VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

APLIKACE AGILNÍ METODIKY V PROJEKTOVÉM ŘÍZENÍ
APPLICATION OF AGILE METHODOLOGY IN PROJECT MANAGEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Patrik Szuchan

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Lenka Smolíková, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Patrik Szuchan**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Lenka Smolíková, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Aplikace agilní metodiky v projektovém řízení

Charakteristika problematiky úkolu:

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

Cíle, kterých má být dosaženo:

Cílem diplomové práce je návrh změn pro zefektivnění projektového řízení, vývoje a práce ve vývojových týmech zavedením agilní metodiky ve firmě.

Základní literární prameny:

BUCHALCEVOVÁ, A. Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky. Praha: Grada, 2005. ISBN 80-247-1075-7.

CARROLL, J. Effective project management. Leamington Spa, Warwickshire, UK: In Easy Steps Ltd, 2012. ISBN 978-1840784466.

KADLEC, V. Agilní programování: metodiky efektivního vývoje softwaru. Brno: Computer Press, 2004. ISBN 80-251-0342-0.

SOMMERVILLE, I. Softwarové inženýrství. Brno: Computer Press, 2013. ISBN 978-80-251-3826-7.

ŠOCHOVÁ, Z. a E. KUNCE. Agilní metody řízení projektů. Brno: Computer Press, 2014. ISBN 978-8--251-4194-6.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně dne 29.2.2020

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

Abstrakt

Diplomová práca pojednáva o zmenách pre zefektívnenie projektového riadenia, vývoja a práce vo vývojových tímoch zavedením agilnej metodiky Scrum. Začiatok diplomovej práce sa venuje teoretickým východiskám a základným pojmom, ktoré sú použité v ďalších častiach práce. Analytická časť rozoberá súčasný stav softvérovej divízie. V návrhovej časti je na základe výsledku analýzy vytvorený návrh zmien v projektovom riadení použitím agilnej metodiky, ktorej cieľom je zefektívnenie projektového riadenia a vývoja softvéru. Záverom návrhovej časti sú spracované ekonomické zhodnotenia návrhu a jeho prínosy pre divíziu.

Abstract

The diploma thesis deals with changes for streamlining project management, software development and work in development teams by introducing the agile Scrum methodology. The beginning of the diploma thesis deals with the theoretical background and basic concepts that are used in other parts of the work. The analytical part analyses the current state of the software division. In the design part, based on the results of the analysis, a proposal for changes in project management is created using the agile methodology, which aims to streamline project management and software development. At the end of the design part, economic evaluations of the design and its benefits for the division are processed.

Kľúčové slová

Scrum, šprint, agilná metodika, vodopádový model, projektové riadenie

Key words

Scrum, sprint, agile methodology, waterfall model, project management

Bibliografická citácia

SZUCHAN, Patrik. *Aplikace agilní metodiky v projektovém řízení.* Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/127542. Diplomová práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Lenka Smolíková.

Čestné prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pani Ing. Lenky Smolíkovej, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Brne dňa 16. mája 2020

.....

podpis študenta

Poďakovanie

Na tomto mieste by som sa chcel poďakovať svojej vedúcej Ing. Lenke Smolíkovej, Ph.D. za pomoc a vedenie pri vypracovávaní diplomovej práci. Zároveň patrí vďaka aj mojej priateľke, ktorá ma počas práce podporovala.

OBSAH

ÚVOD

Svet sa každým rokom zrýchľuje, pričom tento trend bude stále napredovať. Ľudstvo produkuje stále väčšie a väčšie množstvo dát, ktoré sa niekam ukladajú a ďalej spracovávajú. Tieto dáta pochádzajú a sú ďalej využívané hlavne z mobilných zariadení pomocou prístupu na internet. Formu zobrazenia obsahu si môže zvoliť každý sám, väčšina informácií je dostupná prostredníctvom aplikácie alebo webovej stránky. Trend nových informácií sa veľmi rýchlo mení. To, čo bolo jeden deň populárne, môže byť o mesiac zabudnuté a nikto si na to ani nespomenie. Kvalita obsahu je dôležitá, ale rýchle prispôsobenie sa trendu alebo zmene je rovnako dôležité, ak nie viac.

Rýchlym vývojom informačných technológií vzniklo veľa spoločností vytvárajúcich software pre vlastné použitie alebo ako službu. Čím ďalej viac ľudí a firiem hľadá spôsob, ako byť relevantný v dnešnom online svete, a preto vyhľadáva služby a softvérové riešenia ponúkané týmito spoločnosťami. Tie v rámci snahy ostať konkurencie schopné používajú metódy pre zvýšenie schopnosti reakcii na zmenu. Rýchlosť reakcie na zmenu je ovplyvnená aj metódami projektového riadenia, ktoré môžeme rozdeliť na dve základné skupiny a to na tradičné metódy a agilné metódy.

Veľké množstvo firiem využíva tradičné metódy. Dôvody sú rôzne: ich zavedenie do spoločnosti a jej procesov je jednoduchšie alebo spoločnosť ešte nepocítila potrebu pre použitie inej metódy. Tradičné metódy majú aj svoje nevýhody, ktoré sú spoločnosťami akceptovateľné napriek spôsobujúcim problémom do bodu, kým neovplyvňujú jej chod. Novšie spoločnosti, prevažne zaoberajúce sa softvérovým vývojom používajú agilné metódy už od svojho založenia alebo v priebehu ich rastu z malej spoločnosti na strednú spoločnosť. Prispôsobujú sa tak potrebám rýchleho reagovania na dianie v spoločnosti.

Agilné metódy môžu ponúknuť riešenie na niektoré z nevýhod, avšak ich zavedenie do organizácie je zložitejšie. Napriek tomu softwarové spoločnosti upúšťajú v projektovom manažmente od tradičných metodík v prospech agilných alebo ich kombinácii. Agilné metodiky dovoľujú väčšiu flexibilitu pri plánovaní a zároveň vyššiu rýchlosť dodávania hotových častí produktu na trh. Umožňujú lepšie reagovať

na zmeny počas priebehu projektu, pričom podporujú zapojenie zákazníka do vývoja projektu. Proces vývoja prebieha v menších tímoch, v ktorých sa zakladá na vzájomnej spolupráci a komunikácii.

Táto diplomová práca sa zaoberá opisom spoločnosti a jej softvérovej divízii, ktorá využíva tradičnú metodiku, nazývanú Vodopádový model. Následne analýzou procesu riadenia projektu a návrhom zavedenia novej agilnej metodiky Scrum.

