



Obrázek 1.3: Burndown chart. Graf používaný ke kontrole rozpracovanosti úkolů ve sprintu.

Product Backlog Základní princip Scrumu je, že implementujeme vždy nejdůležitější úkoly jako první. Product Owner rozhoduje, které úkoly jsou prioritní a řadí je do seznamu, kterému říkáme Product Backlog. V ideálním případě Product Owner i definuje problémy nebo vylepšení, které chce implementovat. U každého úkolu musí být uvedena jeho velikost. Při prioritizaci se poměřuje, zda úkol přináší dostatečnou hodnotu jako protiváhu času, který na něm vývojář stráví.

1.2.2.3 Schůzky ve Scrumu

Popsané schůzky se využívají ve Scrumu, aby se tým pravidelně scházel a koordinoval. Ostatní schůzky mimo zde uvedených, by se tím měly minimalizovat. Každá ze schůzek má doporučenou maximální délku. Stejně jako délka sprintu, která se nedá měnit.

Sprint planning Během této schůzky se vybírají části SW produktu, které budou realizovány v příštím sprintu. Maximální délka schůzky je osm hodin. Sprint planning odpovídá na dvě otázky:

- Co bude vytvořeno, co má být dodáno v příštím sprintu?
- Jak toho dosáhnout (kdo bude aktivity realizovat)?

Vývojový tým se sám organizuje a rozděluje si práci. První se musí u každého úkolu odhadnout pracnost a následně se domluvit, kdo bude úkol realizovat. Product Owner pomáhá prioritizovat věci v Sprint Backlogu. Definuje se cíl, jehož se má na konci sprintu dosáhnout. Výhoda je, že sprint je tak krátký, že se členům týmu mnohem lépe plánuje, než kdyby museli plánovat např. rok nebo dva dopředu.

Daily standup Maximálně 15 minut trvající schůzka, na které se plánuje na 24 hodin dopředu. Koná se každý den a každý člen týmu během ní sdělí celému týmu 3 věci:

- Co člen týmu udělal minulý den
- Co bude dělat dnes
- Jaké jsou překážky v jeho práci a zda by mu někdo mohl pomoci je překonat

Daily standup je velice důležitá krátká schůzka, která zásadně napomáhá na denní bázi koordinovat tým a neustále se ujišťuje, že bez problémů směřuje ke splnění cílů aktuálního sprintu. Scrum Master se zapojuje hlavně do třetího bodu. Pokud se někdo v práci dostane do slepé uličky Scrum Master pomáhá najít někoho, kdo by mu mohl s problémem pomoci.

Sprint Review Provádí se na konci sprintu. Scrum tým a zástupci byznysu procházejí, co bylo uděláno během sprintu a hodnotí, zda se podařilo splnit jejich očekávání. Schůzka trvá při měsíčním sprintu maximálně 4 hodiny. Sprint Review by mělo obsahovat:

- Product Owner představí hotové věci ve Sprint Backlogu
- Tým mluví o tom, co šlo dobře a kde byly problémy
- Tým předvádí implementované věci
- Diskuze o dalším směřování
- Diskuze, kam by se produkt mohl posunout
- Diskuze ohledně dalších informací – budget, plán, změny...

Na konci Sprint Review se vyčistí Sprint Backlog a připraví se na další sprint.

Retrospektiva Je to čas, který může tým využít k tomu, aby zrekapituloval silné a slabé stránky předchozího sprintu a tyto informace využil do dalšího. Maximální délka je pro měsíční sprint stanovena na 3 hodiny. Retrospektiva se snaží zjistit tři věci:

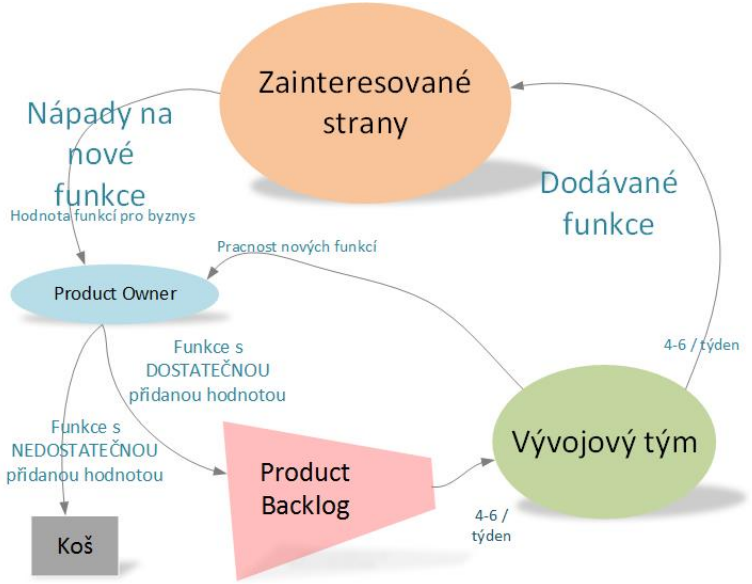
- Jak fungují vztahy, očekávání a procesy mezi lidmi v týmu
- Silné, slabé stránky
- Vytvoření plánu jak se zlepšit

Scrum Master se přímo neúčastní schůzky, ale pokud má někdo nějaký problém během sprintu, motivuje členy týmu a doporučuje jim, aby to zmínili během retrospektivy a pokusili se společně vytvořit plán nápravy.

1.2.2.4 Jak funguje Scrum

Nejznámější metodikou pro agilní softwarový vývoj je Scrum. Před jeho nástupem byly agilní metodiky téměř neznámé.

Scrum je agilní přístup pro vývoj inovativních produktů a služeb. S agilním přístupem, začnete vytvořením Product Backlogu - prioritizovaného seznamu vlastností a schopností potřebných k dokončení úspěšného produktu. Tým vedený Product Backlogem vždy pracuje na nejdůležitějších činnostech vedoucích k úspěšnému dokončení produktu. Pokud by se stalo, že týmu dojdou zdroje (čas, peníze, ...) není to takový problém jako u tradičních metodik, protože vždy budou hotové důležitější věci než věci nedokončené.



Obrázek 1.4: Pohled na životní cyklus Scrumu podle [20]

Práce sama o sobě se provádí v krátkých, přesně časově ohraničených iteracích, které se obvykle pohybují v délce od jednoho týdne do jednoho kalendářního měsíce. Během každé iterace dělá samo-organizovaný tým funkční napříč celou organizací všechnu práci (plánování, vývoj a testování), která je potřebná k vytvoření kompletní, funkční části systému, která by mohla být uvedena do produkce.

Množství práce v Product Backlogu je typicky mnohem větší, než je tým schopen splnit v jedné iteraci, takže na začátku každé iterace, tým plánuje, které úkoly s vysokou prioritou z Product Backlogu vytvořit v nadcházející iteraci. Na konci iterace, tým předvádí dokončené funkce systému všem účastníkům projektu a získává jejich zpětnou vazbu. Na jejím může Product Owner měnit co je v plánu udělat dál a jakým způsobem. Například v případě, že zúčastněné strany vidí dokončenou funkci a pak si uvědomí, že se musí rovněž implementovat do produktu, Product Owner může v produktu jednoduše vytvořit novou položku představující tuto funkci a vložit ji do Product Backlogu na správné místo, aby se na ní pracovalo v budoucí iteraci. Na konci každé iterace by měl tým být potenciálně schopný doručit ten produkt (nebo přírůstek produktu), který může být v případě potřeby nasazen ať už do produkčního nebo jen testovacího prostředí.

Jak každá iterace skončí, celý proces je zahájen novým plánováním další iterace.

1.2.3 Lean Software development

Čím je Agilní manifest pro agilní metodiky, tím je pro Lean kniha Mary Poppendieck a Toma Poppendieck Lean Software Development: An agile toolkit. [21]. Lean metodiky jsou založeny na 7 základních principech:

1. Minimalizovat „odpad“
2. Podporovat učení
3. Rozhodovat se nejpozději, jak je to možné
4. Dodávat výsledky co nejrychleji
5. Posilovat týmového ducha
6. Posilovat konceptuální integritu systému
7. Dívat se na všechno v souvislostech

Minimalizovat „odpad“ Odpad je všechno, co nepřináší zákazníkovi hodnotu produktu. Například pokud vývojáři implementují funkce systému, které zákazník nevyužívá. Agilní a Lean principy napomáhají tomu, aby bylo možné zjistit, co zákazník přesně chce a aby mu právě to bylo dodáno.

Podporovat učení Celý život se musíme vzdělávat, při své každodenní práci musíme zkoušet nové přístupy. Pokud nefungují, musíme se vrátit na začátek, poučit se z chyb a začít znovu. Pokud nebude tento přístup podporován, nebude možné se kontinuálně zlepšovat.

Rozhodovat se nejpozději, jak je to možné Ve vývoji digitálních produktů můžeme v každé fázi vývoje počítat s určitou mírou neurčitosti. Nejlepší způsob, jak se s ní vyrovnat, je rozhodovat se až ve chvíli, kdy je to nutné. K těmto rozhodnutím můžeme mít maximální množství dostupných informací.

Dodávat výsledky co nejrychleji Problémem předchozích epoch ve vývoji SW byl nutnost dodávat naprosto bezchybné produkty. V současnosti je však potřeba mít co nejdříve alespoň základ produktu, abychom věděli, kterým směrem se má dále vyvíjet, abychom mohli dělat lepší rozhodnutí a začali dříve sbírat zpětnou vazbu od našich zákazníků. Stejně tak je to dobrý způsob ke snižování „odpadu“.

Posilovat týmového ducha Kdo zná nejlépe detaily vyvíjeného produktu? Vývojový tým. Lean vychází z agilních principů, které předávají více zodpovědnosti v rozhodování nad produktem do rukou vývojářů. S právy přichází i povinnosti, ale není vždy lehké se rozhodnout. Vývojáři musí být dobře vedeni leadry ze svých řad, kteří mají přirozenou autoritu, musí být povzbuzováni, musí být zbaveni zbytečných procesních detailů a musí dýchat pro tým.

Posilovat konceptuální integritu systému Zákazníka nezajímá, která komponenta systému se zabývá tím, co potřebuje. Celý systém musí vystupovat a hlavně být vyvíjen jednotně. Produkt se pak nadále dá jednoduše zlepšovat a provozovat.

Dívat se na systém v souvislostech Každý se musí dívat na systém jako na celek. Pokud se vývojář soustředí pouze na vlastní část a nebude reflektovat integrace na ostatní části systému, nastane problém a systém jako celek nebude dobře fungovat. Takové přemýšlení vyžaduje velkou dávku sebekázně, protože přirozeně lidské je, starat se jen o své problémy.

1.2.3.1 Kanban

Při Kanbanu se využívají artefakty a postupy tak, aby byly uspokojeny tři zásady. [22]

1. Znázorňuj - k tomuto čelu dobře slouží Kanban board. Je podobný Scrum boardu s rozdílem, že má několik méně podstatných částí navíc. Na Kanban boardu lze jednoduše vidět, jak práce na úkolech probíhají.

2. Omez práci v průběhu - stanovení pevného limitu, například pět úkolů pomůže soustředit se na malé množství úkolů a dokončovat je.

3. Kontroluj si průběh

Dále Kanban doporučuje dodržovat Lean principy. Kanban je často využíván v průmyslové výrobě, nestanovuje žádné jiné artefakty ani role a jelikož není vhodný pro řízení agilních projektů, spokojíme se pro účely práce s konstatováním, že existuje. Principy Lean a Kanban board se používají jako ozvláštnění nebo zpřesnění a zkvalitnění Scrumu.

1.2.4 Extreme programming (XP)

Metodika extrémního programování byla definována Kentem Beckem a jeho spolupracovníky v 90. letech minulého století. Další popis vychází z jeho první knihy o Extrémním programování. [23]

Extrémní programování probíhá jako klasický softwarový vývoj, ale všechny osvědčené postupy dotahuje do extrémů, odtud tedy jeho název. XP týmy vykonávají téměř každou aktivitu vývoje software současně. Analýza, návrh, implementace, testování a nasazení, dokonce s vysokou frekvencí. XP to dělá tak, že pracuje v iteracích pevně časově rozdělených po týdenních přírůstcích práce. Každý týden tým dělá část plánování, nasazení, návrhu, kódování a testování. Pracují na „příběhu“, což jsou velmi malé funkce, nebo jejich části, které mají hodnotu pro zákazníky (konečné uživatele). Každý týden se tým zavazuje poskytovat čtyři až deset příběhů, na jejichž všech částech vývoje po tuto dobu pracuje. Na konci týdne, by měl být software nasazen pro interní kontrolu, někdy lze nasazovat i do produkce. Následující části tradičních fází SW vývoje popisují, jak je vykonávat ve shodě s iteracemi Extrémního programování.

1.2.4.1 Plánování

Každý tým XP obsahuje několik lidí z byznysu (expertů na straně zákazníků), kteří jsou zodpovědní za byznys rozhodnutí a směřují projekt správným směrem, sjednocují jeho vize, vytvářejí příběhy, navrhují plán pro release a řízení rizik. Programátoři poskytují odhady a návrhy, které se propojí s prioritami zákazníků v procesu nazývaném plánovací poker. Společně tým usiluje o dodávání malých, častých releasů, které maximalizují hodnotu produktu. Plánování je nejintenzivnější během prvních několika týdnů projektu. Během zbývajících částí projektu, zákazníci i nadále přezkoumávají a zlepšují vizi a release a plánují postup pro nové příležitosti a neočekávané událostí. Kromě celkového plánu releasu vytvoří tým na začátku každé iterace podrobný plán pro nadcházející týden. Tým se každý den schází na stand-up meetingu.

1.2.4.2 Analýza

On-site expertní zákazníci mohou, ale nemusí být reální zákazníci, v závislosti na typu projektu, ale musí to být lidé nejlépe schopní určit, co by software měl dělat. On-site expertní zákazníci jsou odpovědni za zjišťování požadavků na software. Dělají to tak, že používají své vlastní znalosti a znalosti zákazníků v kombinaci s požadavky na systém. Když programátoři potřebují informace, jednoduše se zeptají. Zákazníci jsou zodpovědní za organizaci své práce tak, aby byli vždy připraveni, když programátoři požádají o informace. Zjistí obecné požadavky na příběh, než programátoři odhadnout její náročnost. Některé požadavky jsou složité nebo obtížně pochopitelné, proto je zákazníci formalizují s pomocí testerů, a to vytvořením user testů. Na pomoc při komunikaci používají programátoři při návrhu společný jazyk, kterému rozumí jak oni, tak on-site expertní zákazníci.

1.2.4.3 Implementace

Při implementaci je důležité neustálé učení a nejlepším učením je implementace. Pokud napíšeme kód, můžeme se z něj poučit a příště napsat lepší. Kód je také nejlepší způsob, jak vysvětlit dalšímu vývojáři, co jsme mu chtěli popsat slovně. Je důležité ke každému kódu psát testy. Jak si povíme níže, pokud ke kódu neexistuje unit test, jako by neexistoval vůbec.

1.2.4.4 Testování

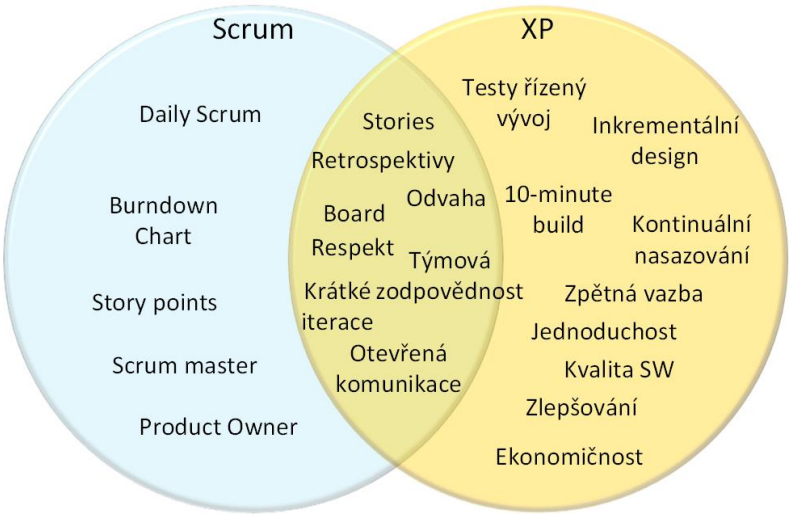
Psaní automatizovaných testů je základ Extrémního programování. Aby se mohla každá malá část systému pravidelně nasazovat, musí být stoprocentně otestovaná. Psaní testů spolu s kódem je mnohem rychlejší, než kdybychom psali testy dodatečně. Je důležité si stanovit jakou míru kvality chceme zajistit a následně podle toho vytvořit testovací strategii.

1.2.4.5 Praktiky používané v XP

Na obrázku 1.5 lze pozorovat společné a rozdílné principy pro XP a Scrum. Zásadní rozdíl spočívá v tom, že XP definuje mnoho zajímavých programovacích technik, které se dnes používají i v neXP týmech. Scrum oproti tomu jako jediný definuje řídicí role.

Plánovací hra Byznys musí učinit následující rozhodnutí:

- Velikost dodávaného produktu
- Priorita jednotlivých úkolů
- Jak se mají jednotlivé komponenty nasazovat do produkce



Obrázek 1.5: Rozdíly metodiky Scrum a metodiky XP. Založené na [2]

- Data nasazování

Jelikož byznys sám nemůže definovat tyto požadavky, potřebuje od IT následující parametry:

- Odhady pracnosti zadaných úkolů
- Pokud by se byznys nějak rozhodl, jaké by to mělo technické dopady
- Procesy fungující uvnitř týmu
- Domluvit se na časovém plánu

tomuto procesu se říká Plánovací hra.

Nasazování malých změn Každé nasazování nového produktu by mělo být co možná nejmenší, ačkoliv funkčnost systému musí být úplná. Nemá smysl nasazovat něco, co funguje jen napůl.

Jednoduchý návrh Dobrý návrh projde testy v jakémkoliv okamžiku, neduplikuje jednou definovanou logiku a má nejmenší počet metod.

Testování Žádná část systému teoreticky neexistuje pokud nemá k sobě zároveň napsány unit testy. K tomuto účelu dnes slouží přístup nazývaný „testy řízený vývoj“.

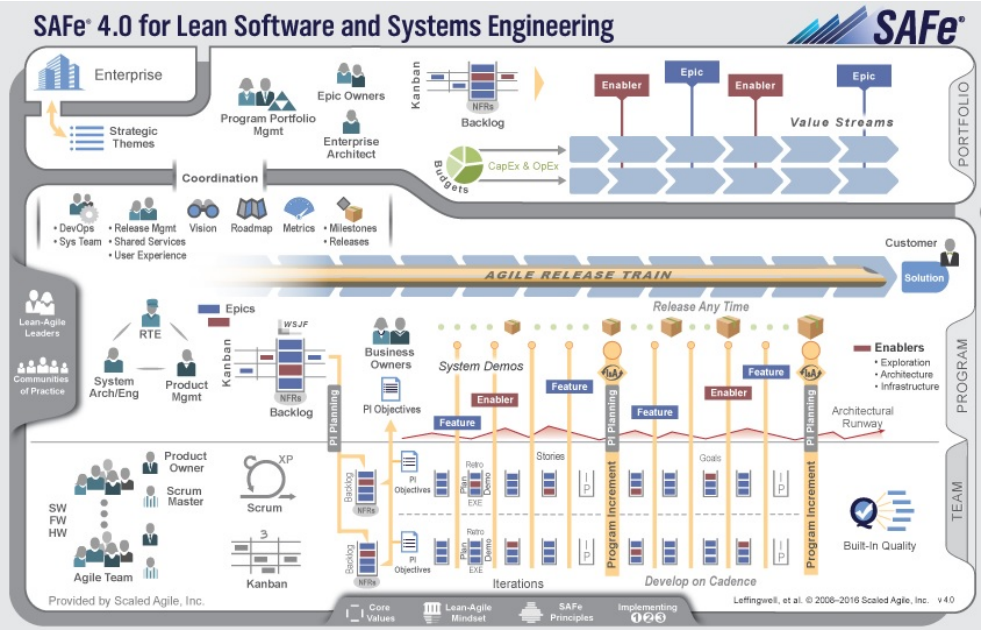
Párové programování Programují spolu dva programátoři, jeden se stará o návrh, kontrolu chyb druhého a druhý se stará o psaní kódu. Toto je velice dynamická metoda, která zásadně snižuje počet chyb v kódu. Názor na její účinnost si musí udělat každý sám.

Nástroje pro kontinuální integraci Implementovaná změna je každý den přímo dávána do produkce a musí být bez jediné chyby. Toto se zajistí psaním unit testů a neustálým testováním.

40-hodinový pracovní týden Tato praktika říká, že žádný vývojář nesmí pracovat přesčas, protože by byl více unavený následkem čehož by byl náchylnější k psaní chyb.