CIELE PRÁCE, METÓDY A POSTUPY SPRACOVANIA

Cílem diplomové práce je návrh změn pro zefektivnění projektového řízení, vývoje a práce ve vývojových týmech zavedením agilní metodiky ve firmě.

Úvodná časť práce sa zaoberá základnými pojmami a teóriou, ktorá je následne použitá v analytickej a návrhovej časti práce. Je popísaný životný cyklus projektu, projektový manažment a klasické metodiky vývoja IT projektov. Následne sú opísané agilné metódy s väčším dôrazom na Scrum.

Analytická časť práce sa venuje predstaveniu spoločnosti a jej divízie určenej na vývoj softvérových produktov. Opisuje jej štruktúru, históriu a spôsob riadenia projektov klasickou metodikou vývoja IT projektov. Proces riadenia projektu je analyzovaný a výsledky sú zhrnuté v závere tejto časti.

V návrhovej časti práce sú detailne rozpísané navrhované zmeny vedenia projektu zavedením agilnej metodiky Scrum. Tieto zmeny zároveň adresujú problémové oblasti zistené v predchádzajúcej časti. Sú definované nové role, na základe ktorých je zložený Scrum tím. Následne je navrhovaný priebeh šprintov a meetingov s použitím softvérových nástrojov. Záverom je spracované ekonomické zhodnotenie návrhu a prínosy jeho zavedenia. V práci sú využité metódy agilného riadenia.

1 TEORETICKÉ VÝCHODISKÁ PRÁCE

Prvá časť diplomovej práce bude venovaná rozboru a definícií jednotlivých pojmov, ktoré budú použité v oboch častiach- teoretickej aj praktickej časti diplomovej práce.

1.1 Projekt

Projekt môžeme definovať ako sériu unikátnych, komplexných a prepojených aktivít, ktoré majú cieľ. Ten musí byť dosiahnutý v zadanom čase, v rámci stanoveného rozpočtu a podľa danej špecifikácie. Pojem aktivita predstavuje prácu, ktorú niekto vykoná, pričom sú na sebe závislé. Zadanie časového ohraničenia, rozpočtu a špecifikáciu zadá manažment alebo klient. Projekt môže mať rôzny rozsah a obsah, príkladom môže byť malý školský projekt na jeden semester, niekoľko mesačné až ročné projekty v rámci zákazkového softwarového vývoja alebo viac ročné projekty ako napríklad výstavby obchodných domov [1]. Základné charakteristiky projektu:

- Každý projekt sa skladá z troch fáz: predprojektová, projektová a poprojektová
- Projekt má priradenú osobu (projektový manažér), ktorá projekt vedie k dopredu stanovenému cieľu
- Má celkový cieľ a jedinečné výstupy
- Obsahuje jednotlivé úlohy, ktoré sa majú vykonať medzi konkrétnym začiatočným a koncovým bodom
- Načasovanie vzťahov medzi jednotlivými úlohami
- Zdroje na vykonanie práce
- Rozpočet

1.1.1 Životný cyklus projektu

Životný cyklus projektu sa zvyčajne delí na tri časti: predprojektová, projektová a poprojektová. Pričom najdlhšia a najnáročnejšia z nich je projektová časť. Tá sa ďalej delí na menšie časti: zahájenie, prípravu a ukončenie projektu.

Predprojektová fáza

Účel predprojektovej fázy je preskúmať príležitosť pre projekt a posúdiť realizáciu daného zámeru. To má za následok vypracovávanie rôznych analýz a štúdií, pričom

ich výsledkom sú dva hlavné typy dokumentov: štúdia príležitostí a štúdia uskutočniteľnosti [2].

Štúdia príležitostí berie do úvahy aktuálnu situáciu organizácii, situáciu na trhu a ich predpokladaný vývoj. Výsledok štúdie môže príležitosť potvrdiť alebo nie a na základe toho odporučí daný projekt realizovať. Spoločnosť nakoniec posúdi, či bude projekt naozaj realizovaný alebo nie [2].

V prípade záujmu o realizáciu sa vykoná štúdia uskutočniteľnosti. Jej výsledkom by malo byť najvhodnejšie riešenie upresňujúce obsah projektu, plánovaný termín zahájenia a ukončenia projektu, odhadované náklady a potrebné zdroje. Tieto dokumenty samy o sebe nerozhodujú o tom, či sa projekt spustí alebo nie, len dodávajú potrebné informácie kompetentným osobám. Tie rozhodnú, akú cestu zvoliť a či má zmysel projekt realizovať [2].

Projektová fáza

Po schválení projektu manažmentom sa projekt oficiálne spustí. Informácie z predprojektovej fázy sa overia a doplnia, definuje sa cieľ. Rozhodnutie o spustení projektu je reprezentované schválením identifikačnej listiny projektu. Tá obvykle obsahuje definíciu cieľa, vymedzuje hranice projektu - vo financiách, čase, predpokladaných zdrojoch a menuje manažéra projektu [2].

Poprojektová fáza

Poprojektová fáza projektu je obdobie po jeho ukončení. Výsledok projektu, napríklad produkt je v bežnej prevádzke. V tejto fáze prebiehajú garancie a záruky i keď projekt už skončil. Pre projekt je dobré urobiť spätné vyhodnotenie, či priniesol očakávané prínosy alebo naopak nebol prínosný. Je potrebné sa zamyslieť nad získanými poznatkami a skúsenosťami a zároveň sa poučiť z chýb, ktoré sa vyskytli počas projektu [2].

1.2 Projektový manažment

Spoločnosti a organizácie majú neustálu potrebu nových produktov a služieb na predaj alebo pre vlastné použitie. Tieto produkty a služby môžu byť nové alebo môže ísť o ich vynovenú verziu. V oboch prípadoch ich vytvorenie vyžaduje založenie projektu. Každý nový projekt potrebuje projektový manažment. Projektový manažment

umožňuje vytvárať dané produkty a služby podľa harmonogramu a s kontrolou nad procesom. Umožňuje sledovanie a aplikovanie zdrojov potrebných na vytvorenie nového produktu alebo služby [3].

Projektový manažér doručí produkt alebo službu tým, ktorí zaplatili za jeho vývoj, čo môžu byť sponzori, stakeholderi, klienti alebo vlastníci. Ku každému projektu sa vo väčšine prípadov pristupuje rovnakým spôsobom. Je definovaná postupnosť krokov, ktorými sa musí prejsť na základe odpovedí na nasledujúce problémy:

- Určenie, aký nový produkt alebo službu zadávateľ chce a jeho ochotu za to zaplatiť
- Popísanie žiadaného produktu alebo služby, ktorý môžu dostať v akceptovateľnom časovom rozmedzí
- Použiť plánovací proces na vytvorenie špecifikácie produktu alebo služby, naplánovať ho a odhadnúť náklady na jeho realizáciu
- Uplatňovanie kontroly a techník pre kvalitu, risk, náklady a zmeny počas realizácie produktu alebo služby
- Doručenie, odovzdanie produktu alebo služby
- Po doručení projektu posúdiť, čo bolo spravené, čo šlo dobre, mieru spokojnosti klienta s produktom alebo službou a v neposlednom rade, ako môže byť proces zlepšený pre ďalší projekt [3]

1.2.1 Míľniky

Míľnik je ukazovateľ v projekte, ktorý znamená zmenu alebo štádium vývoja. Zobrazuje kľúčové udalosti a mapuje pohyb v pláne projektu. Pomáha zaistiť, aby projekt zostal na správnej ceste. Ak sa míľniky v projekte nenachádzajú, sledujú sa len úlohy, a tým pádom sa nemusí nevyhnutne sledovať správna cesta projektu. Pri väčších projektoch nemusí byť jednoznačne rozoznateľná úloha od míľnika. Príklady míľnikov, ktoré je vhodné sledovať v projektoch:

- Dátumy začiatkov a koncov jednotlivých projektových fáz
- Kľúčové dodania
- Schválenia klientom a stakeholdermi
- Dôležité stretnutia a prezentácie

- Kľúčové dátumy alebo výpadky, ktoré môžu ovplyvniť časový plán [4]

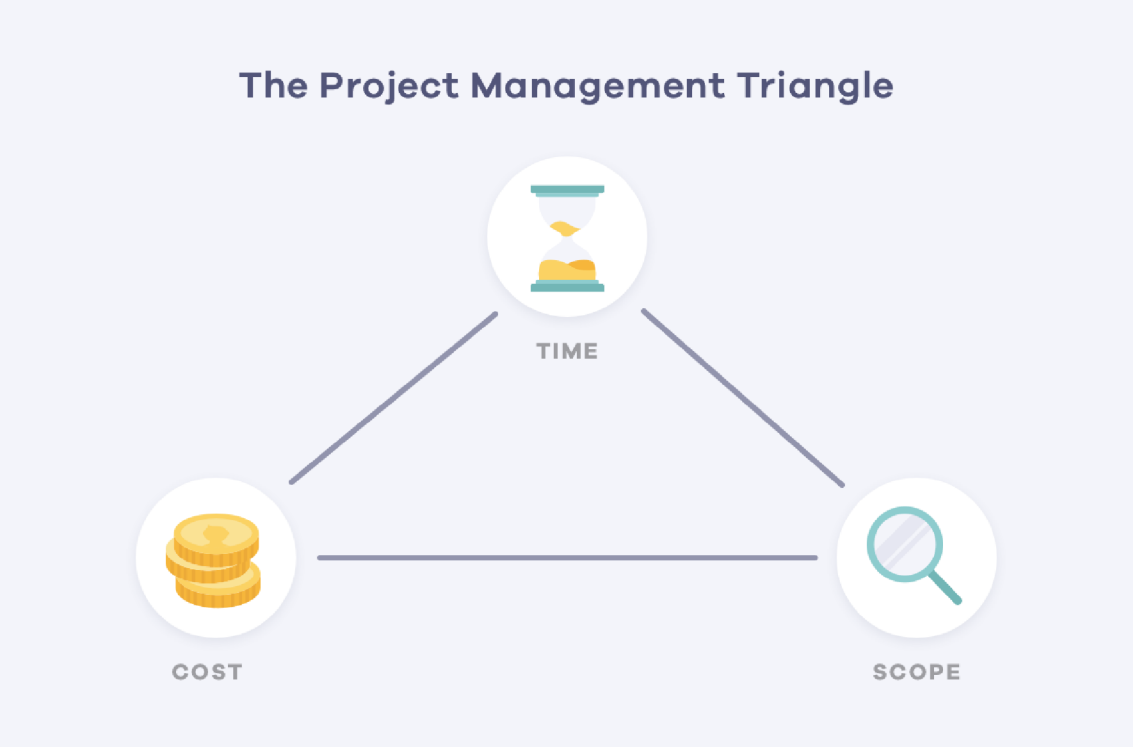
1.2.2 Projektový manažér

Projektový manažér má viacero rolí. Manažuje zdroje potrebné k realizácii projektu, náklady, časový plán a celý proces. Zároveň musí byť schopný vykonávať povinnosti tradičného manažéra, a to: delegovanie, zjednávanie, presviedčanie, organizovanie, koordinovanie, sprostredkovanie a vytváranie tímov. Musí rozumieť dostatočne biznisovému účelu daného projektu, aby vedel správne rozhodovať pri realizácii projektu. Projektový manažér udržiava v rovnováhe viacero uhlov pohľadu, viacero záujmov a rôzne požiadavky, ktoré sú často vo vzájomnom konflikte. Komunikuje so všetkými zainteresovanými stranami a monitoruje náklady, čas, rizikovosť a kvalitu voči definovanému štandardu [5].

S projektovým tímom spolupracuje počas všetkých fáz plánovania a realizácie, aby sa uistil, že sa všetko čo bolo dohodnuté dodržiava a realizuje. Dohliada, či všetci členovia tímu vykonávajú činnosti v súlade s ich zodpovednosťami. Je zodpovedný za všetko týkajúce sa projektu vrátane všetkých problémov projektu, pričom má limitovanú autoritu mimo rozsahu projektu. Vlastníci sa môžu rozhodnúť zmeniť rozsah, prípadne ukončiť projekt [5].

1.3 Projektový trojimperatív

Teória projektového trojimperatívu hovorí o tom, že každý projekt funguje v rámci rozsahu projektu, časového rámca projektu a nákladov. Keď jeden z týchto faktorov zmeníme, vždy ovplyvníme aj ďalšie dva. Úlohou projektového manažéra je vyvážiť tieto tri faktory a riadiť očakávania tak, aby každý rozumel tomu, čo je potrebné pre dosiahnutie úspechu projektu. Často sa hovorí aj o trojuholníku projektového manažmentu, ktorý je znázornený na obrázku č.1 [6].