1.2.5 Scaled Agile Framework (SAFe)

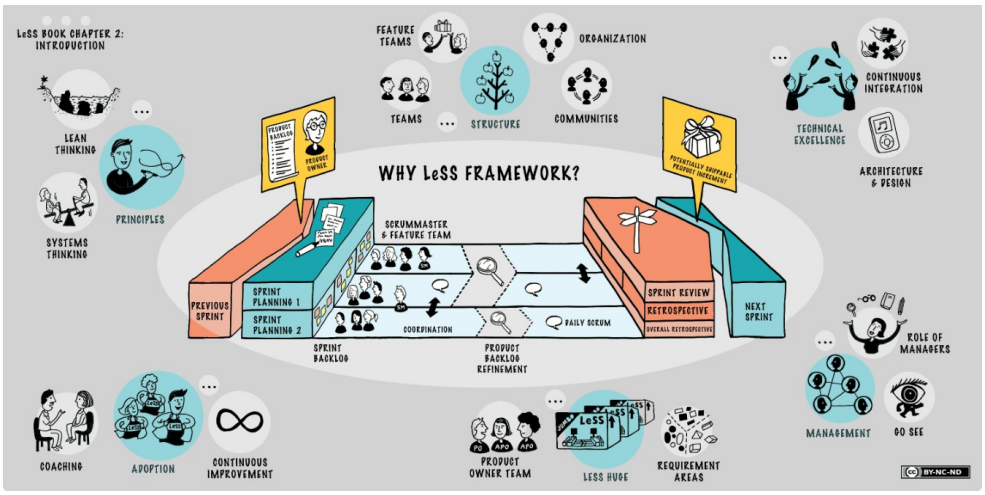
Přehled SAFe frameworku je zobrazený zde 1.6. SAFe je složená metodika, která se v první fázi zabývá řízením týmu, následně řízením více týmů a zadáváním práce z jednoho Program Backlogu a nakonec řízením Program Backlogů na úrovni portfolií, a to za pomoci Kanbanu a se vstupy ze základních strategických dokumentů společnosti jako obchodní strategie a IT strategie. V sekcích dále budou detailněji rozebrány jednotlivé stěžejní body metodiky a hlavně pak ty body, bez kterých by použití metodiky SAFe nemělo význam. [3]



Obrázek 1.6: Přehled metodiky Scalled agile framework [3]

1.2.6 LESS

LESS [4] přináší řešení jak aplikovat principy, účel, části a krásu Scrumu do kontextu většího škálování tohoto procesu směrem k agilnímu řízení projektů nejjednodušeji, jak je to jen možné. LESS využívá Scrum pro řízení jednotlivých týmů a pak řeší, jak jednotlivé Scrum týmy efektivně koordinovat a řídit jejich funkci napříč komponenty a napříč funkcemi. Důležitým konceptem tohoto řešení je společná práce. Týmy musí spolupracovat, protože mají společný cíl. Tímto cílem je doručení SW k zákazníkovi. LESS se dělí na dvě části. Jedna část je do počtu 8 týmů a jmenuje se jen LESS, druhá je LESS Huge pro více než 8 týmů.



Obrázek 1.7: Přehled metodiky Large scalled scrum [4]

1.2.6.1 Co přidat do Scrum, aby z něj byl LESS

- Sprint plainning je rozdělený na dvě části, přičemž první se účastní členové ze všech týmů a rozhodují se, co přidat do Product Backlogu. Dohodnou se na kooperaci na implementaci konkrétních pracovních balíčků. V druhé části se paralelně sejdou jednotlivé týmy a sprint plainning probíhá jako klasický Scrum sprint plainning.

- Je zde malá změna v Daily standupech, na kterých se mohou objevovat i členové jiných témat, pokud se jich témata z jiného týmu týkají.

- Koordinace probíhá tak, že zástupci týmů nebo celé týmy se scházejí, aby se nestalo, že aktivita jednoho týmu negativně zasahuje do aktivity druhého.

- Doporučuje se schůzka Product Backlog Refinement (PBR), kde by se měl sejít jeden PO se členy všech týmů a diskutovat s nimi, který tým bude implementovat jeho požadavky.
- Přibývá jedna schůzka – všeobecná retrospektiva, kde se diskutuje, jak by se mohl zlepšit celý systém a spolupráce týmů spíše než práce samotného týmu

1.2.6.2 Principy LESS

Transparentnost Zaměření na dotahování věcí, krátké cykly, společná práce a společné cíle odstraňují strach z pracoviště.

Méně je někdy více LESS nepotřebuje definovat nové role, protože to by vedlo k rozložení odpovědností na více lidí. Méně procesů i artefaktů. Naopak je potřebná zodpovědnost týmů za dodání produktu a větší zaměření na zákazníka.

Zaměření na celý produkt Jeden Product Backlog, jeden Product Owner, jeden produkt. Celý SW musí být brán komplexně, nesmí být rozdělen na nesourodé části.

Zaměření na zákazníka Vývojáři musí chápat zákazníka a jejich dílo musí řešit jeho problém. Zákazník nesmí čekat nepříjemně dlouhý čas a musí od něj být pravidelně sbírána zpětná vazba. Všichni musí rozumět tomu, co zákazníkovi přináší hodnotu.

Kontinuální zlepšování Cílem je dodávat produkt, který stojí nejmenší prostředky a nemá téměř žádné chyby, který uspokojí klienta, vylepší celé prostředí a vytvoří lepší životní podmínky pro jeho uživatele. Proces musí být neustále vylepšován za účelem přiblížení se těmto cílům.

Myšlení Lean Představuje vytvoření systému v organizaci, kde manažeři učí, jak myslet co nejjednodušeji a jak řídit všechny kolem sebe ke změně.

Myslet systémově Znamená to vidět, rozumět, optimalizovat a modelovat celý systém, ne pouze jeho části, zajišťovat jeho dynamiku. Vývojář se nesmí dívat na svou část systému, ale na celý produkt, protože to je to, co zákazník vidí.

Empirická kontrola procesů Neustále kontrolovat a přizpůsobovat produkt, procesy, chování, návrh organizace a činností k vývoji situačně vhodných způsobů. Je to lepší způsob, než ignorovat kontext a řídit se jen plánem, který byl nastaven.

Teorie obsluhy systémů Pochopit, jak systémy s frontami fungují v oblasti vědy, výzkumu a vývoje. Poznatky následně využít k řízení velikosti front, limitů dělané práce, dělání více úkonů najednou a pracovních balíčků.

1.2.7 Nexus

Nexus je další z frameworků, který vznikl ve snaze škálování agilních frameworků. Zásadní změnu přináší tím, že pro celý produkt je jeden velký Product Backlog, který se dělí na jednotlivé Sprint Backlogy pro jednotlivé týmy. Poté co více scrumových týmů pracuje na svých částech, se na konci Sprintu opět zjistí, které věci z Product Backlogu byly implementovány, které ne a opět probíhá Nexus Sprint planning.

1.2.8 Nástroje podporující agilitu

Jedním ze zásadních bodů při zavádění agilních prvků do velké společnosti, jsou softwarové nástroje, které napomáhají automaticky sestavovat spustitelný kód, nasazovat na aplikační server a testovat SW. Jedním z nejdůležitějších prvků agilního vývoje je iterativní způsob vytváření SW. Vždy po malých iteracích dodáváme nové funkce systému, a pokud bychom museli dávat tyto malé části kódu do provozu ručně, nebyl by vývoj takového produktu ani efektivní, ani ekonomicky výhodný. To je zásadní při rozhodování o tom, zda je agilní řízení finančně výhodné, nebo nevýhodné. Je tedy nutné použít SW nástroje, které napomáhají na jedno zmáčknutí tlačítka myši nasadit nově upravený kód. Sadě takových nástrojů se říká Nástroje pro kontinuální integraci, skládají se ze skriptovacího jazyka, který sestavuje spustitelný kód, integračního serveru, který celý proces řídí, sady nástrojů na unit a funkční testování a konečně z nástroje, který připraví zdroje aplikace na serveru a nasadí samotnou aplikaci. Konkrétní nástroje používané ve zkoumané organizaci budou popsány v praktické části práce.

1.3 Klasické vodopádové metodiky

Vodopádové metodiky pomáhají stabilně řídit projekty. Vodopádový přístup má své výhody a nevýhody, které vycházejí ze samotné podstaty lineárního přístupu k vývoji software. Nejdříve si tyto výhody a nevýhody popíšeme podle [24] a následně si popíšeme nejznámější, často ne plně vodopádové metodiky. Důvod, proč se vodopádovými metodikami zabýváme je, že i do budoucna budou mít své využití ve velkých společnostech, které musí řídit velké nebo integračně složité projekty. Více si pak povíme v sekci o Bimodal IT.

1.3.1 Výhody a nevýhody vodopádového řízení aneb proč není za zanitem

1.3.1.1 Výhody

Předvídatelnost Když všechno jde podle plánu, pak dokonale víme, jaké kroky v projektu nastanou. Předtím, než vývojáři napíší první řádku kódu přesně víme, kolik práce se udělá a kolik peněz to bude stát

Stabilita Protože požadavky jsou definovány v počátečních fázích projektu a dále se nemění, zákazníci přesně vědí, co přesně dostanou a kdy. Zákazníkem je často v bankovním prostředí byznys. To je nejdůležitější pro kritické systémy. V bance se může jednat o platební systém, který převádí peníze klientů.

Šetření peněz Když navrhneme aplikaci dokonale, nemůže se stát, že děláme na části systému zbytečně. Všechny části se využijí

Detailní design Detailní analýza na začátku projektu napomáhá navrhnout vyvíjený produkt od začátku tak, že v pozdějších částech projektu nemusíme plýtvat časem na pozdější rozhodnutí ohledně architektury. Čistě následujeme plán a to dělá programování jednodušším.

Méně přepisování kódu bez změny funkčnosti Agilně vedené projekty mají tendenci neustále upravovat a přepisovat kód. Výsledek je neustálé přepisování k uspokojení zákazníkových potřeb a standardů kódu.

Lepší dokumentace Před začátkem vodopádového projektu je nutné sestavit kompletní dokumentaci s maximální možnou mírou detailu. V dalších částech projektu je dokumentace neustále upřesňována.

1.3.1.2 Nevýhody

Nepřizpůsobivost Pokud se požadavky změní po fázi analýzy a návrhu, není cesty zpět. Tato mezery v ohebnosti také znamená, že nemůžeme získat hodnotu z nových příležitostí.

Pozdější přinášení hodnoty Každý iterativní přístup přináší hodnotu okamžitě s první životaschopnou verzí. Ve vodopádovém přístupu nedostanete nic, dokud není vývoj kompletní.

Nelze začít vývoj s minimem informací Před prvním napsaným řádkem musí být vše dokonale zanalyzováno. Pokud bychom ve fázi implementace chtěli využít iterativní přístup a rychle dodávat hodnotu, nepůjde to dříve než po kompletní analýze, což je samo o sobě v pozdějších fázích projektu.

1.3.2 Čistě vodopádový model

Vodopádový model je naprosto prediktivní model. Předpoklad je takový, že všechny fáze jsou od sebe naprosto oddělitelné a dají se celé udělat než přejdeme do dalšího kroku vodopádu. Neoddělitelné kroky odpovídají životnímu cyklu vývoje software. Vodopád funguje bezchybně za následujících předpokladů:

- Požadavky jsou dokonale předem známé
- Požadavky neobsahují nevyřešené vysokorizikové věci
- Požadavky se po začátku implementace nemění
- Tým má zkušenosti s podobnými projekty
- Je dostatek času vše dělat sekvenčně

1.3.3 Úpravy vodopádu bez použití iterativního přístupu

Vodopád se zpětnou vazbou Mírné vylepšení přináší vodopád, který se při nedostatku informací nebo při změně požadavků může vrátit o do předchozí fáze.

Sashimi Sashimi je vodopádový model „vylepšený“ o možnost překrytí fází. Tj. v době kdy probíhá už návrh aplikace může ještě současně končit sběr požadavků. Tím se odstraní problémy s čekáním na ukončení předchozí fáze. Musí být nastaveny limity, jak se jednotlivé fáze mohou překrývat. Sashimi částečně řeší problém, že se můžeme vrátit do návrhu, pokud přijdou nové požadavky ve fázi implementace.

Inkrementální vodopád Využívá sérii vodopádových kaskád, každá kaskáda končí s doručitelným produktem, který přináší hodnotu. Po každém inkrementu se mohou změnit požadavky nebo se můžeme poučit z předchozího inkrementu a lépe definovat, kterým směrem se projekt má vydat. iterace by se neměly překrývat a měli bychom průběžně sbírat zpětnou vazbu od uživatelů využívajících produkty z předchozích inkrementů.

V-model Je rozšířením vodopádového modelu. Zabývá se nejen vývojem, ale také předáním do provozu. Definuje vrstvy, na kterých při předávání do provozu testujeme, zda byl produkt dobře implementovaný.

1.3.4 Vodopádové metodiky s využitím agility v softwarovém vývoji

Unified Process

1. Počátek – v této fázi někdo přijde s nápadem na projekt Tato fáze je velice krátká. Stačí si definovat základní využití realizovaného produktu, rizika, která by mohla nastat při jeho implementaci, počáteční plán a hlavní cíle projektu. Neobsahuje detailní zadání pro programátory.

2. Elaborace – během této fáze se vytváří požadavky na implementaci na IT. Diagramy tříd, diagrami užití a architektura systému. Účel fáze není vytvořit detailní zadání, ale navrhnout velikost a rizika řešení na základě bližšího popisu a zásahu projektu do stávající infrastruktury.

3. Přechod – Během této fáze se systém implementuje, následně se systém doručuje koncovým zákazníkům a určuje se tým, který ho bude dlouhodobě podporovat.

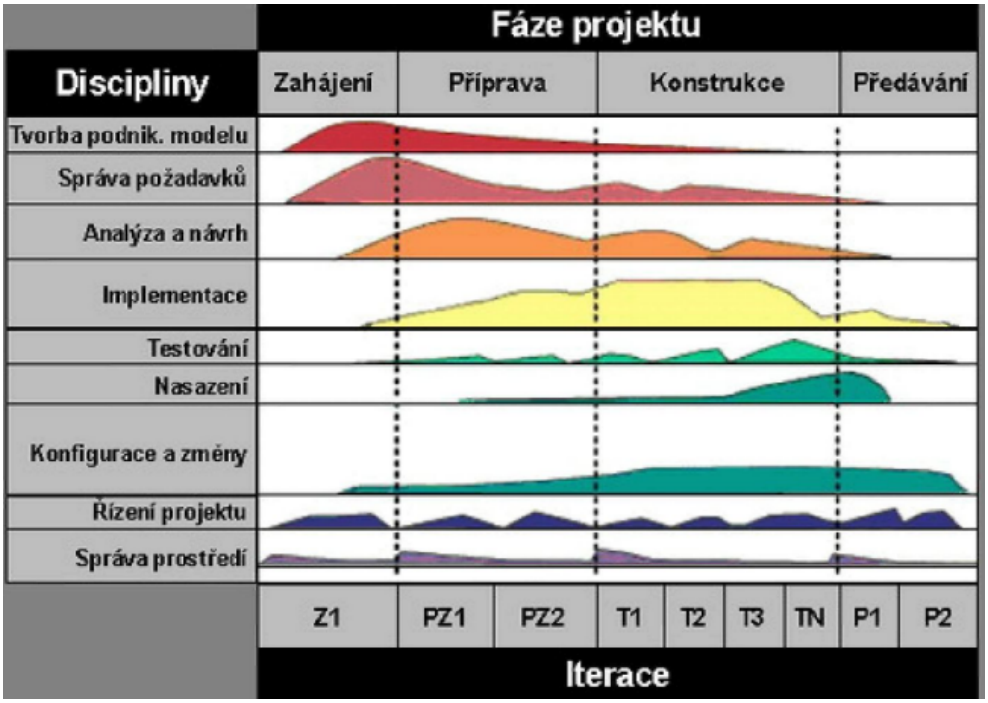
4. Použití – konečníuživatelé používají systém a tým sbírá zpětnou vazbu.

5. Výměna – během této fáze dobíhá starý systém a plánuje se další projekt na výměnu tohoto systému.

Míra pracnosti, která se použije pro jednotlivé fáze a iterace je stejná jako v případě Rational Unified process. 1.8

Rational Unified Process(RUP) Komerční vodopádová metodika vytvořená společností Rational, která byla následně koupena společností IBM využívá objektově orientovaný iterativní přístup řízený případy užití. Také jinak hlavní přístup je vytvořit nejdříve případy, k čemu by se měl systém používat a podle těchto případů užití implementovat funkce tak, aby byly případy uskutečnitelné. Předpoklady pro její použití jsou, že všechny požadavky musí být známy před začátkem implementace. Rational Unified Process [25] je proces softwarového vývoje. Přináší disciplinovaný přístup k přiřazování úkolů a zodpovědností napříč organizací, ve které se uplatňuje. Jeho cílem je ujišťovat se, že společnost produkuje vysoce kvalitní software, který splňuje požadavky klientů, za předpokladu předvídatelného plánu a fixního objemu financí. Vývojové týmy úzce spolupracují se zákazníky, dodavateli i dalšími účastníky procesu,který je neustále vyvíjen a zlepšován. Důležitým prvkem je přístup ke sdíleným informacím napříč týmem. Díky tomu se dosahuje sjednocení cílů a jazyka, kterým spolu členové týmu mluví. RUP se snaží spíše než dlouhou dokumentaci podporovat vytváření unifikovaných modelů, které plně popisují, jak je realizované řešení vytvořeno, a to nejčastěji pomocí UML diagramů. UML je široce uznávaný standard od skupiny Object Management Group. Proces je podporován co největším množstvím automatických nástrojů, které

napomáhají proces urychlit. Je použitelný jak pro malé týmy, tak pro velké týmy.



Obrázek 1.8: Přehled pracnosti v jednotlivých fázích a iteracích UP a RUP. Převzato z [5]

1.4 Porovnání klasických vodopádových metodik a agilních metodik

Oba typy projektových metodik mají základ v softwarovém vývoji, který probíhá vždy stejně - analýza, návrh, implementace, nasazení. V čem si podobné nejsou? Jde hlavně o změnu v podnikové kultuře a zaměstnancích. Je nutné změnit myšlení a přistupovat k pracovním povinnostem jinak. Jinak se plánuje a přístup k vývoji je buď sekvenční nebo iterativní.

1.4.1 Plánování

V tradiční metodice se stanovují cíle v rámci jednotlivých fází, v horizontu měsíců. Například vytvořit důkladnou analýzu za 2 měsíce nebo implementovat změnu za 1 měsíc. Agilní plánování probíhá rychleji, častěji a s důrazem na větší detail. Změny přicházejí rychleji a je vidět jen na konec sprintu, každý

den může pracovník přijít na standup a zjistit to, že co bylo včera, už neplatí, a směr celého projektu se mění, protože vlastník produktu změnil priority.

1.4.2 Lidé

S ohledem na všechny rozdíly mezi tradičním a agilním řízením projektů je jasné, že i lidé, kteří v jednotlivých módech pracují, musí být naprosto odlišní. V agilním módu musí být zvyklí na rychlé změny zadání, komunikaci s lidmi, kteří projekt sponzorují a kteří ho budou používat. Naopak v tradičním řízení je nutné soustředit se na detail, nepřichází tolik změn a práce v tomto módu je klidnější. Vyžaduje však osobní disciplínu, aby byl pracovník ochotný dlouhodobě pracovat na zadaných úkolech a splňovat deadliny. V opačném případě by mu práce nevyhovovala a byla by vysoce stresová.

1.4.3 Procesy

Pracovníci, obzvláště pak v korporátním prostředí, jsou zvyklí, že často trvá déle sdělit změnu v systému než samotná implementace změny. Pokud by se jim tato administrativní zátěž snížila, mohlo by se stát, že v agilním módu nebudou vědět co přesně v jaké fázi vykonávat. V agilním módu musí být pracovník kreativnější a méně se řídit předdefinovanými procesy. Toto by mohl být problém jak pro pracovníky, tak pro management v cílovém řešení a při jeho návrhu bude bráno v potaz, že i v agilním módu budou muset platit jasná pravidla, jak se chovat při vývoji SW.

1.4.4 Přínosy agilních a vodopádových metodik

Tradiční metodiky přináší stabilitu a procesně řízené prostředí, které je vhodné pro pracovníky, kteří chtějí dostat jasně definovaný úkol a mít dostatek času k jeho řešení bez změn zadání. Tradiční metodiky se snaží nechat dostatečný čas na analýzu stávajícího stavu a je tak vhodný pro složité integrační projekty, které nelze implementovat přírůstkově a iterativně. Mód 1 je stavěný pro řízení velkých projektů.