Obrázok č. 1: Zobrazenie projektového trojimperatívu (prevzaté z [6])

1.3.1 Rozsah projektu

Pridávanie funkcionalít alebo požiadaviek do projektu v jeho priebehu môže prerásť v úplne nový projekt. Preto je dôležité definovať a dokumentovať projektové ciele a požiadavky pred začatím práce. Všetci budú vedieť, ako má vyzerať výsledný stav a zároveň máme zdroj pravdy, ku ktorému sa vieme vrátiť v prípade potreby. Pridávanie funkcionalít natiahne čas projektu a jeho rozpočet. Bude potrebné predĺžiť dobu dodania alebo priradiť viac ľudí na danú prácu, čo zvýši náklady. Monitorovaním zmien rozsahu projektu je možné začať diskusiu ohľadom kompromisov skôr, ako sa projekt dostane z plánovaného smeru [6].

1.3.2 Časový rámec projektu

Väčšina klientov vie, že keď chcú mať projekt dokončený rýchlo, bude ich to niečo stáť. Obzvlášť vtedy, keď nie sú ochotní upustiť z rozsahu projektu. Krátke termíny dokončenia vyžadujú viac zdrojov, aby sa práca stihla zhotoviť načas. Detailná dokumentácia poskytuje dobrý základ pre pochopenie časového obmedzenia, pretože sa dá použiť na odhadnutie projektu. Je dobré prizvať do diskusie aj tím a pozrieť sa nie len na čas strávený na jednotlivých úlohách definovaných v projekte. Je potrebné

zarátať do celkového času aj čas strávený na stretnutiach a čas strávený so stakeholdermi počas projektu. Čím presnejší je odhad, tým lepšie bude prebiehať práca na projekte [6].

1.3.3 Náklady

Odhadnutím času a úsilia potrebného na dokončenie projektu získame prvú predstavu o nákladoch. Pri zostavovaní rozpočtu je potrebné zvážiť aspoň tieto náklady:

- Náklady na zdroje na základe odhadnutých hodín
- Materiál
- Vybavenie potrebné na prácu

O nákladoch je potrebné komunikovať včas a často, aby nedošlo k prekvapeniam pri príchode vysokého účtu za vyhotovenie projektu. Ak sa objaví nečakaný výdaj, treba dôkladne vysvetliť jeho dopady na celkový projekt klientovi a nechať ho rozhodnúť, či je ochotný na výdaj pristúpiť. Klient často nie je odborníkom a nemusí si hneď uvedomiť, koľko ho bude nová funkcionalita stáť [6].

1.4 Klasické metodiky vývoja IT projektov

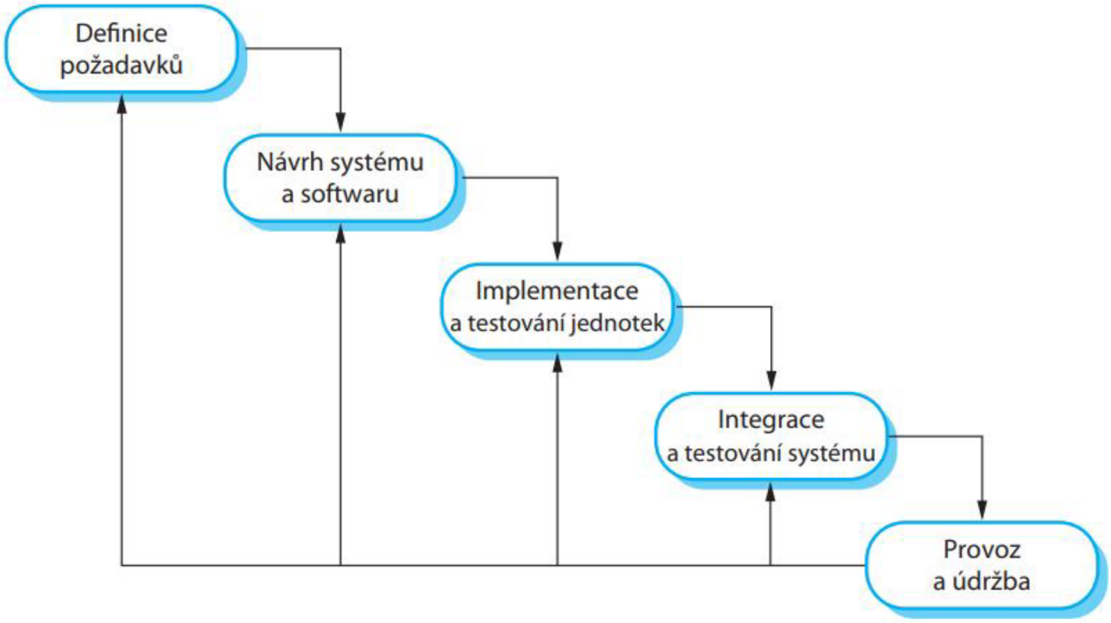
Každý softwarový projekt je unikátny, či už zadaním alebo okolnosťami jeho realizácie. Skladá sa z jednotlivých etáp, ktorých usporiadanie sa môže líšiť na základe zadania projektu. Zároveň majú projekty aj spoločné časti, ktoré sa dajú popísať modelom životného cyklu softvéru - metódou vývoja. Medzi klasické metodiky patria modely opísané v nasledujúcich podkapitolách [7].

1.4.1 Vodopádový model

K jednému z najstarších modelov patrí takzvaný vodopádový model. Jednotlivé etapy sú zoradené za sebou, pričom nasledujúca etapa začína až keď je dokončená etapa pred ňou. Výsledok jednej etapy je zároveň vstupom do ďalšej, pričom sa jednotlivé etapy neprelínajú. Hlavný problém pri použití vodopádového modelu je zákazník, ktorý nie je vždy schopný uviesť všetky požiadavky pred zahájením vývoja. To môže zapríčiniť, že sa neimplementuje všetko, čo by chcel a potreboval. V neskorších fázach projektu, pri predaní softvéru zákazník zistí, že chce ešte niečo pridať, prípadne upraviť. Ak sa rozhodneme dodatočné požiadavky pridať, vraciame sa v životnom cykle na začiatok, namiesto toho, aby sa softvér začal používať. Ako je vidieť na obrázku 2,

pri implementovaní nových požiadaviek sa dostaneme zo štvrtého bodu *Integrácie a testovanie systému* k prvému, *Definícia požiadaviek* [7].

Zavedenie zmien môže byť problematické, pretože sa s nimi nerátalo a už implementovaná funkcionalita na nich nemusí byť pripravená. Štruktúra daného softvéru tak môže začať upadať ešte pred tým, než sa začne používať. V najhoršom prípade sa musí časť starej implementácie zahodiť a vývoj zahodených častí začne znovu [7].