Agilní metodiky jsou vhodnější pro inovativní pracovníky, kterým nevadí ze dne na den se měnící dynamické prostředí. V tomto módu je méně administrativní zátěže, přináší rychlejší řešení v neprobádaných oblastech a technologiích. Realizuje v krátkém čase PoC, které se pak využívají k rozhodnutí, zda je výhodné spustit velký projekt řízení mód 1 pro realizaci změny. Pomocí agilního mód 2 se do velkých společností zavádějí inovace. Mód 2 je stavěný na menší projekty či malé změny na rozsáhlých systémech.

Tabulka 1.1: Porovnání mode 1 a 2

Vlastnost	mód 1	mód 2
Cíl	návratnost	obratnost
Hodnota	cena/výkon	zisk, značka, Customer Experience
Přístup	vodopádový, V-model	Scrum, Kanban
Řízení	plánované	empirické, podle aktuální potřeby
Kvality	známé domény	neprobádané oblasti
Zákazníci	bez kontaktu se zákazníky	přímá spolupráce se zákazníky.
Změny	zřídka	často

1.5 Bimodal IT podle Gartner

Celá sekce je inspirovaná přednáškou Simona Mingaye, která je po registraci dostupná online. [16]

IT se rozdělí na dva módy:

- mód 1 - tradiční mód, předvídatelný, přesný, stabilní
- mód 2 - průzkumný mód, který je agilní a rychlý

Co Bimodal IT není:

- Rozdělení společnosti na dvě dceřiné
- Reorganizace společnosti
- Čistě agilní vývoj
- Stínové IT

Gartner často zaměstnance z mód 1 připodobňuje k maratonským běžcům a zaměstnance mód 2 ke sprinterům. Jaké hodnoty jim musí být blízké stanovuje následující tabulka.

mód 1 bude více zaměřený na fungující systémy a mód 2 na inovativní projekty. Gartner doporučuje Bimodal IT zavádět ve čtyřech fázích:

1.5.1 Ostrůvky agility

Ostrůvky agility jsou uzavřené části společnosti, které komunikují uvnitř agilně a s okolím společnosti komunikují tradičním způsobem. Ostrůvky mají tu výhodu, že agilita na nich jde vyzkoušet bez zásadnějších změn organizace. Takový ostrůvek by měl splňovat následující parametry, být rozsahem malý, při-

nášet finanční hodnotu, mít dostatečně kvalitního partnera v byznyse, nebýt složitý a být inovativní.

1.5.2 Nahrazování tradičního vývoje agilním

Druhý stupeň představuje snahu začít některé produkty vyvíjet agilně. Jsou to zejména produkty vhodné pro agilní vývoj. Často se jedná o malé produkty, u kterých by tradiční řízení trvalo déle a o inovativní produkty, se kterými by si současní zaměstnanci nevěděli rady a zbytečně by ztráceli čas.

1.5.3 Rozšíření jádra

Další rozšíření by mělo zásah do kritičtějších systémů organizace. Jak si povíme dále, pro tento typ změny je potřeba změnu důkladně komunikovat a promyslet všechny důsledky.

1.5.4 Nové jádro aplikací

Pokud jsme i nejdůležitější aplikace ve společnosti schopni vyvíjet agilně, pokročili jsme na nejvyšší zralost konceptu Bimodal IT a čerpáme plné přínosy tohoto konceptu i s jeho riziky.

1.5.5 Struktura organizace po zavedení Bimodal IT

Gartner doporučuje naprosto oddělit dva módy, a to dokonce i organizačně. Jednou z možností, jak tuto změnu provést je oddělit je na IT a připravit oddělení provozu na změny. Druhá možnost je mód 2 vyjmout mimo IT do jiného oddělení.

1.5.6 Doporučení při zavádění Bimodal IT

Gartner stanovuje následující doporučení:

- Odloučení zaměstnanců módu 1 a módu 2
- Nastavit stejné hodnoty pro oba módy
- Oba módy jsou rovnocenné
- Vybudovat procesy a protokoly pro komunikaci obou módů
- Komunikovat změnu k zaměstnancům
- Neustále se učit

1.6 Případové studie - zavádění agility do podobně velkých společností

1.6.1 Spotify

Případová studie je založená na [6]. Ve Spotify mají přes 30 agilních týmů ve 3 městech na světě. Využívají jimi upravenou maticovou strukturu řízení. Zjednodušeně se dá maticové řízení popsat tak, že bychom si dali všechny zaměstnance do jedné matice a v řádcích a ve sloupcích by mohli zaměstnanci patřit do jiných a jinak vytvořených skupin. Mají více organizačních jednotek:

1. Skupina je nejmenší stavební prvek. Přibližně odpovídá jednomu Scrum týmu. Každá Skupina má svého PO, stará se o jednu část produktu - playlist, android aplikace, ios aplikace atd.
2. Kmen je kolekce Skupin, která pracuje v jedné oblasti, např. frontend nebo infrastruktura. Kmen má méně než 100 lidí. Skupiny mají autonomii a mohou spolu interagovat i napříč skupinami
3. Kapitola - v kapitole se sdružují zaměstnanci ze stejného kmenu, kteří mají stejné schopnosti - testeři, vývojáři atd.
4. Cech - cech je uskupení zaměstnanců, které spojuje zájem o oblast řekněme Javy. Pokud se budeme dále v bance bavit o komunitách budeme se bavit o podobných uskupeních.

Celé řešení funguje na základě určité volnosti, autonomie, na druhou stranu každý tým má svoji komponentu, na které pracuje a za kterou je odpovědný.

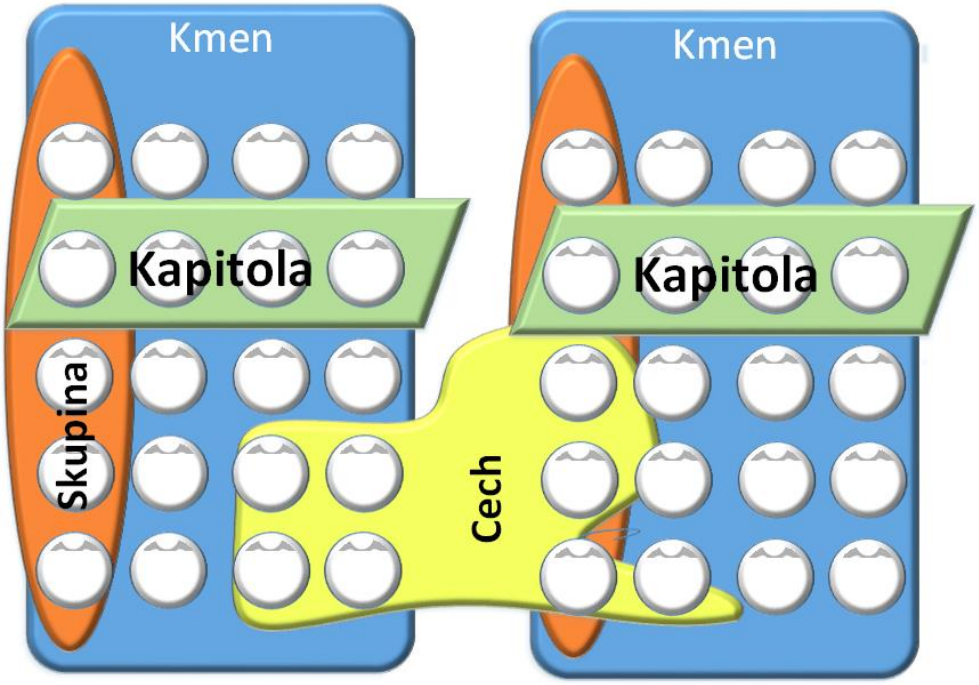
1.6.2 Vodafone Turkey

Případová studie je založená na [26]. V tureckém Vodafone přistoupili k agilní transformaci. Hlavními třemi faktory, které vedly k rozhodnutí o přechodu na agilní řízení, byly zvýšení produktivity, snížení času, který trvá dodání produktu na trh, a zvýšení kvality dodávaného produktu. Agilní transformace byla rozdělena do tří kroků:

1.6.2.1 Pilotní tým Gepard

Vytvoření jednoho týmu, který fungoval agilně, aby se vyzkoušelo, zda je to v rámci firmy možné. Byl vybrán malý projekt pod 50 MD na oddělení CRM. Zjistilo se, že původně rozdělený tým vývojářů a tým testerů si náhle sedl k jednomu stolu a všichni si začali rozumět. Byli jeden tým. Úspěchy týmu byly:

1. Přinesené změny ušetřily miliony na výdajích firmy
2. Zvládl práci 3x rychleji než klasický tým



Obrázek 1.9: Přehled celé maticové struktury Spotify. Vychází z obrázku v [6]

3. Mezi prvním a posledním sprintem se 2,5x zvětšil objem dokončené práce

4. Lidé z byznysu byli nadmíru spokojení

1.6.2.2 Rozšíření agility na více týmů

S rozšířením agility souviselo vyčlenění 7 Scrum týmů, které v krátkých sprintech dodávali software, rychle získávali zpětnou vazbu od zákazníků a přizpůsobovali jí vytvářený produkt. Zásadním poučením z druhé fáze bylo dobré nastavení metrik a standardizace napříč všemi Scrum týmy. Všechny týmy byly odstíněny od zbytku firmy.

1.6.2.3 Rozšíření agility do celé společnosti

V celém konceptu bylo zásadní, že se naučili soustřeďovat se na uspokojení požadavků jejich klientů. Škálování agility do celé firmy ještě Vodafone čeká, ale jisté je, že se z předchozích dvou kroků poučili, jak se má dělat Scrum pouze na úrovni jednoho týmu, a vyzkoušeli si rozšířit Scrum na celé oddělení. Jako v této práci i oni chápou, že pokud by Scrum chtěli rozšířit napříč celou firmou, byly by nutné zásadní zásahy a úpravy v celé organizaci.

1.7 Organizační změny a jejich zavádění

Zavádění změn do velkých společností se začíná jevit jako velká výzva současnosti. Tento vědní obor se zabývá principy, metodikami a doporučeními, které usnadňují adopci změn a pomáhají tak zaměstnancům zvykat si na nové přístupy. Lidský faktor je zásadním prvkem v organizačních změnách. Co by znamenala organizace bez jejích zaměstnanců? Organizace je tvořená zaměstnanci. Sami dávají vzniknout vlastní podnikové kultuře podle toho, jak se ve společnosti k sobě chovají.

Bimodal IT je významná změna v řízení projektů, v přemýšlení IT odborníků nad vývojem softwaru, změna podnikové kultury a vyžaduje čas a dostatečnou míru obeznámení zaměstnanců, proč je změna důležitá. Aby se konceptu nebáli a naopak ho přijali jako změnu k lepšímu. K dosažení těchto cílů napomáhá disciplína zvaná Změnové řízení organizace. Bimodal IT by mělo probíhat jako iterativní změna. Nejdříve v menším měřítku, s postupným rozšiřováním působnosti.

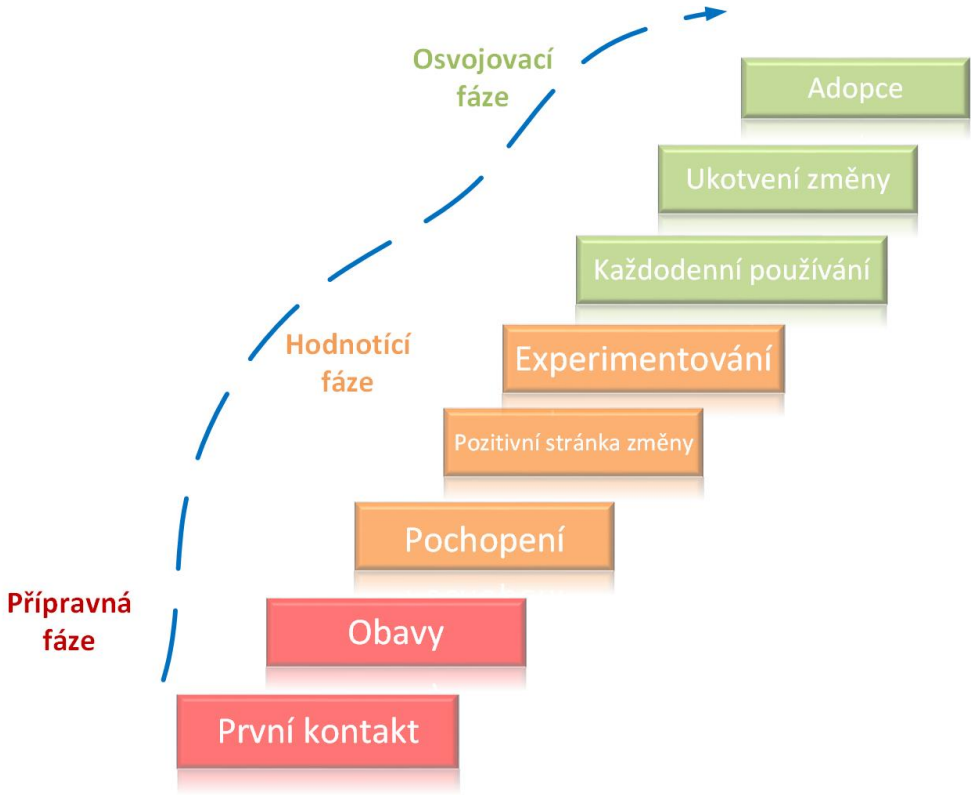
Je nutné dbát na komunikaci s lidmi, kterých se změna týká. Rovnice

$$QXA = E$$

kde Q je kvalita řešení, A je míra toho, jak lidé, kterých se změna týká tuto změnu přijmou a výsledkem produktu těchto činitelů je efektivita. Ta vyjadřuje, že i když máme dokonalý produkt, tak pokud ho lidé nepřijmou a nezačnou používat, výsledný efekt je stále nulový.

Dalším principem, který musíme při zavádění změn mít na paměti je křivka, která ilustruje, jakými fázemi adopce si prochází většina lidí, kteří se potýkají se změnou na pracovišti. Je přirozená reakce, že v prvních fázích se každý bojí neznámého a změnu odmítá. Postupně se se změnou seznamují a začínají chápat přínosy. Pokud dostanou prostor si nové metodiky a postupy vyzkoušet, mohou sami usoudit, zda se jim změna zamlouvá a případně s novinkami experimentovat. Dostáváme se do důležitého bodu adopce, v této fázi by měli koncept každodenně využívat a pokud se jim bude zamlouvat, stane se součástí jejich rutinního fungování a se změnu přijmou. Pokud ne, přestanou ji používat. U každého zaměstnance můžeme tyto fáze vnímat skrz jejich nadřízené a pokud hrozí opuštění správné cesty ke změně, můžeme zaměstnance více motivovat. 1.10 Je to jako s mobilními aplikacemi. Nainstalujete si aplikaci, vyzkoušíte si jí, zjistíte k čemu by se mohla aplikace hodit a když jí nepoužíváte, časem jí pravděpodobně odinstalujete.

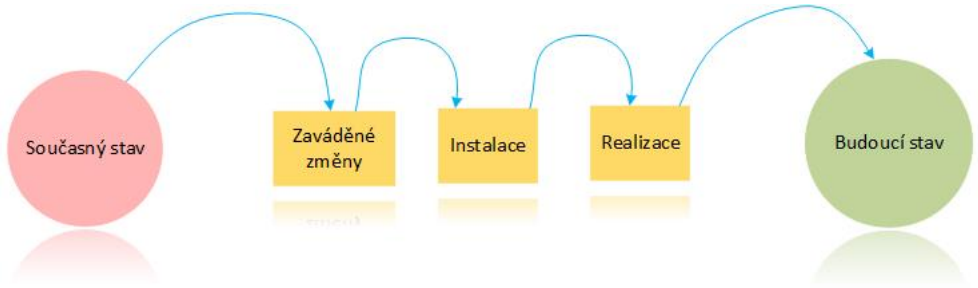
Přijmutí změn organizací Mohli bychom si myslet, že změna se dá v organizaci udělat příkazem. Zavedení změny oznámením zaměstnancům, že změna



Obrázek 1.10: Křivka fází přijímání změny.

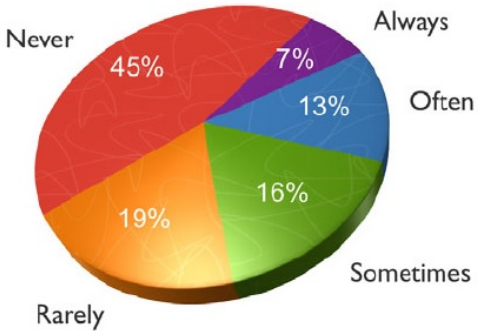
vstupuje v platnost je pouze první krok. Po tomto oznámení nastává fáze Instalace. Vyznačuje se tím, že změna není ukotvena a používána, zaměstnanci se s ní musí seznámit a začínají ji používat. Po instalaci nastává fáze realizace. Fáze realizace se vyznačuje tím, že zaměstnanci si už změnu osvojili a využívají výhod, které jim přináší. Přechod z instalace do realizace je často nepříjemný, protože může stát energii, ale ve fázi instalace zatím jen částečné přínosy, což může zaměstnance demotivovat. Až skončí fáze realizace, zaváděný koncept je plně funkční a získáváme z něj přínosy.

Příklad z reálného světa IT může být zavádění nástrojů pro kontinuální integraci. Nástroje se musí instalovat, nastavit, je s jejich instalací mnoho práce. Pokud ale každá část celku nefunguje korektně, nepřináší žádný přínos. Když se podaří zprovoznit všechny komponenty konceptu, vývojářům odpadá práce s nasazováním hotového kódu na vývojová prostředí. Nástroje sami nasazují a testují kód a pokud by se se změnou kódu zanesla do kódu chyba, okamžitě dostanou emailové oznámení.



Obrázek 1.11: Proces realizace změny

Využívá se jen 20 procent funkcí systému. Proč? Vývojáři se neřídí potřebami zákazníků. Výzkum firmy Standish Group ukázal, že pouze 20 procent funkcí systémů je reálně využíváno uživateli. 45 procent funkcí není nikdy využíváno. Výsledkem tohoto pozorování je, že by stačilo 20 procent funkcí systému dodat zákazníkovi a uspokojilo by to jeho potřeby. To je jeden z hlavních principů agility. Nepracovat rychleji, ale vyvíjet jen to, co přináší hodnotu zákazníkovi. [7]



Obrázek 1.12: Jak často jsou využívány funkce software. Obrázek převzat z [7]

Adopce změn. Rozdělení lidí ve společnosti odpovídá přibližně normálnímu rozdělení. Je jen málo zásadních odpůrců změny a málo nadšených inovátorů, kteří se chtějí měnit každý den. Většina lidí pak jde s většinou a přizpůsobuje se jako rychlejší nebo pomalejší většina. Většina nadšených inovátorů má zásadní problém s tím, že většina není tak nadšená ze změny jako oni. V případě, že s tím počítáme nebo pokud máme praxi v zavádění změn, zjistíme, že nejlepší způsob, jak zavádět změny, je vyhledat inovátory, nadchnout je pro změnu a doufat, že většina ostatních se k nim přidá. Někdy můžeme mít dobrou změnu, kterou zatím většina nepřijímá. Je potřeba neustále vysvětlovat přínosy změny a čekat, až se většina přidá na naší stranu. [7]

1.7.1 Co je to hodnota

Shrnuto na obrázku 1.13. Pět jednoduchých pravidel podle [8] definuje hodnotu ve smyslu softwarového vývoje a způsoby, jak dosahovat vyšších hodnot takto:

1. Hodnota je to co chceme. Funkce systému přináší hodnoty. Doručování funkcí systému dříve přináší hodnoty dříve.
2. Řízení hodnocením přínosů funguje lépe než řízení založené na sledování procesů a termínů projektu.
3. Plánování, které funkce systém bude mít je jednoduché. Nikdy nemůžeme plánovat přesně.
4. Vhodný výběr funkcí, které jsou nutně potřebné k udržení neustále funkčního díla napomáhá přinášet konstantně hodnoty.
5. Neustálé testování je klíč k funkčnímu dílu, které opravdu pracuje pro zákazníka.

Obrázek 1.13: Co znamená hodnota v různých oblastech softwarového vývoje [8][str. 76]

1.7.2 Komplexnost problémů a využití konceptu Bimodal IT k řešení všech typů problémů

1.7.3 Budoucnost softwarového a produktového vývoje

Tradiční vodopádové metodiky si zakládají na stabilitě a pevném řízení projektů. Agilita dává více volnosti, projekty už nejsou tak striktně definované ani

rozsah prací není pevně definován. Tento trend volnějšího řízení a stále kratších iterací dále postupuje. Hlavním současným představitelem nového směru ve vývoji hlavně nových inovativních produktů je metodika os společnosti Google „Design Sprint“.

Google Design Sprint vychází ze základu agilních metodik a posunuje bariéry dále. V google sprintu se funguje v týdenních iteracích, které jsou rozděleny do 5 fází. Každá fáze trvá jeden den. Fáze jsou podle [27] následující:

- Pochopení problému - zjišťujeme detaily problému, který bude náš nový produkt řešit. Kde problém nastává, komu, proč.
- Řešení problému - vymýšlí se nejlepší způsoby, jak problém z předchozího dne vyřešit. Nejpoužívanější techniky jsou vytváření scénářů, vytváření příběhu či myšlenkových map. Na konci fáze se vybere jen jedno řešení vhodné k rozepsání.
- Základní prototyp - vytvoří se obrazovky nebo nákresy, jak by budoucí produkt mohl reálně vypadat. Při vytváření obrazovek nás může napadnout mnoho věcí, které jsme předtím neřešily, které mohou napomoci zlepšit funkčnost produktu.
- Funkční prototyp bez reálných dat - v prototypovacím nástroji se vytvoří oživené obrazovky, které zobrazují nereálná data.
- Poslední den zbývá na otestování funkčního prototypu na reálných uživatelích. Může být provedeno formou placeného průzkumu, na našich známých nebo jen poprosit náhodné kolemjdoucí.

Na začátku dalšího týdne se začíná znovu od začátku. Google s touto metodikou každoročně vyprodukuje 2000 nových prototypů a čtvrtinu z nich dokončí ve finální produkt.

Kapitola 2

Případová studie a návrh zavedení agilních metodik do velké české banky

2.1 Způsob práce na praktické části

Autor zvolil pro spolupráci jednu velkou českou banku, která souhlasila se spoluprací pouze v případě, že nebude zmíněn název a nebudou v práci použity interní materiály. Dále tedy bude zmiňována jen jako „banka“. V průběhu času byl iterativně vytvářen návrh zavedení konceptu Bimodal IT, a to na míru, přesně podle požadavků získaných při konzultacích v bance. První návrh vycházel čistě z rešeršní práce a zkoumání materiálů společnosti Gartner, která koncept definovala. Další návrhy se pak čím dál více přibližovaly požadavkům banky. Došlo k několika zásadním změnám původního konceptu, které budou dále diskutovány. Výsledkem tohoto téměř dvouletého snažení je výsledný popis současného stavu řízení projektů, dále pak popis navrhovaného řešení, předpoklady nutné pro korektní fungování konceptu a akční plán, jak pokračovat v nastoleném trendu zavádění agility do banky.

2.2 Popis zkoumané organizace

Toto jsou agregované ukazatele, podle kterých si lze udělat představuo velikosti banky:

- Průměrný počet zaměstnanců v minulém roce: 8426
- Odhadovaný počte IT pracovníků: 500-1000
- Provozní výnosy: 28 miliard Kč
- Provozní náklady: 12 miliard Kč

- Čistý zisk: přes 12 miliard Kč
- Počet zákazníků: 1600000
- Počet dceřiných společností: 11
- Sama banka součástí nadnárodní skupiny: ano

2.2.1 Organizační struktura

Organizační struktura byla sestavena z veřejně dostupných informací na obrázku 2.1. Detailnější organizační struktura podléhá utajení. Je zajímavá ve více aspektech:

- Tři oddělení středního rozsahu nejsou součástí čtyř hlavních částí, ale jsou přímo podřízené řediteli. Jsou to oddělení lidských zdrojů, interního auditu a marketingu. Jsou to klíčová oddělení pro strategii a chod banky a ředitel si je řídí samostatně.
- Dále se banka dělí na pět hlavních částí, z nichž každou řídí jeden ředitel, který je součástí nejvyššího vedení banky (boardu).
- Neustále se zvyšující důležitost IT se zatím neprojevila do organizační struktury. IT zůstává vcelku nelogicky pod strategiemi a financemi. Tento typ organizačního uspořádání je spíše z doby, kdy se IT bralo jako nákladové středisko a oddělení financí mělo za úkol co nejvíce snižovat náklady na provoz IT systémů.

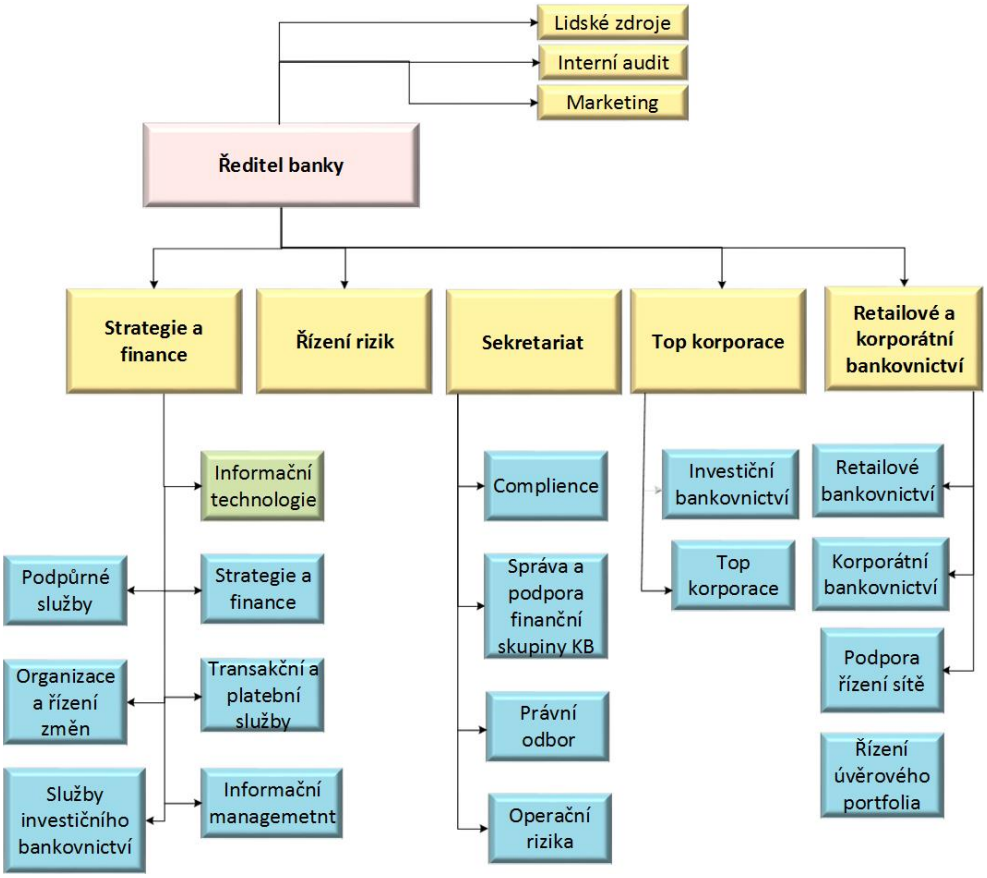
Jen pro představu - uvnitř IT je ještě dalších 5 úrovní manažerských pozic od ředitele po vedoucího nejmenší vývojové jednotky. Nejmenší vývojové jednotky mají 10 až 30 zaměstnanců.

2.2.2 Hierarchie řízení projektů v bance

Řízení dodávek softwarových celků probíhá na více úrovních. Diagram 2.2 přibližuje vazby mezi komponentami sloužícími k dodání nového produktu nebo softwaru. Tyto části jsou dále blíže popsány.

2.2.2.1 SW vývoj

Softwarový vývoj je plně v kompetenci IT oddělení podle toho, zda má na implementaci konkrétního typu software potřebné vědomosti a dostatek zaměstnanců, kteří by změnu mohli implementovat, ať už v rámci projektu nebo v rámci malých změn v liniovém řízení, realizuje změnu vlastními silami, pokud ne, implementaci deleguje na externí dodavatele a pouze řídí dodávky

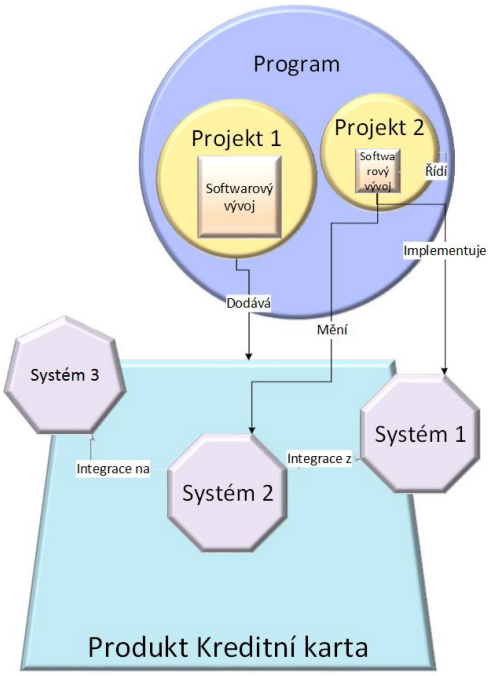


Obrázek 2.1: Organizační struktura banky

a kontroluje výstupy od externího dodavatele. Konkrétně si můžeme softwarový vývoj představit následovně. Techničtí lidé dostanou zadání, ve kterém je v požadované míře detailu popsáno, co má být výsledkem realizované činnosti a hledají vhodné cesty jak těchto výsledků dosáhnout. Z předchozích fází projektu by mělo být jasné, rozdělení na jednotlivé úkoly mezi týmy a zasažené systémy a programátoři a analytici si následně rozdělí jednotlivé úkoly a pracují na nich.

2.2.3 Projekt a projektové řízení

V současnosti funguje v bance velice složitá vodopádová metodika přibližně odpovídající tomu, jak by měl probíhat vývoj SW. V rámci ústupků směrem k agilnímu řízení bylo povoleno definovat požadavky na projekt ve formě uživatelských příběhů a poslední fáze, ve které se implementuje výsledný produkt, může probíhat iterativně v agilním módu. Výzva, které toto řešení čelí spočívá v tom, že je v předchozích fázích přesně dáno, co má být dodáno, s



Obrázek 2.2: Hierarchie a vazby mezi základními prvky ve vývoji nového software.

jakým rozpočtem a v jakém čase. Agilita pak v tomto případě pozbývá svého základního významu ve smyslu, že se nemůže dynamicky měnit rozpočet a požadavky podle požadavků byznysu. Pozitivně se dá vnímat iniciativa jednoho oddělení banky, které využívá agility k liniovému řízení pracovníků a implementaci malých změn do systému. Tento vývoj probíhá v plném agilním módu, požadavky byznysu jsou prioritizovány a implementovány podle jejich důležitosti.

Projekt v malé firmě a ve velké korporaci může mít naprosto odlišné parametry. Projekt v bance musí splňovat předem definované náležitosti:

- Jasně vymezený schválený rozpočet, definovaný rozsah a termín dodání
- Stanovené základní role: Projektový manažer, byznys vlastník, technický vlastník

Projekt má již před začátkem implementace jasně daný rozsah, který má splnit, fixně daný rozpočet a čas, během něhož má dané řešení dodat. Pokud v těchto měřítkách nastane změna, musí se obhajovat před komisí. Nesplnění těchto kritérií se bere jako neúspěch projektu a jeho projektového manažera.

Projekty v bance se dělí na dva typy:

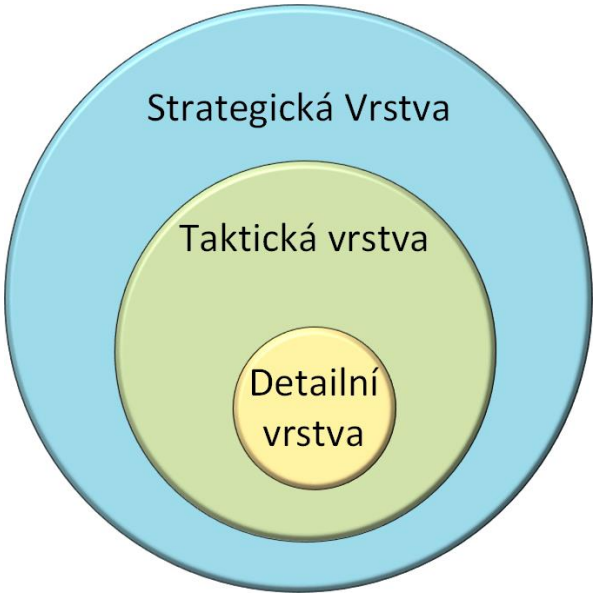
1. infra projekt
2. klasický (byznysový) projekt

Rozdíl mezi těmito dvěma typy spočívá v tom, že infra projekt přináší pouze technologickou změnu. Byznysový projekt naopak přináší změnu jak technologickou, tak i byznysovou. Jak si takovou byznysovou změnu můžeme představit? Např. zákazníci mohou používat místo podpisu biometrické údaje jako jsou otisk prstu nebo oční sken. Je to nový přínos pro zákazníka a změna má jak technologickou implementaci, tak se mění přístup k zákazníkovi a např. se musí změnit i legislativní podmínky upravující část podpisů.

Řízení projektů je jasně definované interními směrnicemi vydanými oddělením pro řízení projektů, dále pak detailnější projektovou metodikou. Obojí podléhá utajení, ale můžeme si obecně říci, jaké jsou vrstvy v definování řízení projektů a jak se tyto vrstvy ovlivňují.

2.2.3.1 Úrovně definované pro řízení projektů

Celé řízení projektů a návazná projektová metodika jsou definovány ve třech vrstvách. Vždy platí, že vyšší vrstvy definují principy, které musí nižší vrstvy dodržovat, ale mohou si definovat vlastní pravidla, která neporušují pravidla vyšších vrstev.



Obrázek 2.3: Vrstvy řízení projektů v bance

Strategická vrstva je vodopádová a je plně definovaná oddělením pro řízení projektů a definují se zde základní koncepty pro řízení projektů. V této vrstvě je definováno, že projekt musí mít vždy 4 fáze:

1. Počátek: V první fázi se musí definovat, jaká je vize projektu. Rozsah této vize by měl být 1-10 stránek.
2. Návrh: Ve druhé fázi se rozhodne, které části banky by změna zasáhla, v jakém rozsahu, vytvoří se odhady pracnosti a vypočítá se odhadovaná cena.
3. Implementace: Ve třetí fázi se navržené a odhadnuté změny rozdělí konkrétním řešitelům a změna se implementuje.
4. Zhodnocení: V poslední fázi se hodnotí, zda projekt přinesl vylepšení či úspory, které se od řešení čekaly a tato fáze pokračuje i po ukončení projektu

Mezi každou fází projektu je sestavena komise ze zaměstnanců IT oddělení, oddělení pro řízení projektů a zástupců byznysu. Tato komise má právo většinou hlasů rozhodnout, zda projekt půjde do další fáze nebo se ukončí. Dokonce i mezi 3. a 4. fází se může rozhodnout, že realizovaný projekt nebude využit.

Nejdůležitější metrikou v řízení projektu je čas jeho trvání. Banka se dlouhodobě snaží zkrátit délku projektů a změnu realizovat po částech, případně ve více navazujících projektech. Čas dodání se počítá pouze ve 2. a 3. fázi projektu

Taktická vrstva

Taktickou vrstvu definuje každá část banky za sebe a obsahuje například další role v řízení projektu specifické pro IT či marketing, určité základní postupy, nejdůležitější dokumenty společné pro každý projekt apod.

Pro náš případ je nejzajímavější taktická vrstva pro IT oddělení pokrývající první tři fáze. Poslední fáze - zhodnocení projektu je v pravomocích oddělení pro řízení projektů. V rámci taktické vrstvy si IT oddělení definovalo následující oblasti s tím, že každá oblast ještě může mít stanovených více rolí:

- Architektura - napomáhá definovat strukturu, použité technologie a návrhové vzory.
- Infra - oddělení infra definuje strukturu použitého serverového řešení.
- Finance - finance jen okrajově kontrolují, zda projekt nepřesahuje stanovený rozpočet.

- Projektové řízení - projektový manažer s přesahem do IT napomáhá koordinovat práce na projektu a je jediným místem, které může každý zaměstnanec IT kontaktovat, pokud chce o něm získat jakékoliv informace.
- SW vývoj - analytici a vývojáři, kteří se zapojují hlavně ve fázi implementace
- Provoz - Provoz přebírá a provozuje hotové řešení, spolupracuje na projektu po celou dobu projektu a pomáhá definovat požadavky hlavně na výkon serverů
- Oddělení softwarové bezpečnosti - kontroluje, zda byly splněny všechny požadavky na bezpečnost, zejména penetrační testy

Existují tři nejzásadnější dokumenty:

1. Vize projektu před začátkem druhé fáze.
2. Koncept řešení na konci druhé fáze.
3. Technická dokumentace na konci třetí fáze.

Detailní vrstva Detailní vrstvu má v gesci sám projektový tým. V této vrstvě se definuje počet členů týmu, jak dlouhé budou sprinty / sloty, jaká bude komunikace v týmech, jaký bude režim eskalace problémů. Zde je již nyní největší prostor pro využití agilního řízení. Tým si sám může říci, že bude vyvíjet na základě uživatelských příběhů definujících činnosti, které bude uživatel moci dělat se systémem, a ty se pak následně rozdělují mezi vývojáře a implementují. Vývojáři se mohou scházet na Daily standupech a koordinovat svoji práci. Může se používat Scrum board případně Kanban board. Implementace může probíhat ve sprintech plánovaných na 14 dní až měsíc.

2.2.4 Produkt

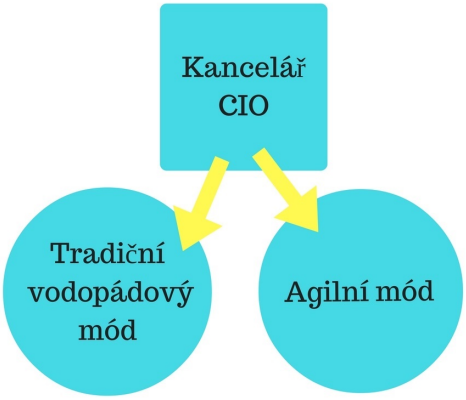
Produkt můžeme chápat dvěma způsoby, a to jednak jako byznysový produkt, například karta, hypotéka apod. Druhý, zjednodušený pohled pro IT, je vnímat produkt jako SW produkt, například Systém XY. Definice produktu je zcela zásadní pro jeho korektní definici, jelikož můžeme dále definovat Product Ownera. Z teoretické části víme, že tato role určuje priority ve vývoji produktu. Je pro nás v tomto bodě zásadní, zda se jedná o SW produkt nebo byznysový produkt. Ve zkoumané instituci se již dříve rozhodlo, že pro zjednodušení se o produktu bude mluvit jako o SW produktu. Problém byznys produktu spočívá v tom, že např. hypotéka může zasahovat do nespočtu SW produktů, nad kterými už dnes operují jejich IT vlastníci a BU vlastníci a domluva napříč tímto liniovým řízením by byla velice nesnadná, ne-li nemožná.

Při dalších krocích by se tato definice mohla upravovat, ale pro první krok je dostačující. O akčním plánu a detailech těchto kroků se budeme zmiňovat v této práci později.

2.3 Bimodal IT jako způsob zavádění agility

2.3.1 Obecný popis

Bimodal IT je koncept, který popisuje, že ke klasickému (vodopádovému) způsobu řízení (mód 1) se vytvoří alternativní agilní přístup k řízení projektů (mód 2) za účelem vyšší flexibility a přizpůsobivosti novým trendům v IT. K rozdělení tedy dojde na úrovni projektů, oddělení IT zůstane jednotné. Stávající vodopádový model se zachová a postupně se zavádí agilní mód, kvůli složitějším projektům, zejména integračně složitým na začlenění do stávajícího ekosystému banky a využití výhod jak klasického, tak agilního způsobu řízení. Pro plné využití agilního způsobu řízení projektů je nutné řídit agilní mód odděleně, ale zajistit, aby agilní i klasický mód sledovaly společné cíle, ani jeden z módů nesmí být privilegovaný. Již v základu konceptu se počítá s tím, že budou nastávat konflikty mezi agilním a vodopádovým světem. Tyto konflikty by měla řešit nově vzniklá kancelář CIO. Kancelář CIO kontroluje oba módy, rozhoduje, kterým způsobem se budou projekty v bance řídit a dohlíží na rovnocennost a bezproblémovost komunikace. Řeší případné konflikty a nesoulady. Graficky je základ konceptu zobrazen na obrázku 2.4.

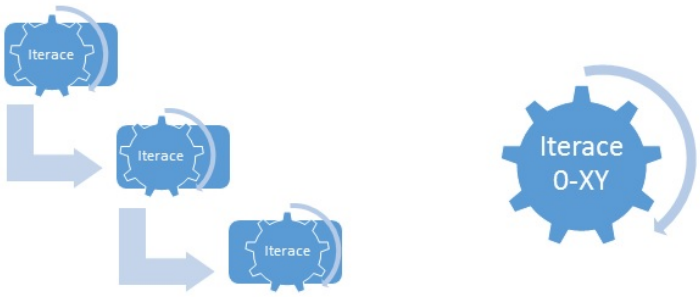


Obrázek 2.4: Základní struktura konceptu Bimodal IT.