Obrázok č. 2: Schéma vodopádového modelu (prevzaté z [7])

Výhody vodopádového modelu

Vodopádový model môže byť vnímaný ako idealistický model softwarového vývoja. Model ako taký je veľmi jednoduchý, ale napriek tomu je možné ho použiť aj pri reálnom vývoji. Jeho hlavné výhody sú:

- Jednoduchý na pochopenie
- Každá časť modelu je jasne definovaná
- Procesy, vývoj a výsledok sú dobre dokumentované
- Model je dobre použiteľný pre malé projekty, prípadne projekty, ktorých požiadavky sa nebudú meniť behom vývoja [7]

Nevýhody vodopádového modelu

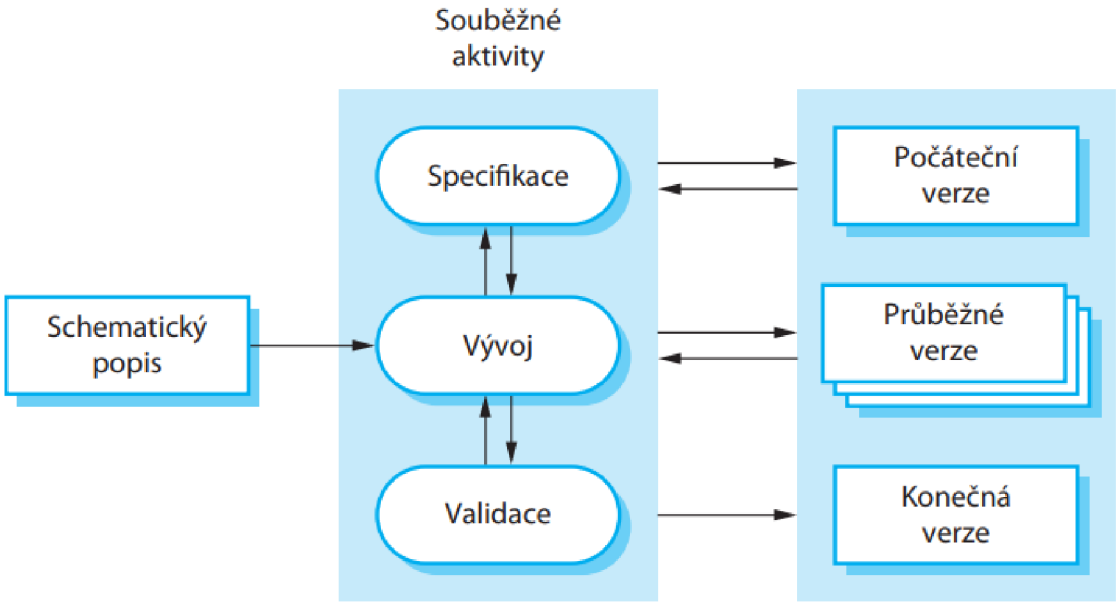
Pri väčších a dlhšie trvajúcich projektoch je veľká pravdepodobnosť zmeny. Zákazník dostáva do rúk hotový produkt, ktorý prešiel všetkými fázami modelu, čo môže byť po značne dlhom čase. Potreby, ktoré mu tento software mal pomôcť naplniť sa medzitým už mohli zmeniť. Zároveň mohol zabudnúť uviesť všetko čo chcel v počiatočnej fázy zadávania požiadaviek. Z toho vyplývajú aj jednotlivé nevýhody vodopádového modelu:

- Žiadna spätná väzba počas jednotlivých fáz projektu - zadávateľ dostane do rúk až hotový produkt.
- Zložité zavádzanie následných požiadaviek na zmenu - predpokladá sa, že zadávateľ vie hneď od začiatku presne čo chce a požiadavky sa nebudú meniť. To sa však vo väčšine prípadov nestáva.
- Fázy sa neprekrývajú - z čoho vyplýva, že ďalšia fáza môže začať, až keď sa ukončí predchádzajúca fáza. Tento spôsob nie je udržateľný pri reálnych projektoch a v záujme zvýšenia efektivity a redukcie nákladov sa fázy môžu prekrývať [7].

1.4.2 Inkrementálny model

Inkrementálny model, ako jeho názov napovedá, má na rozdiel od vodopádového modelu viac vývojových cyklov. Cykly inkrementálneho modelu sú rozdelené do menších modulov. V prvom cykle sa vytvorí prvá verzia softvéru s obmedzenou funkcionalitou. Vytvorený softvér sa poskytne zadávateľovi k prípadným komentárom. V ďalšom cykle sa pridáva ďalšia funkcionalita, pričom je možné zakomponovať aj prípadné komentáre zadávateľa na predchádzajúcu verziu. V každom jednom cykle vznikne nová verzia, ktorú si vie zadávateľ spustiť a otestovať [7].

Po implementácii požadovanej funkcionality a zakomponovaní komentárov zadávateľa vznikne konečná verzia. Obrázok 3 schéma inkrementálneho vývoja. Na schéme je možné badať procesy špecifikácie, vývoja a validácie ako zložky súbežnej aktivity. Vzájomne sa prelínajú a je medzi nimi rýchla spätná väzba [7].



Obrázok č. 3: Schéma inkrementálneho modelu (prevzaté z [7])

Inkrementálny vývoj postupuje takým spôsobom, ako sa bežne riešia problémy. Často sa postupuje v niekoľkých krokoch k danému riešeniu, než aby sa dopredu rozpracovalo úplne celé riešenie. Tým sa zaistí aj to, že ak sa urobí niekde chyba, stačí sa vrátiť o niekoľko verzií softvéru späť [7].

Výhody inkrementálneho modelu

Prvá verzia systému obsahuje najdôležitejšie a najpotrebnejšie funkcie. Zákazník môže zhodnotiť, či mu systém poskytuje to, čo požadoval v skoršej fáze vývoja. Ak má nejaké pripomienky, stačí zmeniť poslednú verziu softwaru alebo pridať ďalšie žiadané funkcionality do ďalších verzií. Inkrementálny vývoj poskytuje tri základné výhody oproti vodopádovému modelu:

- Menšie náklady na zavedenie požiadaviek zákazníka počas vývoja softvérového riešenia.
- Ľahké získanie spätnej väzby od zákazníka na vykonanú prácu - zákazník môže komentovať jednotlivé verzie softvéru a zároveň má prehľad, koľko z požiadaviek je už implementovaných.
- Možnosť používať softvér aj keď ten ešte neobsahuje všetky funkcionality. Zákazník nemusí čakať, kým sa všetko implementuje, aby mohol softvér používať [7].