2.3.2 Proč úplně oddělit oba módy

V prvních fázích zavádění konceptu Bimodal IT je nutné využít agilní přístup uvnitř vodopádového, konkrétně v softwarovém vývoji. Proč tento přístup

nepřináší takový přínos, jaký bychom čekali? Pokud postupně pracujeme na projektu nebo produktu tak, že si nejdříve vydefinujeme vodopádově, jak by měl vypadat, definujeme celý návrh a použité technologie a následně se přesuneme do fáze implementace toho, co jsme navrhli, je nemožné přijmout zásadní změnu zadání a vrátit se do předchozí fáze už z podstaty vodopádového stylu řízení. Přicházíme tak o zásadní benefit agility, totiž reakci na změny. Pokud celý produkt nedefinujeme na základní úrovni, nepokusíme se implementovat prototyp, nikdy nemůžeme využít agilitu k „nahlédnutí do budoucnosti“ a ověření, zda má daná technologie, produkt nebo systém smysl. Musíme si umět přiznat, že jsme udělali chybu, poučit se z ní, ohlédnout se na předchozí iterace a změnit svůj přístup podle toho, co vyšlo a co ne. Nesmíme se bát, že něco „nevyjde“ nebo že „uděláme chybu“. Chyba není neúspěch, protože i zjištění, že některé technologické novinky nejsou implementovatelné v prostředí konkrétní společnosti, je hodnotné. Všechny problémy, které by mohly nastat, zjistíme v kratším čase. Postupným definováním požadavků zajistíme, že budeme implementovat to, co implementovat chceme my i naši zákazníci či koneční uživatelé.



Obrázek 2.5: Agilní řízení pro fáze vodopádového projektu versus plná agilita

2.3.3 Byznysové / IT požadavky

1. Obecné

   - **Zkrátit délku projektu (v čase).** Bylo rozhodnuto, že nejdůležitější metrikou v projektech je čas. Je nutné proto celkově projekty zkrátit.

     Jednou z možností by bylo zkrátit projekty příkazem a implementovat pouze menší části. To by však vedlo často pouze k rozdělení projektu do tří menších a zkrácení by proběhlo jen opticky. Výhoda tohoto řešení je v tom, že například z dlouhého projektu by mohly být schváleny jen dvě třetiny. Co když ale nejdůležitější je právě poslední část projektu?

Druhá korektnější možnost představuje využít Bimodal IT a pomocí agilního módu se snažit zjistit možnosti využití nových technologií, hledat nové cesty a efektivněji využívat čas.

- **Zlepšit reakci dlouhotrvajících projektů na nepředvídatelné změny v bance v čase projektu.** Toto je podobná situace. Obecně je v bance problém s dlouhotrvajícími projekty, protože jsou tak velké, že jsou teoreticky neřiditelné a často se nesplní požadavky na čas, rozpočet i kvalitu dodávek. Agilní mód by mohl pomoci řešit problémy a posouvat tyto velké projekty kupředu.
- **Lépe spolupracovat v rámci projektu.** Agilita napomáhá zlepšovat vztahy a krátkodobý iterativní vývoj snižuje rizika dodání výsledku práce, který nebyl požadován.

2. Na straně byznysových částí banky

   - **Současné IT projekty nedrží krok s potřebami byznysu.** Často nastává problém, že procesy, které se musí na IT oddělení dodržovat, trvají určitý čas a je potřeba jasně definovat zadání. To není pro všechna oddělení banky přijatelné. Jediným řešením je obejít IT a zajistit jiné zdroje, na financování externí firmy, která tyto požadavky nemá a reaguje okamžitě.
   - **Komunikace s IT je složitá, byznys nemůže během projektu měnit požadavky.** Tento problém by opět vyřešil agilní přístup. Jedna ze zásadních výhod agility je neustálá komunikace s byznysem, k němuž směřuje nový požadavek na časové možnosti. Všechny schůzky jsou ale koncipované tak, aby se mimo agilní schůzky nemusely konat žádné statusy ani jiné schůzky, kde by se byznys čistě informoval o stavu projektu. Byznys by mohl kdykoliv měnit zadání. Změnit původní řešení by pak stálo peníze, ale určitě méně, než kolik by stálo začít projekt znovu.
   - **Některé projekty je jednoduší a levnější outsourcovat než zadat vlastnímu IT.** Problém s častým outsourcingem by se dal pomocí agility také vyřešit. Pokud by se agilně nenašla cesta jak neznámý projekt realizovat, mohl by být i tak outsourcován.

3. Na straně IT

   - **IT trvá 3 měsíce pouze schvalování a administrativa k projektům.** Tím je myšleno, že pokud by se měl schválit prázdný projekt, trvá v průměru tři měsíce, než se sestaví komise, které schvalují přechod mezi jednotlivými fázemi, než by všichni účastníci projektu vyplnili nutné dokumenty (jen prázdné) a než by byla uzavřena 3. fáze projektu, některé projekty by byly za tři měsíce realizovatelné v agilním módu.

- **Současná metodika není vhodná pro menší a inovativnější projekty.** Když je potřeba udělat například malý test, zda daná technologie je využitelná v prostředí banky (PoC), je potřeba malou část kódu implementovat v malém rozsahu a vyzkoušet, zda je to přijatelná možnost. Tyto aktivity jsou často hrdinstvím jedinců, nikoliv aktivita řízená projektově. Tyto aktivity by se mohly řídit agilním módem.
- **Náklady na IT jsou vyšší kvůli nízké úspěšnosti projektů (až v pozdější fázi projektu se zjistí, že projekt není realizovatelný).** Opět se objevuje problém v nemožnosti ve vodopádovém řízení udělat chybu a vrátit se do předchozích fází. V současnosti se problém řeší tak, že například ve fázi implementace se provádí analýza, jak by se dal daný problém obejít, případně se úplně zruší implementace určitých funkcionalit systému.

2.3.4 Koncept řešení - návrh cílového stavu

Po úspěšné realizaci konceptu Bimodal IT v celé bance je podle akčního plánu představeného později v práci, očekáván stav, kdy nový mód 2 je nezávislý na mód 1. Jsou sjednoceny cíle obou módů, a ačkoliv se oba módy neovlivňují, navenek se prezentují jako jedno IT. Všichni zaměstnanci chápou důležitost obou módů a nevidí mezi nimi rozdíl jak v kvalitě, kariérním růstu, tak v odměnách za dobře odvedenou práci. Rozdíl mezi mód 1 a mód 2 je pouze v jiném způsobu řízení a přístupu k práci na projektu. mód 1 je zaměřen na dlouhodobější projekty se složitou integrační architekturou, mód 2 je zaměřen na rychlejší dodání inovativního projektu, se zaměřením na úzkou, intenzivní spolupráci s byznysem. Vznikla nová metodika pro agilní mód řízení projektů a pokud nastanou komplikace v komunikace mezi módy, řeší je kancelář CIO.

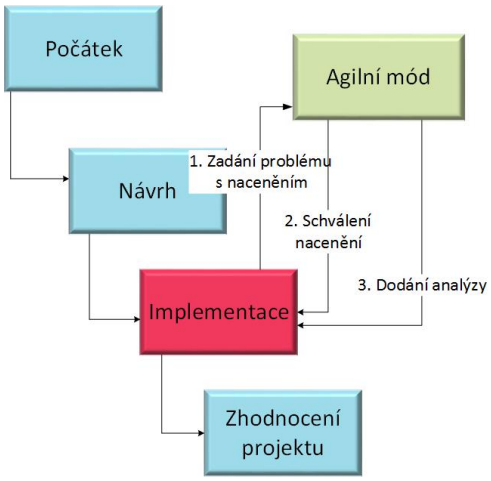
2.3.5 Součinnost obou módů

V případě, že by na projektu vedeném pomocí mód 1 bylo nutné využít schopností mód 2, došlo by k rozhodnutí pomocí indikačních parametrů, že se projekt bude řídit kombinovaně. Zde je nastíněno, jak by taková spolupráce mohla probíhat.

První možnost počítá s tím, že před začátkem projektu se agilně vytvoří prototyp, který bude sloužit jako důkaz pro klasický projekt o jeho realizovatelnosti. Tento přístup by měl zamezit tomu, že se až při implementaci klasického projektu zjistí, že daný projekt nelze realizovat.

Druhá možnost je, že se klasicky řízený projekt dostane do „slepé uličky“ a požádá o pomoc agilní mód, který se pokusí přistoupit k problému jinak a v několika iteracích vytvořit výstup, který napomůže klasickému projektu v

dalším postupu, např. projekt chce využít nové technologie, se kterými nemá žádné zkušenosti. Využije možností mód 2 a tým sestavený z těchto lidí v krátkém čase prozkoumá výhody a nevýhody použití technologie. Vše samozřejmě musí být předem konzultováno se zaměstnanci, kteří by v agilním módu fungovali, a řádně naceněno. Agilní mód pak může říci, že za daný omezený rozpočet zanalyzuje obě technologie a pokud se to podaří, naplánuje i realizaci důkazu, jak by technologie v rámci banky mohly fungovat. Průběh pak bude agilně plánován a kvalita analýzy bude záležet na tom, jak rychle půjdou práce kupředu. Také jinak rozsah dodané práce nelze v agilním módu předem garantovat.



Obrázek 2.6: Řízení projektu kombinací módů

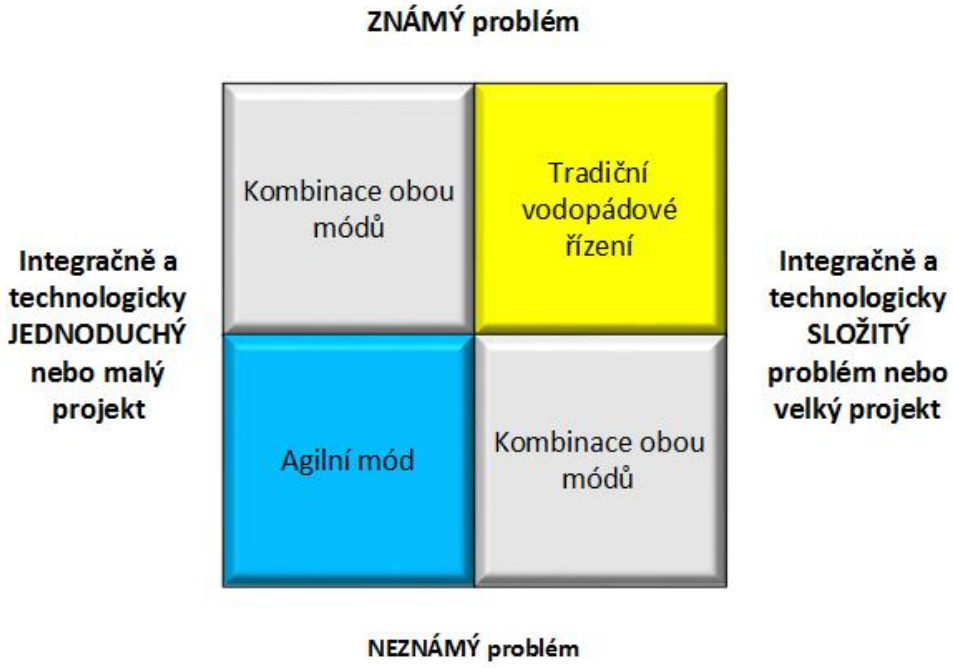
2.3.6 Rozhodování o způsobu realizace projektu

Na základě následujícího postupu bude možné jednoznačně určit, zda daný projekt je vhodné řídit tradičně, agilně nebo kombinací obou přístupů. Kombinace obou přístupů bude možná pouze v případě splnění pravidel v části práce 2.3.5. O způsobu řízení projektu bude rozhodovat kancelář CIO. Rozhodnutí se provede na základě následujících parametrů: integrační složitost a velikost projektu versus známost problému. Mohou nastat čtyři varianty:

- Velký nebo integračně složitý problém a zároveň neznámá oblast - naprosto nová technologie a zároveň vysoká prointegrovanost s ostatními systémy. Příkladem takového projektu by mohla být nová integrační platforma typu ESP, která bude zjednodušovat komunikaci mezi systémy, ale prozatím nikdo v bance nemá s touto technologií zkušenosti. Zde je doporučen kombinovaný přístup, kde v rámci agilního módu se budou zaměstnanci učit pracovat s novou technologií a tyto poznatky

budou sdílet s kolegy, kteří budou řešit zásadní změny v architektuře a integraci.

- Velký nebo integračně složitý problém a zároveň známá oblast. Jedná se o nejvhodnější oblast pro tradiční vodopádové řízení. Jasně definovaná doména a velký integrační projekt.
- Malý nebo integračně jednoduchý problém a zároveň neznámá oblast. V předchozím způsobu řízení se často tyto problémy outsourcovaly. V novém způsobu je to přesně oblast pro plně agilně řízený projekt.
- Malý nebo integračně jednoduchý problém a zároveň známá oblast. Zde je prostor pro součinnost módu, agilní přístup nepřinese mnoho nového, ale na malé změny je vhodnější. Naopak tradiční přístup může poučit zaměstnance fungující agilně o známé doméně.



Obrázek 2.7: Parametry, podle kterých se bude posuzovat, jakým způsobem projekt řídit

2.3.7 Výhody Bimodal IT

Benefity, které koncept Bimodal IT přinese odpovídají byznys/ IT potřebám. Bimodal IT je jeden ze způsobů, jak se vyrovnat s stínové IT, které trápí

banku, jelikož byznys si radši nakupuje hotové řešení (např. formou SaaS) než implementaci vlastním IT. Pokud se byznys jednotkám banky dobře vysvětlí přínosy Bimodal IT pro byznys, dosáhne se lepší komunikace IT vs. byznys a celý koncept bude pro banku vysoce rentabilní. Aby se mohla zlepšit komunikace, snížit náklady na projekty a snížit čas dodání jednoduše implementovatelných projektů, je nutné zavést mód 2. Tím se zvýší důvěra byznys vzhledem k IT a sníží celkové náklady banky na IT projekty.

2.3.8 Finanční rozvaha agilního řízení

O peníze jde vždy až v první řadě. Při splnění předpokladů agility v bodě Nástrojů pro kontinuální integraci 2.4.3 bude mít agilně řízený projekt stejný rozpočet, jako kdyby se řídil tradičním způsobem. Pokud tomu tak nebude, narostou náklady na nasazování řešení. Cílem agility je snížit náklady na nevyužívané funkce. Agilita často mění zadání a nevyužívá všechny části implementovaného kódu. Výhodou je, že výsledek vývoje přesně odpovídá tomu, co chce zákazník, v tomto případě byznys. Možná není implementace na objem levnější, ale je implementováno jen to, co přináší hodnotu. Vývoj se tedy automaticky nezlevní, ale reálně dojde k vyššímu využívání dodaných funkcí systému. Velice zjednodušená manažerská kalkulačka vytvořená autorem s názvem Cost calculator verze 3 je na přiloženém CD.

2.3.9 Změny oproti původnímu návrhu Gartner

Původní koncept od společnosti Gartner počítá s rozdělením IT byznys produktů. Jinak řečeno produktem by mohlo být internetové bankovnictví a celé týmy by pak mohlyna něm pracovat na základě Product Backlogu celého tohoto produktu. Takto to však v bance nefunguje a jednotlivé týmy se často specializují na jeden konkrétní systém, který pak často poskytuje služby více byznysovým produktům. Další, co v bance chybí, je dlouhodobý rozvoj těchto produktů. To v praxi znamená, že projekt dodá produkt, na kterém se do budoucna pracuje jen velmi málo a jeho údržba je bez financí z projektu složitá. Tyto dva detaily v důsledku znamenají, že agilní vývoj se nedá dlouhodobě využít k implementaci a následné údržbě a je nutné rozdělit banku podle projektů a ty řídit buď vodopádově nebo agilně.

2.3.10 Rizika navrhovaného řešení

Ať už se jedná o projekt nebo jakoukoliv jinou aktivitu, je dobré stanovit jak přínosy, pokud budou cíle aktivity naplněny, tak na druhé straně rizika, která mohou nastat. Je vhodné se předem se připravit na situaci, že opravdu nastanou u nejpravděpodobnějších rizik a u těch s největším dopadem či u kombinace obojího. Pokud se rozhoduje o projektu, často se rizika vyčíslují v penězích a pokud má projekt finanční přínos, počítá se s tím, že může přinést hodnotu plus minus hodnotu rizika.

2.3.10.1 Metodika stanovení rizik a protiopatření

Pro analýzu rizik byla využita metodika OWASP Risk Rating Methodology od neziskové společnosti OWASP. Tato metodika se skládá z 6 bodů:

1. Identifikování rizik - čistě zjistíme rizika, která by mohla nastat a sestavíme jejich seznam.
2. Určení míry pravděpodobnosti rizika - pravděpodobnost rizika je důležitá při rozhodování, zda je riziko reálné. V našem případě rizika projektu a jejich pravděpodobnost nelze empiricky zjistit. Tím pádem musíme použít expertní odhad pravděpodobnosti.
3. Jak velký dopad bude mít riziko, když nastane - jakkoliv je riziko pravděpodobné, pokud na nás nemá žádný dopad, nemusí nám vadit, ani v případě, že skutečně nastane. Dopad určuje vzniklé škody.
4. Dopad a pravděpodobnost dopadu se protnou operací podobnou kartézskému součinu podle níže uvedené tabulky v úroveň rizika, ta je tedy exaktně vypočítávána.
5. Rozhodnutí, kterým rizikům předcházet - pokud je úroveň důležitosti rizika vysoká nebo kritická musíme se jimi zabývat. Nejlepší postup je stanovit, jak se v akčním plánu s těmito riziky vypořádáme pokud nastanou nebo realizovat protiopatření, abychom u rizik snížili pravděpodobnost, dopad nebo je úplně eliminovali.
6. Rozšíření modelu o doplňující měřítka - nebude realizováno

2.3.10.2 Identifikování rizik

Během konzultací autor identifikoval následující rizika:

Absence spolupráce středního managementu Koncept má podporu ve vrcholovém řízení a stejně tak se postupně zavádí agilita na nejnižším detailním stupni řízení projektů. Aby ale koncept bezchybně fungoval, musí o něm být přesvědčen i střední management. Výzva v tomto bodě bude vycházet ze skutečnosti, že střední management se často sestavuje z dlouhodobých zaměstnanců, kteří v bance působí ve velké míře více než 10 let. Lze proto očekávat, že jenom menší část manažerů bude agilitu hned ze začátku podporovat.

Absence spolupráce zaměstnanců Změna je vytvářena v zájmu běžných zaměstnanců banky. Pokud však sami nebudou o změnu stát a budou se jí bránit, mohou získávání přínosů z konceptu zpomalovat. Případ z praxe: Kolega, který je v bance 10 let a má významné postavení, i když nikoliv v hierarchii, ale těší se velkému respektu svých kolegů řekl, že se žádných standupů zúčastňovat nebude. Nikdo mu význam těchto schůzek nebyl schopný vysvětlit,

a tak se jich tento kolega nezúčastňoval. Neexistuje za něj ale adekvátní náhrada, takže rozvázání pracovního poměru nepřipadá v úvahu. Takové situace nakonec působí jako špatný příklad pro ostatní. Pravidla by měla platit pro všechny stejně.

Nedostatečná spolupráce oddělení pro řízení projektů. Hlavně v krocích 2 a 3 je nejdůležitějším bodem při zavádění konceptu úzká spolupráce s oddělením pro řízení projektů. Zde nepůsobí žádní začátečníci. Většinou se jedná o manažery s více než 10-letou praxí. Pokud nebudou chtít spolupracovat na vytváření nové metodiky pro agilní mód a neuvidí jeho přínosy, budou se jen těžko přemlouvat ke spolupráci.

Nemožnost rozhodnout se, jakým módem projekt vést Toto riziko je dosti teoretické. K rozhodování jak projekt vést je dán jasný nástroj v sekci 2.7. V krajním případě by se mohlo stát, že projekty budou hraniční, a toto riziko tak dává smysl jen tehdy, pokud by těchto hraničních projektů bylo mnoho nebo pokud by nástroj nebyl sestaven v dostatečné míře přesnosti pro rozhodování.

Nedostatek zkušeností s vedením agilních projektů. Oproti klasické vodopádové metodice je agilní řízení v bance kratší dobu a z toho pramení rizika nedostatku zkušeností. Rozhodně je toto riziko vyšší u projektových manažerů, kteří by si museli zvykat na nový způsob řízení projektů.

Špatná komunikace mezi módem 1 a módem 2. Oba módy představují naprosto odlišné světy. Jejich srovnání bylo vysvětleno v teoretické části. Z toho pramení neshody ohledně termínů, v komunikaci a jejích formách, v dodávaných dokumentech nebo používaných výrazech či rolích. Reálně se v zavedení agility v kroku 1 níže uvedeného akčního plánu ukazuje, že nejčastěji bývá problém s termíny. Zatímco agilní mód je v třetí iteraci, navrhl si základ aplikace a rozhoduje se, co bude dále implementovat, vodopádový model právě plánuje projekt a jeho termíny a potřebuje vědět, jaké služby agilního projektu budou dostupné za půl roku. Agilní projekt pak není schopen tyto odhady sestavit. Tyto problémy se často řeší tak, že agilní projekt něco slíbí a snaží se svoji cestu upravit tak, aby se požadavky projektů střetly.