Nevýhody inkrementálneho modelu

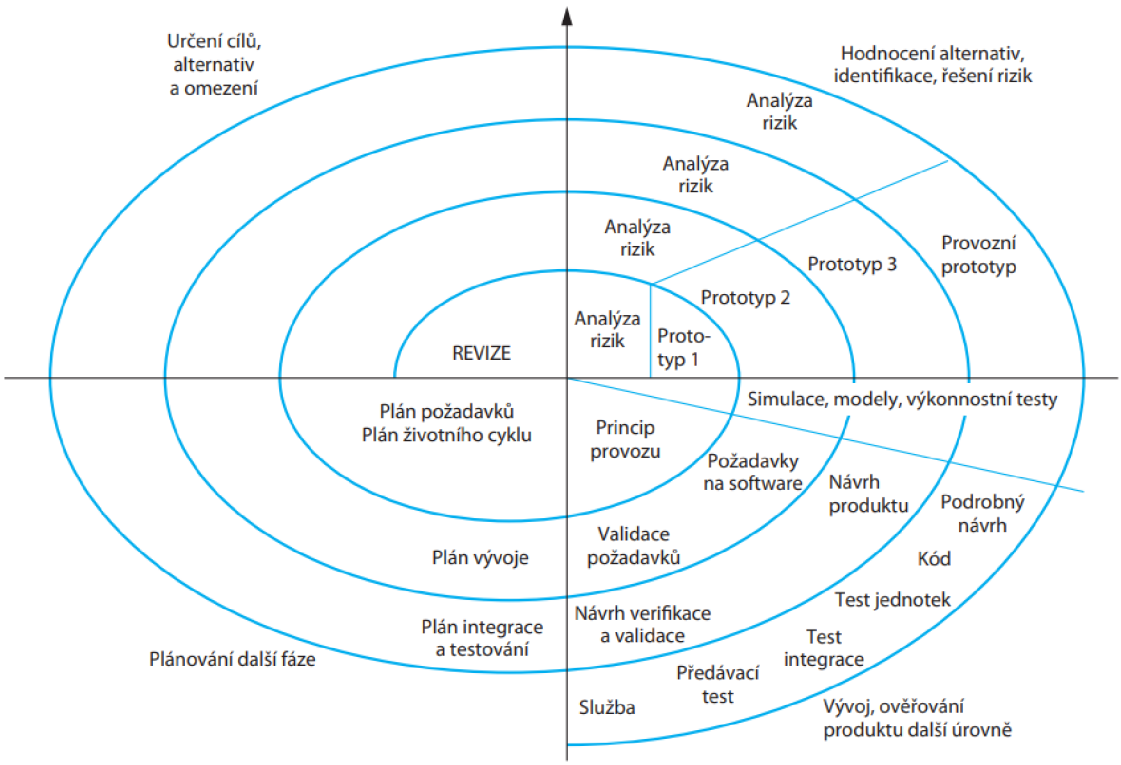
Nedostatky inkrementálneho vývoja sú výraznejšie pri veľkých a komplexných systémoch, na ktorých pracuje viacero tímov. Riadenie komplexnejšieho projektu vyžaduje stabilnú architektúru s ohľadom, na ktorú je potrebné definovať zodpovednosť pre jednotlivé tímy. Tieto pravidlá a procesy sa plánujú dopredu. Nedostatky inkrementálneho prístupu z hľadiska manažmentu:

- Proces nie je dostatočne viditeľný. Pri rýchlom vývoji systému nie je efektívne vytvárať dokumenty, ktoré odzrkadľujú stav každej verzie systému. Manažéri nevedia sledovať postup, pretože nemajú pravidelné výstupy z jednotlivých verzií.
- Pridávanie nových funkcionalít v nových verziách softvéru ovplyvňuje kvalitu jeho kódu. Bez investícií do refaktoringu, ktorý by kód zdokonalil a pripravil na ďalšie zmeny, bude štruktúra softwaru upadať. Tento úpadok sa odrazí na náročnosti zavádzania zmien a tým pádom na ich predražení [7].

1.4.3 Špirálový model

Schéma modelu softvérového vývoja založená na riziku, je znázornená na obrázku č. 4. Proces vývoja softwaru je reprezentovaný špirálou, čím sa líši od predchádzajúcich. Jednotlivé slučky špirály predstavujú fázy procesu vývoja. Špirálový model obsahuje činnosti pre riadenie rizík, ktoré ich majú obmedzovať. Každá slučka v modeli sa delí na štyri časti, ktoré označujeme aj pojmom sektory:

- Určenie cieľov - definovanie výstupov pre danú fázu projektu. Sú vyhodnotené riziká projektu, môžu sa naplánovať náhradné stratégie v závislosti na týchto rizikách.
- Hodnotenie a obmedzenie rizík - každé zistené riziko sa analyzuje a následne sa vykonajú kroky pre odstránenie daného rizika.
- Vývoj a validácia - na základe vyhodnotenia rizík sa vyberie vhodný model vývoja systému.
- Plánovanie - po každom ukončení danej slučky špirály sa rozhoduje, či pokračovať ďalšou. V prípade pokračovania sú vytvorené plány na ďalšiu fázu projektu [7].



Obrázok č. 4: Schematické znázornenie špirálového modelu (prevzaté z [7])

Výhody špirálového modelu

Hlavnou výhodou modelu je zameranie sa na analýzu rizík, ktorá pomáha zavčas eliminovať chyby a nevhodné alternatívy. Model je pripravený na rozširovanie softvérového produktu a zavádzanie prípadných zmien. Zameriava pozornosť na znovu použitie už vytvoreného a navrhnutého softvéru a začlenenie kvality do vývoja [7].

Nevýhody špirálového modelu

Jedna z hlavných nevýhod je rozpracovanie všetkých fáz každého cyklu špirály. Zameranie sa na analýzu rizík je výhodou, tak aj nevýhodou, keďže sa spolieha na odborné zhodnotenie. Ak je vyhodnotenie rizika nesprávne, je možné, že vývoj nebude prebiehať tak ako sa očakáva [7].

V Špirálovom modeli chýbajú konkrétne kroky, ktoré hovoria o tom, kedy prejsť do ďalšej fázy modelu. Je potrebné pridať míľniky, usmernenia alebo kontrolné zoznamy. V závislosti od toho, či je projekt interný alebo na zákazku, sa odvíja aj sloboda v rozhodovaní. Ovplyvnený je hlavne časový odhad dokončenia projektu.

Pri zákazkovom vývoji je potrebné nejaký mať, čo pri komplexných projektoch nie je jednoduché [7].