Nezájem o agilitu ze strany byznys jednotek. Každý se sžívá s agilitou jiným tempem. I v byznysových jednotkách bude mnoho zaměstnanců, kteří budou proti agilitě. Nejčastější problém bývá, že se s ní zaměstnanci ani nechtějí seznamovat nebo samy byznys jednotky nejsou motivované mimo svoji pozici ještě IT „pomáhat“ vyvíjet SW. Dalším důvodem je nedůvěra ve schopnosti IT. Tomuto principu se říká stínové IT, kdy byznys jednotky nevěří, že by IT bylo schopno dodávat, co od něj byznys požaduje, a v čase, kdy

Tabulka 2.1: Míra pravděpodobnosti rizik

Nespolupráce středního managementu	50 %
Nespolupráce zaměstnanců	30 %
Nedostatečná spolupráce oddělení pro řízení projektů	55 %
Špatná komunikace mezi módem 1 a módem 2	80 %
Nezájem o agilitu ze strany byznys jednotek	75 %
Problémy s financováním agilních projektů	40 %
Dvě firemní kultury v jedné společnosti	100 %
Nemožnost rozhodnout se, jakým módem projekt vést	20 %

to byznys požaduje. Za svoje peníze si nakoupí externí firmy, které mu dodávají hotové řešení na klíč, na něž nemusí definovat požadavky, řešit projekt a řídit vývoj. Řešení je hned, a pokud se byznys jednotkám nelíbí, je okamžitě upraveno přesně v duchu agility.

Problémy s financováním agilních projektů. Financování projektů mód 2 nebude stejné jako mód 1. Nebude možné ve fixním rozpočtu přesně definovat, co za rozpočet bude dodáno. S tímto problémem se bude nutné v dalších fázích rozumně vyrovnat a pokud ne, je to jednoznačně jedno ze zásadních rizik do budoucna.

Dvě firemní kultury v jedné společnosti Toto riziko je definované už v základu konceptu Bimodal IT. Pro banku by roztříštění firemní kultury mohlo znamenat zásadní riziko.

2.3.10.3 Určení míry pravděpodobnosti rizika

Autor po více než rok trvající spolupráci expertně zhodnotil, jak mu připadají jednotlivá rizika pravděpodobná. Vzhledem k množství konzultací se zaměstnanci napříč bankou by tyto odhady měly být dostatečně přesné.

2.3.10.4 Jak velký dopad bude mít riziko, když nastane

U rizika nezáleží jen na tom, jak je pravděpodobné, ale také na míře jeho dopadu na projekt či společnost. Pokud se riziko týká managementu nebo zasahuje výrazně do práce každého zaměstnance, dopad je vyšší, pokud ne, dopad je nižší. Speciální je případ dvou firemních kultur v jedné společnosti. Tímto jsou zasaženi všichni zaměstnanci, ale koncept Bimodal IT má nástroje, jak riziko mitigovat. Pro názornost si uvedeme příklady, jak by nízký, střední a vysoký dopad mohl vypadat:

Tabulka 2.2: Dopady rizik na koncept

Nespolupráce středního managementu	Střední
Nespolupráce zaměstnanců	Střední
Nedostatečná spolupráce oddělení pro řízení projektů	Střední
Špatná komunikace mezi módem 1 a módem 2	Vysoký
Nezájem o agilitu ze strany byznys jednotek	Vysoký
Problémy s financováním agilních projektů	Střední
Dvě firemní kultury v jedné společnosti	Střední
Nemožnost rozhodnout se, jakým módem projekt vést	Nízký

Tabulka 2.3: Výpočet úrovně rizik

Úroveň rizika				
Dopad rizika	VYSOKÝ	Střední	**Vysoká**	**Kritická**
	STŘEDNÍ	Nízká	Střední	**Vysoká**
	NÍZKÝ	Žádná	Nízká	Střední
		NÍZKÁ	STŘEDNÍ	VYSOKÁ
		Pravděpodobnost rizika		

- Nízký dopad - neohrožuje výrazně životaschopnost konceptu, maximálně může znamenat krátké prodloužení časového plánu
- Střední dopad - ovlivňuje zásadněji měřítka procesu, v krajních případech může zapříčinit neúspěch
- Vysoký dopad - pokud nastane, může silně ohrozit zavádění celého konceptu

2.3.10.5 Úroveň důležitosti rizika

Úroveň důležitosti rizika, slouží pro konečné rozhodnutí, zda se danému riziku bude nutné vyhnout zavedením protiopatření či nikoliv. Stanovuje se průnikem pravděpodobnosti rizika s jeho dopadem podle následující tabulky definované v metodice OWASP.

Když si podle tabulky určíme, kterými riziky bychom se měli zabývat, bude to vypadat následovně:

2.3.10.6 Navrhovaná protiopatření

Z analýzy rizik jasně vychází, že bude nutné předcházet třem rizikům s vysokou úrovní rizika. Podle [28] máme čtyři možnosti, jak reagovat na rizika:

Tabulka 2.4: Vypočtená úroveň rizik

Nespolupráce středního managementu	Střední
Nespolupráce zaměstnanců	Střední
Nedostatečná spolupráce oddělení pro řízení projektů	Střední
Špatná komunikace mezi módem 1 a módem 2	Vysoká
Nezájem o agilitu ze strany byznys jednotek	Vysoká
Problémy s financováním agilních projektů	Střední
Dvě firemní kultury v jedné společnosti	Vysoká
Nemožnost rozhodnout se, jakým módem projekt vést	Nízký

- Akceptace rizika - víme o riziku, monitorujeme jeho průběh - na současné úrovni nezajišťujeme protiopatření.
- Mitigace - snižování dopadů a pravděpodobnosti zaváděním protiopatření
- Přenesení na jiný subjekt - například pojištění nebo zajištění řešení externí firmou
- Vyvarování se - úplné vyřešení - například odstraněním zasažené komponenty projektu, produktu nebo plánu

Špatná komunikace mezi módem 1 a módem 2 Navrhované řešení: Snížení pravděpodobnosti rizika. V konceptu Bimodal IT řeší tento problém kancelář CIO , která do budoucna bude řídit souběh obou módů. Další možnost, jak se vyhnout problémům tohoto typu je proškolení obou módu ohledně druhého módu, definování sporných bodů a až v případě nevyřešení bude nutné předložení kanceláři CIO k vyřešení. Ta bude tedy působit jako arbitr v případech, že se nedohodnou módy mezi sebou.

Nezájem o agilitu ze strany byznys jednotek Navrhované řešení: Snížení pravděpodobnosti rizika. Je nutné přesvědčit byznys jednotky, že agilní mód je schopný též reagovat na změny a přistupovat k vývoji SW inovativně a to i v oblastech, které nejsou tak známé, což před zavedením konceptu neuměl. Sníží se tak prostor pro stínové IT.

Dvě firemní kultury v jedné společnosti Navrhované řešení: Akceptace. Jelikož při nasazení konceptu Bimodal IT se počítá jak s agilním, tak s tradičním způsobem řízení projektů, riziko nastane na sto procent a musí se dobře řídit a kancelář CIO musí být připravená koordinovat problémy mezi těmito dvěma světy. Riziko patří k obecným problémům konceptu a i samotný koncept od společnosti Gartner se jím zabývá.

2.4 Předpoklady nutné k tomu, aby celý koncept fungoval

K tomu, aby bylo zavedení konceptu Bimodal IT bezproblémové, musíme při zavádění do tak komplexního prostředí, jakým je velká banka podpořit jeho nasazování vhodným způsobem a musí být jasně definováno, co pro nás znamenají základní pojmy, a splněny předpoklady, které v konečném důsledku zajistí, že bude agilita fungovat korektně. Některé z předpokladů nejsou nutné k využití konceptu Bimdoal IT, některé ano. Ale obecně platí, že bez nich bude Bimodal IT zaváděno a využito obtížněji a nebude mít přínosy, jaké by mohl mít.

2.4.1 Funkční týmy

Pro co nejlepší využití agility v rámci týmů je nutné, aby každý z týmu mohl pracovat na všech úkolech z Product Backlogu. Abychom tuto vlastnost mohli zajistit, musí být členové týmu navzájem zastupitelní. Nesmí se specializovat na konkrétní činnost jako je vývoj, testování, analýza, ale ani na konkrétní úzkou část systému. V minulosti se zaměstnanci učili jednu malou oblast, ve které byli naprostí odborníci, ale dnešní doba je mnohem pružnější. Mnoho věcí se rychle mění a jeden systém může být nahrazen jiným. Občas může být potřeba udělat 20 analytických úkolů a ani jeden vývojářský. Tým se musí umět maximálně přizpůsobit a využít svůj čas efektivně.

2.4.2 Změna podnikové kultury

Často se při práci ve velké společnosti setkáte s větami typu: Kdy budete mít implementovanou komponentu A systému XY? Jakou pracnost budete mít na projektu X za jeden a půl roku? Jak se bude dál vyvíjet rozvoj integrační komponenty? Všechny tyto otázky napovídají, že pracovníci ještě neadoptovali agilní principy. Problém není v tom, že by se neplánovalo, avšak plánování probíhá průběžně a není vždy jasné, která funkce se bude implementovat dříve a která později.

Pokud bychom tyto věty upravili do podnikové kultury, ve které je agilita úspěšně zavedena, mohli bychom je změnit asi následovně. Potřeboval bych funkci A vašeho systému XY, máme na její implementaci vymezené prostředky, zkuste, jestli by v ve třech sprintech nešla implementovat funkce A. Není to kritická změna, ale rád bych poprosil Product Ownera, aby jí dal vyšší prioritu, protože náš systém je úzce napojený na váš a až v příštích čtvrtletích budete potřebovat pomoci od nás, vyjdeme vám také vstříc a určíme prioritu vašich požadavků nejvýše, jak jen to bude možné.

2.4.3 Nástroje pro kontinuální integraci

Nástroje pro kontinuální integraci jsou původem z metodiky Extrémní programování. Při iterativním vývoji dochází k zásadní změně v implementaci a nasazování. Zjednodušeně se častěji nasazuje a testuje v kratších cyklech a přichází se tak rychleji na chyby, případně mohou být rychleji zaváděny změny. Stinnější stránka tohoto konceptu je, že v současné době je nasazení (release) případně i do produkce nelehká a nákladná záležitost. Aby se agilita značně neprodražila, je nutné implementovat nástroje na automatický build, nasazení a testování.

V bance se autor práce účastnil aktivity, která realizovala Proof of concept takového řešení založeného na orchestrování Jenkinsem, nasazením nástrojem Urban Deploy a testy v Team City. Nasazování pak probíhá automaticky každý den, a když se vývojáři vrátí druhý den do práce, hned vidí, zda je kód v pořádku a případně implementují opravy. Nasazování do produkce pak probíhá na stisk jednoho tlačítka, kdy nastavení pro testovací prostředí, které se otestuje jako funkční, se použije i pro produkci, jen je před nasazením zkontrolováno oddělením provozu. Na obrázku 2.8 je přehled implementovaného konceptu podle bankovního standardu. Pojďme si projít celý proces.

Jenkins Jenkins celý proces takzvaně orchestruje. To znamená, že celý proces řídí. Nejdříve získá kód z GITu, spustí statickou analýzu kódu v Sonar Qube, spustí buildovací skript v Gradle, získá spustitelnou verzi kódu, kterou uloží do Nexusu a aktivuje nasazení na aplikační server pomocí nástroje Urban Code Deploy. Pokud některý z kroků skončí chybovým hlášením Jenkins vypíše detailní log o chybách a odešle emailem notifikaci vývojářům, kteří se v nejkratší době snaží problém opravit. Další technologie budou dále blíže popsány

Gradle Proces začíná vytvořením buildu pomocí skriptovacího jazyka Gradle [9]. Build je spustilná část kódu opatřená zpravidla verzí. Samotný kód se získává z verzovacího systému GIT. Gradle je spouštěn integračním serverem Jenkins, který celý proces řídí. Když je build hotov, vytvořený spustitelný kód se uloží opět do GITu.

GIT GIT je známý verzovací systém, ve kterém se nejčastěji provádí příkaz commit, kterým se původní kód obohatí o nové části a odstraněný kód se smaže. Verzovací se nazývá proto, protože s každým dalším commitem si ukládá původní verzi a v případě chyby v nově napsaném kódu se k ní může vývojář kdykoliv vrátit.

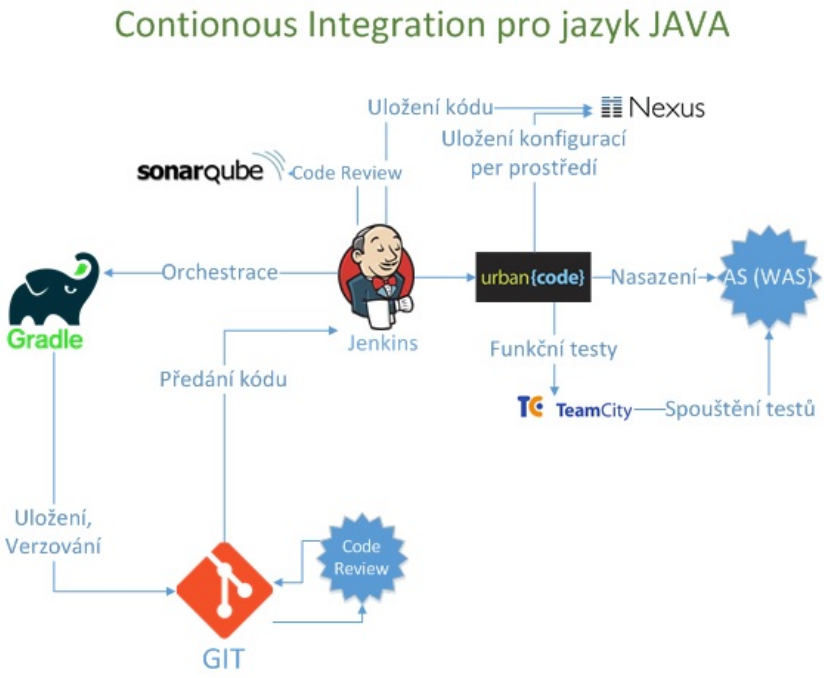
Sonar Qube je nástroj pro statickou analýzu kódu. To znamená, že kód se nespouští, ale jen ho Sonar prochází a podle jeho předchozího nastavení ohod-

nocuje kód, zda jsou v něm prohřešky proti doporučením k psaní kvalitního kódu.

Urban Code je nástroj pro automatické nasazení funkčního bezchybového kódu na aplikační server. Aplikačním serverem je zpravidla pro jazyk Java IBM Websphere aplication server. Ve všech případech bankovních aplikací fungují aplikace ve více prostředích. Pro každé prostředí má Urban Code připravené jiné konfigurace serveru. Tyto konfigurace se ukládají do Nexusu popsaného níže.

Nexus Nexus je inteligentní úložiště, které umožňuje ukládat kód, graficky zobrazovat uložený kód a zachovat všechny relativní cesty při migraci. V Nexusu se kromě spustitelné verze kódu ukládají i nastavení technologie Urban Code pro každé prostředí.

Team City Team City je nástroj pro vytváření a správu funkčních testů. To znamená, že kód se spouští a program fyzicky procházejí jeho části a testuje funkčnost tlačítek, obrazovek či filtrů.



Obrázek 2.8: Přehled konceptu Nástrojů pro kontinuální integraci. Zdroje obrázků pro konkrétní technologie: [9, 10, 11, 12, 13, 14, 15]

2.4.4 Dostatek Product Ownerů

2.4.4.1 Motivace Product Ownerů

Obsadit roli Product Ownera je složité. Zákazníky jsou totiž lidé z byznysových jednotek banky, kteří mají za úkol systémy obsluhovat a nevnímají jako důležitou náplň své práce pomáhat IT s vývojem těchto aplikací. Jak jsme si řekli výše, spolupráce byznysových jednotek je pro agilní řízení zásadní, jelikož hlavně skuteční uživatelé napomáhají definovat, co se se systémem dělá jednoduše a v čem mají naopak problém. Vyvinutý SW pak lépe odpovídá jejich potřebám, ne pouze představám návrhářů systémů, kteří dedukují, k čemu by systém mohl sloužit.

2.4.4.2 Kde hledat Prodcut Owners

V současnosti je několik modelů, jak pracovat s PO. Nejčastěji používaný je PO z byznysových jednotek. Tj. když se vyvíjí aplikace pro marketingové účely, je PO z oddělení marketingu. Další možnost je, že někdo z vývojového týmu, který má dostatečnou znalost systému se zvolí jako PO a prioritizuje práce na SW jen z IT pohledu. Další nevalidní možnost je, že PO úplně chybí. Jedna ze zvažovaných možností do budoucna je, že by byli v rámci banky profesionální PO, kteří by se vždy naučili, co systém dělá, bavili se s byznys jednotkami a definovali a prioritizovali požadavky na systém. Obecně ale platí, že každý jeden PO má naprosto odlišný přístup ke své práci a ani jedna varianta nezaručuje, že PO bude fungovat správně.

2.4.4.3 Doporuční do pozdějších fází akčního plánu

Situaci s Product Owners je nutné systematicky řešit. Do řešení by se mohlo zapojit oddělení pro řízení projektů, které by poskytlo „Proxy Product Owners“. Proxy Product Owneři by byli speciálně vyškolení lidé, kteří by byli delegováni oddělelním pro řízení projektů. Na jednotlivých produktech nebo projektech by měli za úkol prioritizaci úkolů v Product Backlogu a pomáhali by tak řídit projekt. Zatím tato možnost není, ale byl by to jeden ze způsobů, jak se vypořádat s nedostatkem Product Ownerů ve 2. nebo 3. fázi akčního plánu. Další možností je, zapojit byznys jednotky, udělat např. kampaň nebo navýšit FTE pro oddělení, která by se na vytváření nových produktů chtěla účastnit, a vytvořit jim větší prostor, aby v pracovní době dělali Product Owners. Současná situace vypadá tak, že PO mají svoji práci jako vedlejší činnost ke své hlavní náplni v byznysových jednotkách. Vytvoření prostoru pro řízení celého projektu agilně napomůže zvýšení zájmu o tyto pozice, jelikož management bude chápat zvýšené potřeby na alokace byznysových lidí pro účely plnění role PO a navíc tato role přináší určité pravomoci v rozhodování o svěřeném systému, které by mohly motivovat zaměstnance byznysových jednotek, aby se sami přihlásili, že mají o roli Product Ownera zájem.

2.4.5 Podpora oddělení pro řízení projektů k vytvoření agilní části metodiky

2.4.5.1 Nová projektová metodika pro mód 2

Oddělení, které se stará o vytváření a aktualizaci projektové metodiky musí vytvořit prostor pro vznik její nové větve. Tato část projektové metodiky bude umožňovat řídit projekty plně agilně. Daná úprava nebude jednoduchá a bude se na ní muset shodnout více částí banky. V klasickém vodopádovém řízení je nastaveno mnoho prvků kontroly, hlavně pak mezi jednotlivými částmi projektu je vždy sestavena komise, která má pravomoc okamžitě projekt ukončit, pokud není spokojená s výstupy předchozí fáze projektu. Zavedení agilní části metodiky nesmí být vnímáno jako snaha tyto kontroly obejít a provést neuvážená a neschválená rozhodnutí. Jinými slovy nesmí se dělat neschválené kroky a zavádět chaos do fungujícího systému. S tím souvisí uklidnění projektových manažerů, že jejich role zůstane neměnná a po zavedení agilního mód 2 jim nebudou odebrány pravomoci, které doposud měli. Vývojové týmy budou nadále řídit vývoj a projektoví manažeři budou pomáhat v řízení projektu, financí apod. Oddělení pro řízení projektů je nejvyšší autorita a definuje strategický základ pro řízení projektů. Není úkolem oddělení pro řízení projektů definovat, jak má agilita fungovat na taktické a detailní úrovni. Oddělení bude definovat strategickou vrstvu a v taktické vrstvě bude řada na IT oddělení, aby si samo definovalo konkrétní kroky, které budou muset být splněny, aby agilní mód byl rovnocenný s mód 1 a aby byl dostatečně kontrolovatelný.

2.4.6 Zaměstnanci rozumí přínosům Bimodal IT

Zaměstnanci v kroku 1 akčního plánu Při zavádění agilního řízení k implementaci malých mimoprojektových úprav systémů, měl autor možnost zjišťovat pohled spolupracovníků na prováděné změny s cílem zjistit, jak složitě se na ně adaptují a zda by jim něco při přechodu na agilní řízení pomohlo. Velká skupina zaměstnanců se domnívala, že před zavedením agilního řízení měli svoje vlastní způsoby řízení, které fungovaly dobře. Jejich změna za agilní schůzky a agilní artefakty jim ale nedělala větší problémy a časem si zvykli a teď už by se ke starému způsobu řízení nevrátili. Složitější přizpůsobování se přechodu na agilní řízení bylo pro pracovníky, kteří v bance pracují 10 a více let. Pokud byli zvyklí na určité zažité postupy a poté je museli měnit za jiné, mohlo to pro ně být nepříjemné. Na druhou stranu změna vždy přináší nový elán do práce. Další výzva pro agilní řízení je, že IT pracovníci dlouhodobě nevidí reálný přínos své práce. Pracují vzdáleni od byznysové stránky jimi zpravovaných systémů a tudíž nejsou schopni posoudit, kolik financí jejich systém bance ušetří. Mnoho zaměstnanců si myslí, že jejich práce není důležitá a přitom jejich systém ročně šetří bance miliony korun. Výzva pro agilitu je tedy spojit IT a byznys, aby IT zaměstnanci viděli přínosy své práce a tím byli i více motivováni naplňovat své cíle. Z manažerské psychologie totiž

víme, že pochvala má větší účinek na motivaci zaměstnanců než zvýšení platu. Obecně by to mohl být velký přínos pro banku, pokud by byli pracovníci více flexibilní, zavádění změn do prostředí banky by bylo jednodušší, případně by sami zaměstnanci přicházeli s tím, že chtějí něco změnit.

Zaměstnanci ve 2. a 3. kroku akčního plánu V dalších krocích nebude změna nařízena, každý projekt by se dal řídit jak agilně, tak klasicky vodopádově a zaměstnanci by si tak sami mohli vybrat, do kterého projektu půjdou či nepůjdou pracovat, pokud by jim to umožňovalo řízení kapacit. Koncept by mohl fungovat jako spojené nádoby, pokud by některý způsob řízení projektů byl lepší, protože by si ho vybrali jak projektoví manažeři, tak ostatní členové projektových týmů. Okamžitě by se tak ukázalo, kteří pracovníci mají blíže k agilnímu přístupu, plnému změn a prototypů postupně dodávaných před konečným řešením, kdo má blíže k prostředí, které není jasně definované a naplánované. A naopak bylo by zřejmé, kteří pracovníci raději setrvají v původním vodopádovém řešení, jasně definovaném, kde je od začátku jasné, co se bude dělat, a aktivity jsou spíše typu řešení známých dobře probádaných problémů. Počet projektů by se mohl měnit podle toho, který způsob řízení by byl zaměstnanci preferovaný.

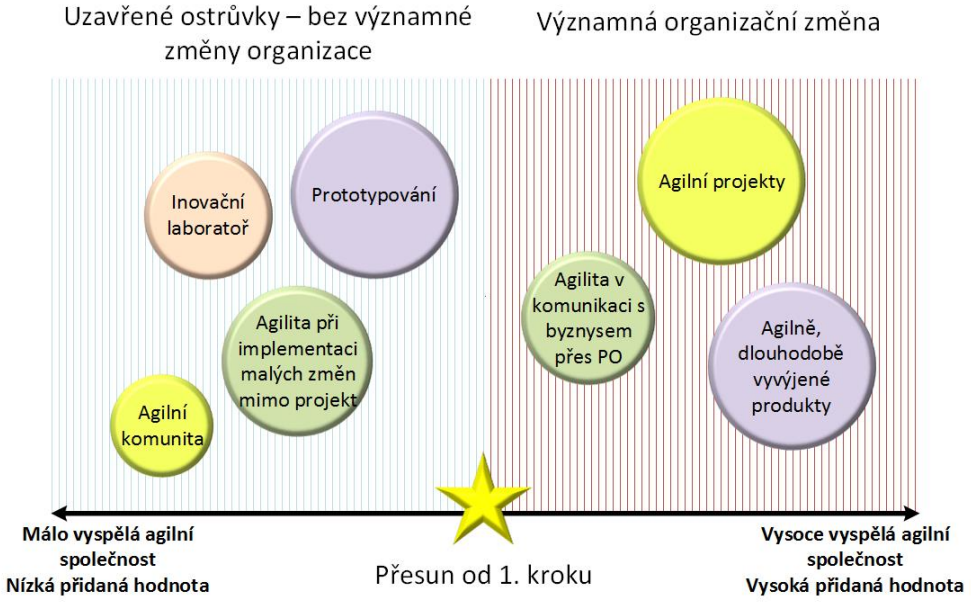
2.5 Akční plán a kroky nutné k zavedení Bimodal IT

Podle kapitoly 1.7 víme, že změny v organizaci vyžadují čas na jejich adopci. Pokud si dáme práci a změnu správně prezentujeme, zvyšujeme tak pravděpodobnost úspěchu a maximalizujeme přínosy. Zkoumáním současné situace v bance, plánováním dalších nutných kroků k realizaci konceptu Bimodal IT a inspirací z 1.6.2.3 autor vytvořil akční plán, jak ve třech krocích dokončit transformaci banky podle Bimodal IT. V prvních fázi jde o malé uzavřené změny, v dalších krocích se staví na úspěchu těchto kroků a aktivity se postupně rozšiřují až na úroveň celé banky.

2.5.1 Krok 1 - Uzavřené ostrůvky

Krok 1 představuje počátek zavedení Bimodal IT do společnosti, ten je v základu jednoduchý, ale pokud bychom ho chtěli ze dne na den zavést do společnost korporátních rozměrů, zjistíme, že cesta k němu nebude tak přímá, jak by se mohlo na první pohled zdát. Podle doporučení od poradenské firmy Gartner 1.5 a po poučení z případové studie z tureckého Vodafone 1.6.2.3 je nejlepší si udělat vlastní důkaz životaschopnosti celého projektu na malých uzavřených částech, kde nemůže nastat problém v komunikaci s okolím. Pro tyto malé ostrůvky není nutné dělat zásadní změny v organizaci, ale na druhou stranu nám nepřinášejí takové množství výhod, jak je tomu u plného využití

Bimodal IT. V prvním kroku je zásadní, aby si každý, kdo s těmito ostrůvky přijde do styku „vyzkoušel“ jak agilita funguje a jaké by mohla mít přínosy při rozšíření do celé banky. Tento proces pomalu probíhá celé 2 roky, které autor práce v bance konzultoval. Bimodal IT je hlavně o zaměstnancích, změně jejich přístupu a přemýšlení nad problémy. Vše musí jít pomalu a oni si musí pomalu zvykat. Dále budou popsány koncepty již aktuálně v bance fungující. Na pravé straně grafu výše jsou změny, které vyžadují zásadní organizační změnu a významné zásahy do organizační struktury nebo oblasti projektové metodiky. Tyto kroky se musí důkladně připravit a naplánovat. Přikročí se k nim, ve chvíli, kdy budou všichni klíčoví zaměstnanci dostatečně adaptovaní na agilní principy a nebude jim přechod činit výraznější problémy.



Obrázek 2.9: Popis změn ve zkoumané bance rozdělený podle [16].

2.5.1.1 Prototypování

V bance dlouhodobě využívaný proces prototypování je moderní způsob přístupu k vyvíjení nového SW produktu. Tento koncept se používá k vyjasnění požadavků na SW produkt. Slovní vyjádření nebo popis vývojovými diagramy nemusí vyznít jednoznačně, proto se přistoupí k implementaci jednoduchého prototypu, který může mít jen základní funkce systému vytvořené pouze pomocí prototypovacího nástroje nebo jednoduchého HTML bez integrace na ostatní systémy a bez reálných dat. Historicky se tento přístup používal k ověření využitelnosti nové technologie v prostředí banky. Jelikož tento proces nebyl nijak řízen, bylo ověření, zda je technologie životaschopná závislé na ně-

kolika málo jedincích, kteří byli nadšeni novými technologiemi a vytvářeli tyto důkazy životaschopnosti mimo pracovní dobu. V rámci Inovační laboratoře popsané níže, se dnes daří vymýšlet nové koncepty a použití nových technologií. Vznik nových nápadů a následně prototypů za použití nových technologií je systematicky podporován.

2.5.1.2 Agilita při implementaci malých změn v systému implementovaném projektem

Takovým malým změnám se v bance říká byznys a technické SE (small enhancement). Na tyto aktivity je vymezen rozpočet pro liniové mimoprojektové řízení. Rozpočet je zatím řádově menší než na projekty (přibližně v poměru 80:20), ale vývojové jednotky si samy mohou určit způsob realizace malých technologických změn. Je to tedy ideální místo pro využití agility. Pro zajištění bližší komunikace probíhají pravidelné schůzky s byznysem, který dané aplikace obsluhuje. Je stanoven Product Owner pro systém určující priority pro zásahy na systému podle byznysových priorit. Vývoj a nasazování těchto změn probíhá ve 2týdenních sprintech a každý měsíc se nasazují nové verze takto vyvíjených SW. Nad některými systémy jsou implementované Nástroje pro kontinuální integraci a nasazení probíhá automaticky. Probíhá úzká spolupráce s konkrétními uživateli systému. Každé 1-3 sprinty probíhá prezentace změn, které se do systému nasazovaly a jejich předvedení konečným uživatelům. Jedna velká část vývoje přešla kompletně na agilní vývoj malých změn - přibližně jedna pětina celého vývoje. Ostatní buď přešly částečně nebo zatím váhají.

2.5.1.3 Agilní komunita

Další z aktivit na podporu informovanosti zaměstnanců je agilní komunita. Schází se jednou za 14 dní, má určeného vedoucího, který ji organizuje a vybírá probíraná témata. Může na ni přijít každý v rámci své pracovní doby. Agilní komunita mimo jiné řeší stav zavádění agility v bance, co by se dalo udělat pro rychlejší přechod směrem k agilnímu řízení, sjednocení názorů na agilitu mezi jednotlivými vývojovými týmy apod. Agilní komunita nemá žádné rozhodovací pravomoci, ale její členové jsou představitelé středního managementu IT napříč bankou. Pokud se na nějakém rozhodnutí shodnou na agilní komunitě, mají dostatečné možnosti uvést ho formálně v platnost. Agilní komunita prozatím definovala:

- Další kroky nutné k zavedení agility a jejich časový plán
- Definici produktu
- Doporučení, jak se mají chovat Scrum Master a Product Owner
- Dále prozkoumává možnosti agilního testování

2.5.1.4 Inovační laboratoř

V bance v současnosti působí dvě Inovační laboratoře:

- IT Inovační laboratoř
- Marketingová inovační laboratoř

Obě tyto laboratoře úzce spolupracují. Marketingová inovační laboratoř sbírá návrhy na nové nápady skrz webový portál. Ty jsou dále vyhodnocovány, a pokud se rozhodne, že některý z nich je zajímavý a téma je z oblasti IT, pak se přistoupí ke komunikaci s IT Inovační laboratoří, která spolupracuje na vytváření prototypu. IT Inovační laboratoř funguje i samostatně a přináší nové nápady mimo banku, transformuje je do podoby, aby byly v bance využitelné a následně iniciuje jejich vytvoření. Mezi úspěšné projekty patří například aplikace pro Apple Watch nebo vytváření interních sociálních sítí na intranetu banky. O těchto technologických novinkách se nebudeme dále bavit, vystačily by na další práci.

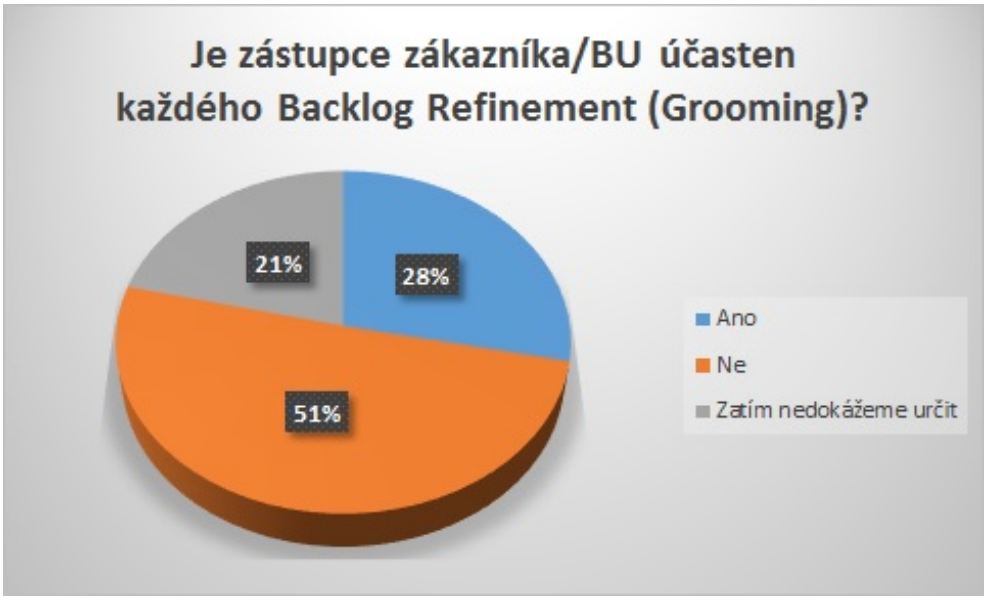
2.5.1.5 Agilní dotazník

Přibližně ve 2/3 prvního kroku zavádění konceptu Bimodal IT byl proveden na oddělení, kde fungovaly Scrum týmy průzkum. Kompletní výsledky průzkumu jsou uvedeny v příloze. D Z výsledků je nejzajímavější hlavně výsledek, který je přiložený k této části. 2.10 Z něho vyplývá, že 51 procent Product Ownerů se nezúčastňuje Groomingu. Toto se dá vnímat jako výzva pro druhý krok. Aktivně zapojit byznys je jeden z klíčových bodů, které se musí zajistit.

Pozitivní rozhodně je, že 51 procent dotázaných odpovědělo, že mají automatizované buildování. Nástroje pro automatický build a nasazování jsou zásadním kladem do dalších fází. Další pozitivní zjištění je, že 93 procent týmů je sestavených pro dlouhodobou podporu produktu. Výzva do budoucna je, že jen 59 procent Prodcut Ownerů je respektováno vlastními kolegy. Do dalších kroků bude potřeba toto číslo udržet a pomalu zvyšovat.

2.5.2 Krok 2 - Zavedení agility pro komunikaci IT s byznysem

Ve druhém kroku si Bimodal IT klade za cíl zajistit, aby všechny části IT intenzivně komunikovaly se svými kolegy na byznysu. Pro každý systém musí být stanovený PO, který bude mít kompetence k tomu, aby prioritizoval požadavky kladené na jeho systém. PO musí fungovat jako jediný bod, ke kterému budou směřovat požadavky týkající se daného systému. PO musí na základě případné konzultace s IT vyhodnotit, zda je změna pro jeho systém optimální a vyřešit jakým způsobem a kdy má být implementována. Tento model mimo jiné snižuje míru chaosu v systému. Nesmí být běžnou praxí, že změny na systémech se realizují bez informování jeho byznysového vlastníka. Ve druhém



Obrázek 2.10: Zapojení byznysu do groomingu

kroku je nutná podpora Oddělení pro řízení projektů, které musí iniciovat změnu na všech byznysových jednotkách. Tyto jednotky budou potřebovat podporu i od svého managementu, jelikož při zavádění konceptu budou kladeny vyšší nároky na časové možnosti byznysových zaměstnanců. To by mohlo způsobit nároky na zvýšení počtu zaměstnanců. Pokud si byznysové jednotky zvyknou na každodenní komunikaci s IT prostřednictvím Product Ownera, bude to další krok k adopci konceptu Bimodal IT a jen krok k tomu, aby se celé projekty mohly řídit agilně.

2.5.3 Krok 3 - Rozšíření agility do celé banky

Poslední třetí krok předpokládaný v horizontu jednotek let bude navazovat na druhý krok. Ve druhém kroku se byznys naučí komunikovat s IT. Konceptu nebude stát v cestě žádná překážka za vedením plně agilních projektů. Zde bude opět hrát velkou roli oddělení pro řízení projektů. Budou potřeba zásadnější zásahy do strategické vrstvy řízení projektů. Pro menší inovativnější projekty bude možné využít agilního módu a sestavit agilní projektový tým s projektovým manažerem a v úzké spolupráci s byznysem vést projekt s nejmenšími možnými náklady. Podle systému, který je nejvíce zasažen se vybere hlavní Product Owner projektu a ten se může v případě, že nebude schopen řešit situaci v projektu poradit s Product Owners ostatních systémů. Tento krok přinese organizaci nejvíce přínosů a koncept Bimodal IT bude plně využit.

2.5.3.1 Škálování agility

V třetím kroku budou hrát důležitou roli frameworky používané ke škálování agility. Častými argumenty, proč neřídit celý projekt agilně je, že agilita nemá stanovená pravidla, jak řídit více agilních týmů, jak k agilním projektům přistupovat v řízení programů a celého portfolia, jak dlouhodobě plánovat, když přesně nevíme co dodáme, jak se vyrovnat s neurčitostí. Na všechny tyto otázky se snaží odpovědět frameworky na škálování agility. Nejdále je framework Scalled agile framework, který popisuje jak pomocí principů Lean řídit projektové portfolio. Jednodušeji k problému přistupuje Large scalled Scrum, který řeší koordinaci více týmů najednou. Oba tyto frameworky budou pro poslední bod ideální možností, jak řídit větší agilní projekty, které ale řeší neznámý problém.

2.5.3.2 Schvalování pokračování projektu řízeného agilním módem

Pro zvýšení důvěryhodnosti mód 2 bude nutné pečlivě sledovat kroky, které projekt provádí. Po celou dobu se budou používat nástroje na kontrolu odvedené práce (Jira) a vykazování odpracovaných hodin (interní systém pro vykazování stráveného času v práci). Co je ale nejdůležitější, budou se muset opět sestavovat komise,ale nikoliv po jednotlivých fázích projektu, ale iterativně. Můžeme si to představit následovně. Každý měsíc se sejde komise, agilní tým jí předvede svoje výstupy za poslední měsíc, odůvodní, kterým směrem se ve vývoji vydali a proč. Komise pak rozhodne buď o přidělení rozpočtu na další měsíc nebo o zrušení projektu a spokojení se s výstupy, tak jak byly prezentovány. Samozřejmě zde bude prostor pro podmínečné propuštění do dalších sprintů, a pokud nebudou splněny podmínky z minulé schůze komise, bude projekt uzavřen.

Závěr

Cílem práce bylo prozkoumat koncept Bimodal IT definující způsob využití agilního i vodopádového přístupu v jedné organizaci. Cílem práce bylo nejen prozkoumat koncept teoreticky, ale po konzultacích ve vybrané společnosti navrhnout konkrétní časový plán, jak koncept zavést s definicí cílového stavu a jeho přínosů a rizik.

Jelikož se jedná o velkou organizační změnu v korporátním prostředí, bylo od začátku jasné, že se zavedení konceptu Bimodal IT nepodaří přes noc. Bylo tedy nutné dobře si prostudovat využitelné metodiky jak vodopádové tak agilní a dostupné údaje od společnosti Gartner o Bimodal IT. Následně projít případové studie a inspirovat se v nich při zavádění konceptu do banky. Po konzultacích a v návaznosti na zjištěnou situaci v bance a po spojení se zjištěnými informacemi z teoretické části byl sestaven popis koncového stavu s odděleným módem 1 a 2. Byl stanoven postup, který pro každý projekt stanovuje, jestli je vhodnější ho řídit agilně nebo vodopádově

Průběžně autor podporoval všechny uzavřené ostrůvky definované v prvním kroku. Konkrétně se autor podílel na zavádění PoC pro nástroje pro kontinuální integraci, zúčastnil se všech agilních komunit, pracoval v Inovačním labu a konzultoval koncept s garantem projektové metodiky na taktické úrovni na IT.

Zároveň po konzultacích na oddělení pro řízení projektů a na schůzích agilní komunity byl stanoven plán na další roky. Počítá se v něm v prvním kroku rozšíření agility na úroveň komunikace byznysu a IT. V posledním kroku se výhledově počítá s plně agilně řízenými projekty. Druhý krok je plánován na příští rok a třetí krok je plánován v horizontu 2-4 let.

Výzvy, kterým se v dalších krocích bude čelit, prostupují všemi vrstvami vedení projektů. Ve strategické vrstvě bude potřeba intenzivně komunikovat

s oddělením pro řízení projektů a vysvětlovat jaký je plán a jak ho sloučit se strategií banky.

Následně budou zavádění agility čekat výzvy v taktické vrstvě, která ze strategické vychází. Až bude možné využít agilní způsob řízení projektů, bude nutné přizpůsobit projektovou metodiku a přístup k řízení projektu iterativnímu přístupu s jeho specifiky.