1.5 Agilné metodiky vývoja IT projektov

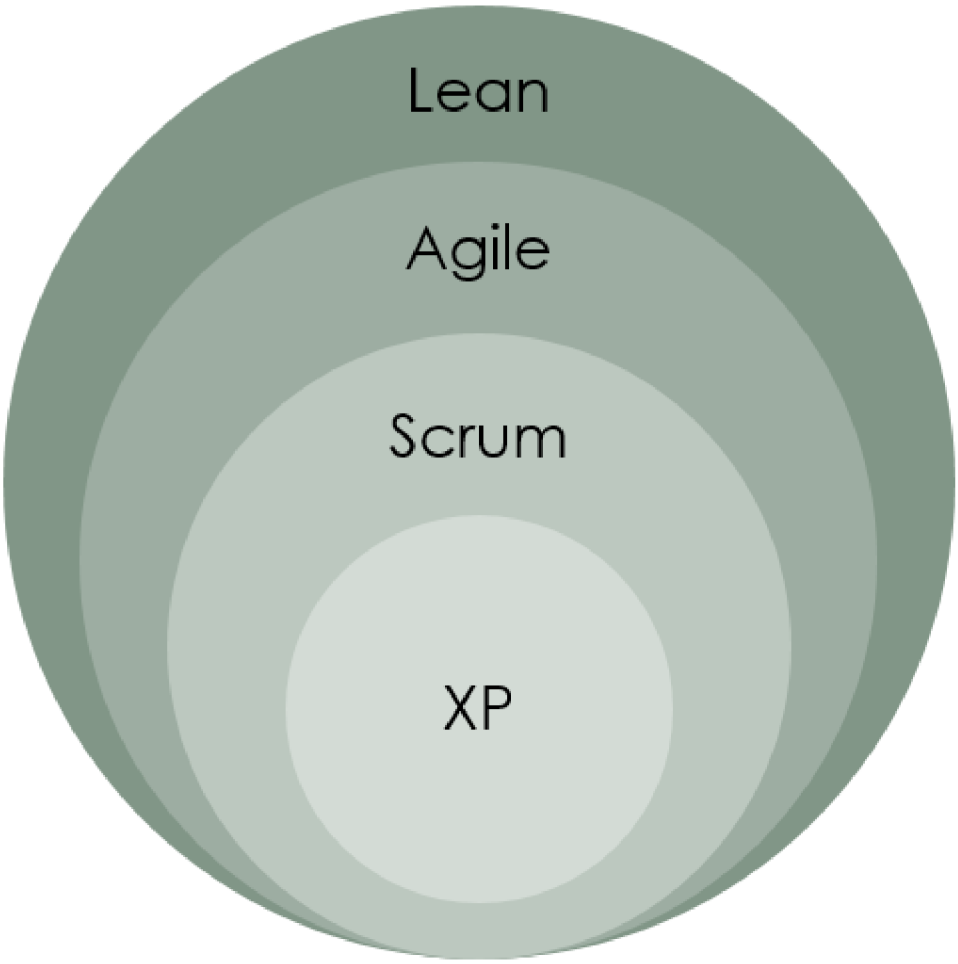
Požiadavky na rýchle zavedenie zmien do softvérových produktov a služieb sú stále vyššie. S týmito nárokmi nedokážu všetky tradičné metodiky udržať krok, a tak vznikli nové agilné metodiky. Preferujú sa v prípadoch, kedy je potrebné rýchle prispôsobenie sa zmenám v priebehu samotného vývoja. Najčastejšie zmeny požiadaviek sú od samotného zákazníka, prípadne ďalších kľúčových stakeholderov [8].

Základom agilných metodík je iteratívny proces, podpora tímovej spolupráce a otvorená komunikácia. Zároveň sa predpokladá aj zapojenie zákazníka do projektových procesov, čím sa získa jeho spätná väzba. Po každej iterácii je možné prehodnotiť priority a prípadne zakomponovať zákazníkovu spätnú väzbu do práve prebiehajúceho vývoja [3].

Agilné metodiky sa vyvinuli z takzvaného agilného manifestu, ktorý bol podpísaný 17 softvérovými vývojármi v roku 2001. Hlavná myšlienka manifestu je: objavovanie lepšieho spôsobu vývoja softvéru tým, že ho tvoríme a pomáhame s tvorením ostatným. Agilný manifest kladie váhu viac na:

- **Jednotlivcov a interakcie** ako procesy a nástroje
- **Fungujúci softvér** pred obsiahlou dokumentáciou
- **Spoluprácu so zákazníkmi** pred vyjednávaním zmluvy
- **Reakciu na zmenu** oproti dodržiavaniu plánu [9]

Rozdiel medzi agilnými metodikami je v rozsahu, ktorý pokrývajú. Z obrázku č. 5 sa môže na prvý pohľad zdať, že Agile je podmnožina Lean, prípadne Scrum je podmnožinou Agile. Pričom to však znamená, že Lean pokrýva najväčší rozsah v zmysle princípov. Agile je vyšší level - hodnoty a princípy, bez konkrétne definovaných praktík. Scrum je niekde medzi, nie je limitovaný na softvérový vývoj, ale vyžaduje časovo ohraničené udalosti (šprinty) a produkčný backlog. Extrémne programovanie je konkrétnejšie a limitované na vývojovú doménu [10].



Obrázok č. 5: Zobrazenie prepojenia agilných metodík (prevzaté z [10])

1.5.1 Lean

Lean metodika je viac filozofiou ako konkrétnymi normami a postupmi. Je definovaná pravidlami a princípmi, ktorých cieľom je efektívne využívanie zdrojov a zrýchlenie procesu vývoja. Jednou z hlavných myšlienok je odstránenie všetkého zbytočného z vývoja produktu a robiť veci až keď sú potrebné. To znamená, že sa obmedzí rozpracovaná práca na tú najnutnejšiu a tým sa sústredí na najrýchlejšie dokončenie všetkých požiadaviek [11].