Nakonec v Detailní vrstvě jsou snahy dlouhodobě používat artefakty agilního řízení jako Scrum Board nebo Product Backlog. Týmy často dodávají práci iterativně a snaží se rychle reagovat na změny. Směrem od technických vedoucích týmů je dlouhodobý tlak lépe plánovat aktivity na projektu a po krátkých iteracích se scházet a inkrementálně vytvářet jak zadání projektu, tak implementaci. Na detailní vrstvě se pak využijí předchozí zkušenosti a celý koncept se uvede v život.

Literatura

[1] Rubin, K. S.: *Essential Scrum*. Upper Saddle River, NJ: Addison-Wesley, první vydání, c2012, ISBN 01-370-4329-5.

[2] Stellman, A.; Greene, J.: *Learning Agile*. Beijing: O'Reilly, first edition. vydání, [2014], ISBN 978-144-9363-826.

[3] Scalled Agile Framework. 2016. Dostupné z: `http://www.scaledagileframework.com/`

[4] Large Scalled Scrum. 2014. Dostupné z: `www.less.works`

[5] RUP – Rational Unified Process. 2017. Dostupné z: `http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/rup/`

[6] Scaling Agile @ Spotify. 10/2012. Dostupné z: `https://ucvox.files.wordpress.com/2012/11/113617905-scaling-agile-spotify-11.pdf`

[7] Smith, G.; Sidky, A.: *Becoming agile*. Greenwich, CT: Manning, první vydání, c2009, ISBN 19-339-8825-8.

[8] Jeffries, R.: *The nature of software development*. Dallas, Texas: The Pragmatic Bookshelf, první vydání, [2015], ISBN 19-412-2237-4.

[9] Gradle. 2016. Dostupné z: `www.gradle.org`

[10] Jenkins. 2016. Dostupné z: `www.jenkins.io`

[11] GIT hub. 2016. Dostupné z: `https://git-for-windows.github.io/`

[12] Urban Code. 2016. Dostupné z: `https://www.ibm.com/developerworks/community/blogs/nfrsblog/entry/top_urban_code_technotes_in_2014?lang=en`

[13] Sonarqube. 2016. Dostupné z: https://www.sonarqube.org/

[14] Weeks, D.: Nexus Reaches 50,000. 2015. Dostupné z: http://www.sonatype.org/nexus/2015/02/27/nexus-reaches-50000/

[15] Sher, P.: JetBrains is Hiring for TeamCity – Join Our Team. 2015. Dostupné z: https://blog.jetbrains.com/blog/2015/04/22/jetbrains-is-hiring-for-teamcity-join-our-team/

[16] Mingay, S.; Mesaglio, M.: Bimodal IT. 2014. Dostupné z: https://www.gartner.com/doc/2798217/bimodal-it-digitally-agile-making

[17] Krejčí, P.: IT jako součást byznysu v modelu Management of Business Informatics. 2014.

[18] Agilní manifest. 2001. Dostupné z: www.agilemanifesto.org

[19] Kunce, E.: *Agilní metody řízení projektů*. Brno: Computer Press, první vydání, 2014, ISBN 978-80-251-4194-6.

[20] Agile Product Ownership in a nutshell. 2012. Dostupné z: http://blog.crisp.se/2012/10/25/henrikkniberg/agile-product-ownership-in-a-nutshell

[21] Poppendieck, M.; Poppendieck, T.: *Lean software development*. Boston: Addison-Wesley, 10 vydání, 2006, ISBN 03-211-5078-3.

[22] Hammarberg, M.; Sunden, J.: *Kanban in action*. Shelter Island NY: Manning, první vydání, 2014, ISBN 16-172-9105-6.

[23] Beck, K.: *Extreme programming explained*. Reading: Addison-Wesley, 2000, ISBN 02-016-1641-6.

[24] Stephens, R.: *Beginning software engineering*. Indianapolis, IN: John Wiley and Sons, první vydání, 2015, ISBN 11-189-6914-6.

[25] Gibbs, R. D.: *Project management with the IBM Rational Unified Process*. Upper Saddle River, NJ: IBM Press/Pearson plc, první vydání, c2007, ISBN 03-213-3639-9.

[26] Delighting Vodafone Turkey's Customers via Agile Transformation. 2014. Dostupné z: http://www.scrumcasestudies.com/wp-content/uploads/2014/10/AgileTransformationInVodafoneTurkey.pdf

[27] Banfield, R.; Lombardo, C. T.; Wax, T.: *Design sprint*. Sebastopol, CA: O'Reilly Media, Inc., druhé vydání, [2016], ISBN 14-919-2317-2.

[28] Newton, P.: *Managing project risks*. USA: Free-management-ebooks.com, první vydání, 2015, ISBN 978-1-62620-986-4.

Příloha **A**

Seznam použitých zkratek

BU Bussiness Unit - neIT část banky

CIO z angl. Chief Informatics Officer - ředitel informatiky

CRM Customer Relationship management - sw nástroje a proces pro řízení svztahu se zákazníkem

FTE TBD

HR - human resources - oddělení zabývající se náborem, pracovními smlouvami, pečování o zaměstnance, odměňováním a benefity

HTML HyperText Markup Language

MD Man Day - práce jednoho zaměstnance 8 hodin

PoC z angl. Proof of concept - důkaz, že nějaký projekt je v realizovatelný

PO Product Owner - role v agilním řízení

RUP Rational Unified Process - vodopádová metodika založená na UP

UML Unified Modeling Language - souhrn diagramů používaných ke kompletní analýze softwaru

UP Unified Process - vodopádová metodika

XP Extreme programming - agilní metodika

Příloha B

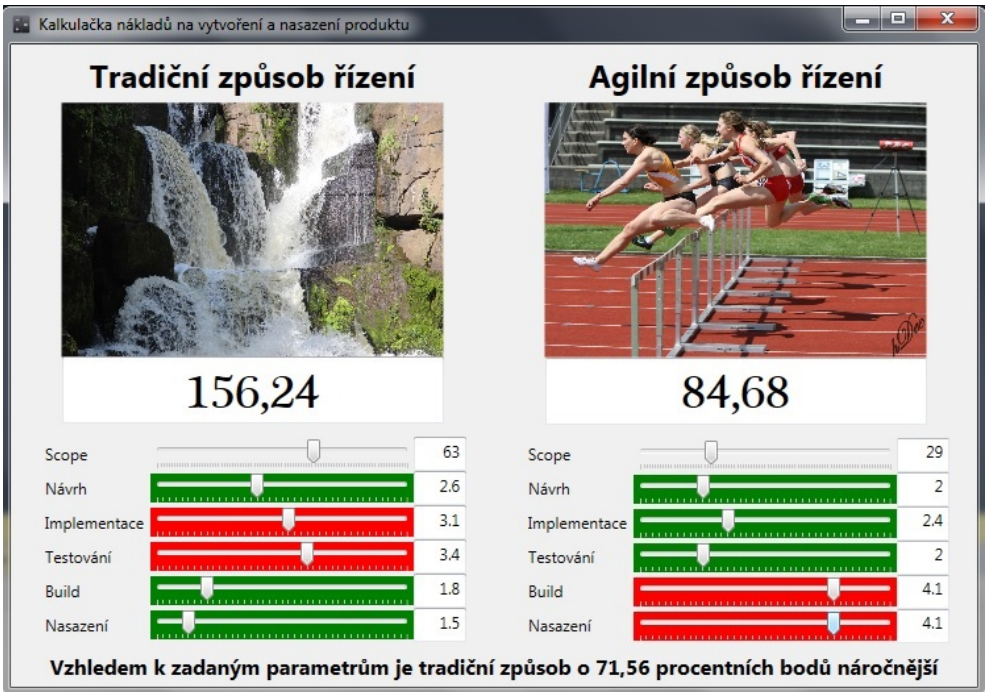
Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha C

Popis programu manažerské kalkulačky pro výpočet nákladů na projekt

Program je ve 3. verzi. Jsou zde všechny části implementace podle Životního cyklu vývoje software. Jednotlivé části mají shodnou složitost. Pokud na posuvnících nastavíme úroveň složitosti jednotlivých částí, program spočítá, který způsob je nákladnější a o kolik procentních bodů. Jeho vývoj byl zastaven po 3. sprintu. Pokud by se dále rozšiřoval, bylo by nutné parametry blíže přiblížit realitě, zadat kolik by jednotlivé úkony stály a kolikrát by se museli provádět plus další činnosti v projektu. Pak by výpočet byl blíže realitě. Současný vzhled kalkulačky:



Obrázek C.1: Aktuální vzhled manažerské kalkulačky na výpočet nákladů na projekt

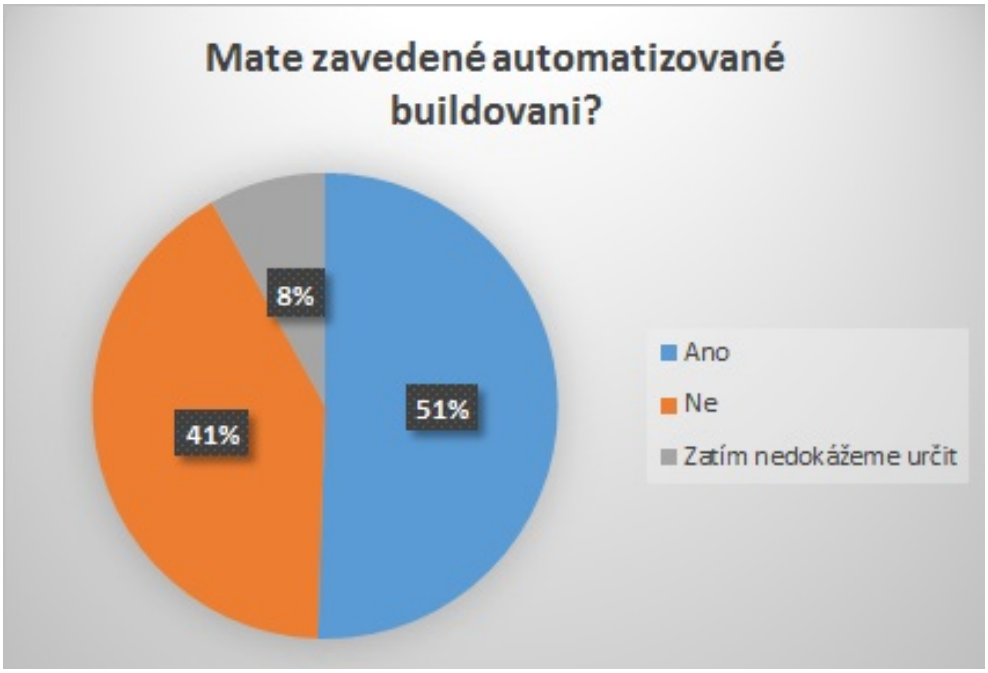
Příloha D

Dotazník o stavu agility

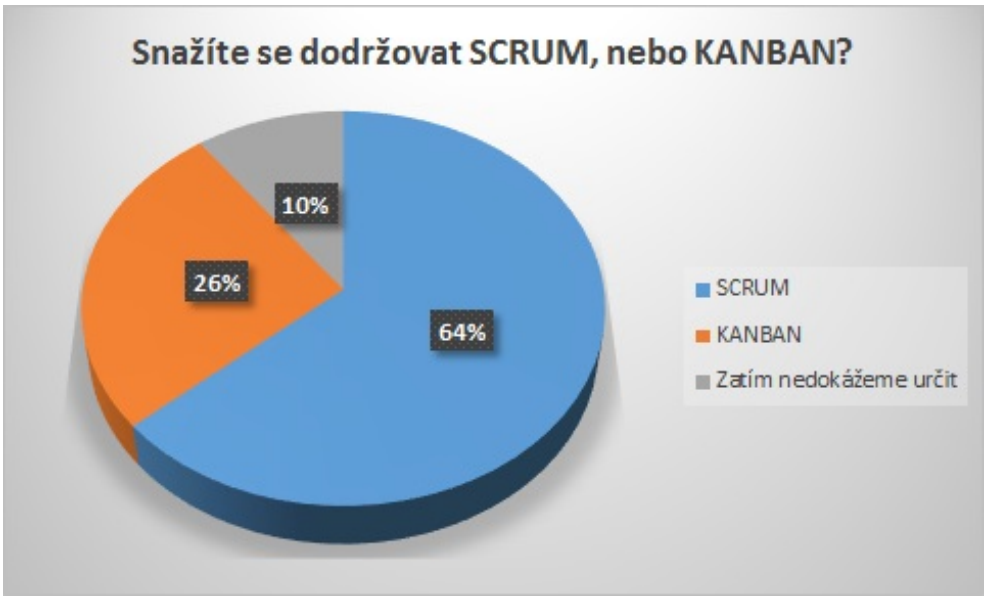
Pro představu přikládám výsledky dotazníku, který byl dělán na oddělení, kde se zavádělo agilní řízení přibližně ve 2/3 prvního kroku.

Obrázek D.1: Agilní dotazník: Otázka 1

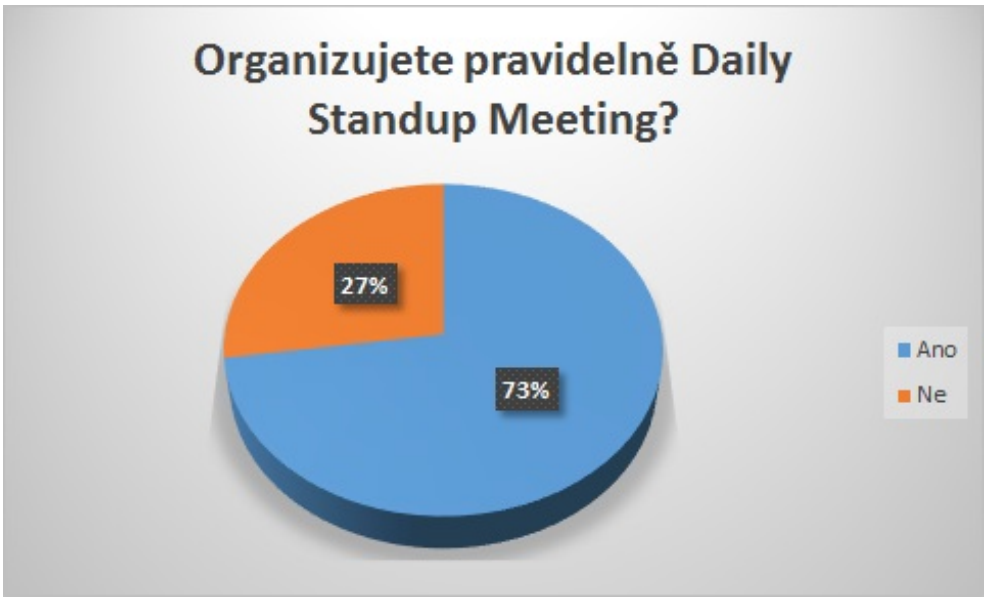
Obrázek D.2: Agilní dotazník: Otázka 2



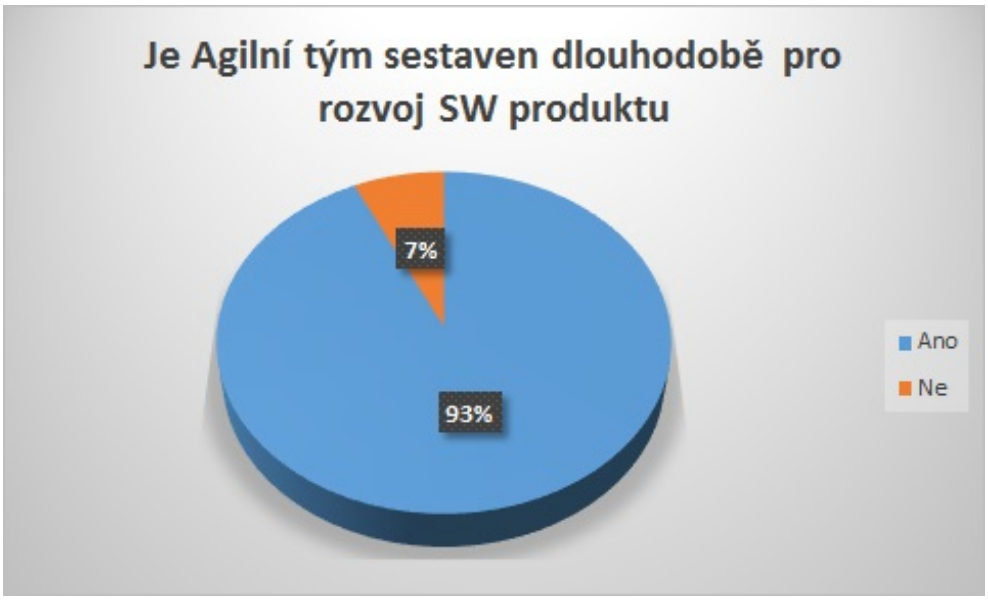
Obrázek D.3: Agilní dotazník: Otázka 3



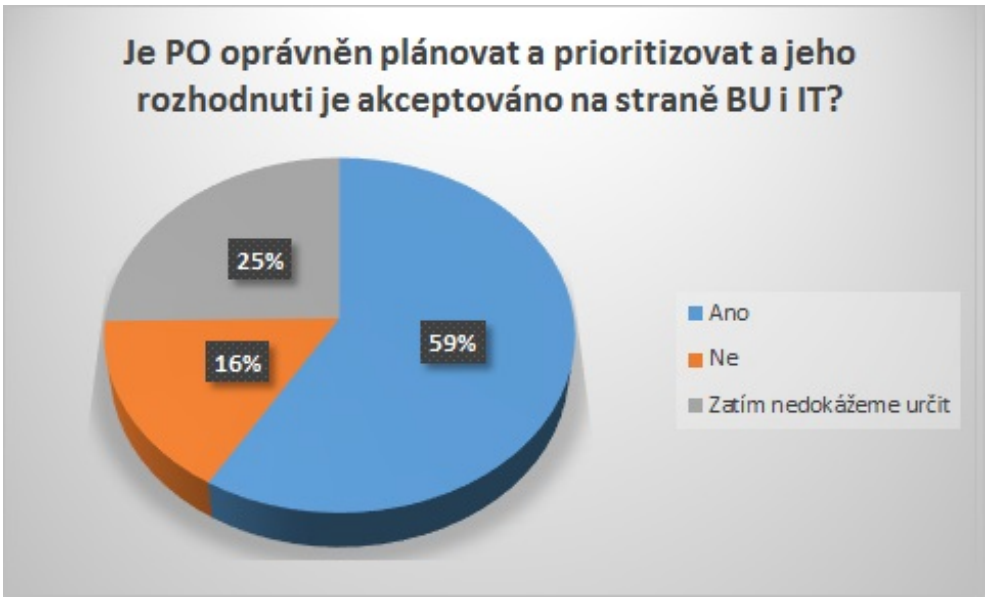
Obrázek D.4: Agilní dotazník: Otázka 4



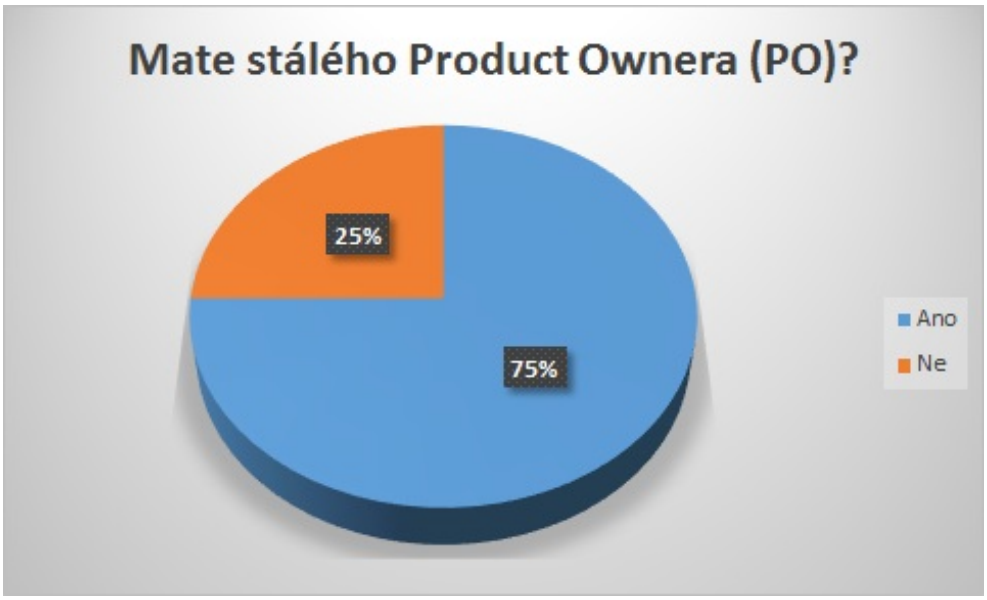
Obrázek D.5: Agilní dotazník: Otázka 5



Obrázek D.6: Agilní dotazník: Otázka 6



Obrázek D.7: Agilní dotazník: Otázka 7



Obrázek D.8: Agilní dotazník: Otázka 8

Obrázek D.9: Agilní dotazník: Otázka 9