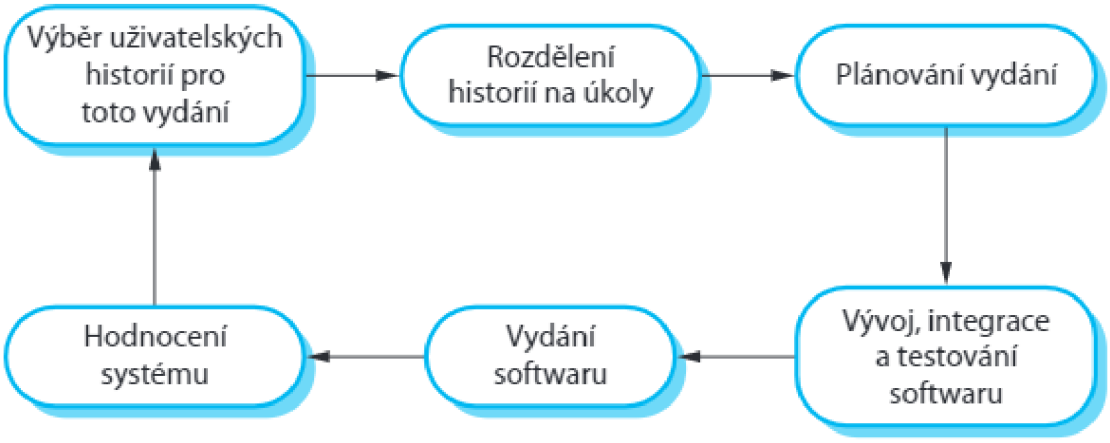
Všetky agilné metodiky a teda aj Lean sa sústredia na dodávanie hodnoty pre zákazníka. Dodávať kvalitný produkt čo najrýchlejšie a najkvalitnejšie, ako je to možné. Implementovanie konkrétnych postupov ako to docieliť, je na každej firme. Lean metodika je založená na týchto princípoch:

- Odstránenie všetkého, čo neprináša hodnotu

- Dodávajte prácu, ako najrýchlejšie to ide
- Dajte tímu dôveru a zodpovednosť
- Rozhodujte sa čo najneskôr
- Zamerajte sa na celkový výsledok
- Zlepšujte sa a učte sa v priebehu práce [3]

1.5.2 Extrémne programovanie

Extrémne programovanie pravdepodobne predstavuje najznámejšiu agilnú metódu. Jej prístup posúva uznávané optimálne metódy ako napríklad iteratívny vývoj na "extrémnu" úroveň. V extrémnom programovaní môže niekoľko programátorov behom jedného dňa vyvinúť niekoľko nových verzií systému, ktoré sa následne integrujú a otestujú. Požiadavky sa označujú ako užívateľská história, ktoré sa následne implementujú ako rad úloh. Developeri pracujú v pároch a pred začatím programovania vytvoria testy pre každú úlohu. Keď nový kód integrujú do systému, musia všetky testy prejsť úspešne. Spôsob vývoja je znázornený na obrázku č. 6. Jednotlivé vydania systému sú oddelené iba malou časovou medzerou [7].



Obrázok č. 6: Schéma cyklu vydania v extrémnom programovaní (prevzaté z [7])

Extrémne programovanie zahrňuje veľa postupov, ktoré odrážajú princípy agilných metód ako napríklad:

- Podpora inkrementálneho vývoja pomocou malých a častých vydaní systému - požiadavky sú spísané v jednoduchých zákazníckych scenároch, podľa ktorých sa rozhoduje aká funkcia bude v inkremente zahrnutá.

- Spolupráca zákazníka alebo zástupcu zákazníka priamo s vývojovým tímom, vývoja sa priamo účastní a definuje akceptačné testy systému.
- Princíp ľudia pred procesmi vychádza z programovania v dvojiciach a kolektívneho vlastníctva kódu.
- Reakcia na zmenu je postavená na pravidelnom vydávaní systému zákazníkovi, príprava testov pred začatím vývoja a refaktoring kódu, ktorý zabraňuje degradácii kódu [7].

Zákazníci sa priamo podieľajú na špecifikácii a stanovení priorít systémových požiadaviek. Plnia rolu člena vývojového tímu a s ostatnými členmi dané scenáre diskutujú a vytvárajú. Vývojový tím sa snaží vybraný scenár implementovať do nasledujúceho vydania systému. Ešte pred začatím implementácie sa každý scenár rozdelí na úlohy, odhadne sa ich prácnosť a tým prácnosť celého scenára. Následne zákazník určí priority scenárov a vyberie ten, ktorého funkcie bude po vydaní novej verzie možné použiť okamžite [7].

Požiadavky sa časom menia, neimplementované scenáre sa môžu zmeniť alebo sa nemusia vôbec implementovať. Ak je potrebné vykonať zmeny v systéme po jeho dodaní, vytvoria sa nové scenáre a znovu prejdú fázou odhadu prácnosti a prioritizáciou. Zákazník rozhodne o tom, či majú prioritu pred novými funkciami alebo nie [7].

Zásada tradičného softwarového inžinierstva spočíva v tom, že je potrebné navrhovať so súhlasom na zmeny. Developeri by mali počítať s budúcimi zmenami a navrhovať software tak, aby sa tieto zmeny ľahko implementovali. Extrémne programovanie túto zásadu neuznáva s argumentom, že nestojí za to venovať čas na zovšeobecnenie programu, aby ho bolo možné lepšie meniť. Akceptuje, že zmeny nastávajú, mení však organizáciu softwaru až vtedy, keď tieto zmeny skutočne nastanú. Tak ako pri väčšine agilných metodík si spoločnosti, ktoré prijmú vývoj extrémneho programovania, väčšinou vyberú len niektoré postupy s ohľadom na svoj špecifický cieľ [7].

1.6 Kanban

Kanban je vizuálny pracovný postup využívajúci kanban tabulu. Práca je rozdelená na malé položky, ich popis je na kartičkách alebo lístočkoch prilepených na kanban tabuli. Na tabuli sú vytvorené stĺpce, ktoré znázorňujúce rôzne fázy alebo stavy danej

položky. Základné fázy položiek sú: položka určená na vykonanie označovaná ako To Do, práve vykonávaná položka Doing a dokončená položka Done. Takéto základné rozdelenie je zobrazené na obrázku č. 7. Tabuľa môže mať fyzickú alebo elektronickú podobu a je možné ju v prípade potreby rozšíriť o ďalšie stĺpce [3].

Obrázok č. 7: Kanban tabuľa (prevzaté z [12])

Kanban sám o sebe nie je proces. Ten si z neho vytvára každá spoločnosť sama. Pri jeho implementácii, ale nesmie zabudnúť na jeho tri princípy:

- Vizualizácia pokroku
- Obmedzenie rozpracovanej práce
- Minimalizovanie času priechodu [3]

Vizualizácia pokroku

Na rozdiel od niektorých procesov nepredpisuje spôsob, akým má byť práca vykonaná. Vyžaduje, aby bol jej pokrok dokumentovaný spôsobom, ktorý je ľahké vizualizovať. Pri začiatkoch používania Kanbanu je potrebné vizualizovať celý proces tak, ako existuje. Až potom budú spôsoby zlepšenia procesu jasnejšie. Vizualizácie následne slúžia ako spôsob komunikácie stavu procesu, projektu alebo zásob [3].

Obmedzenie rozpracovanej práce

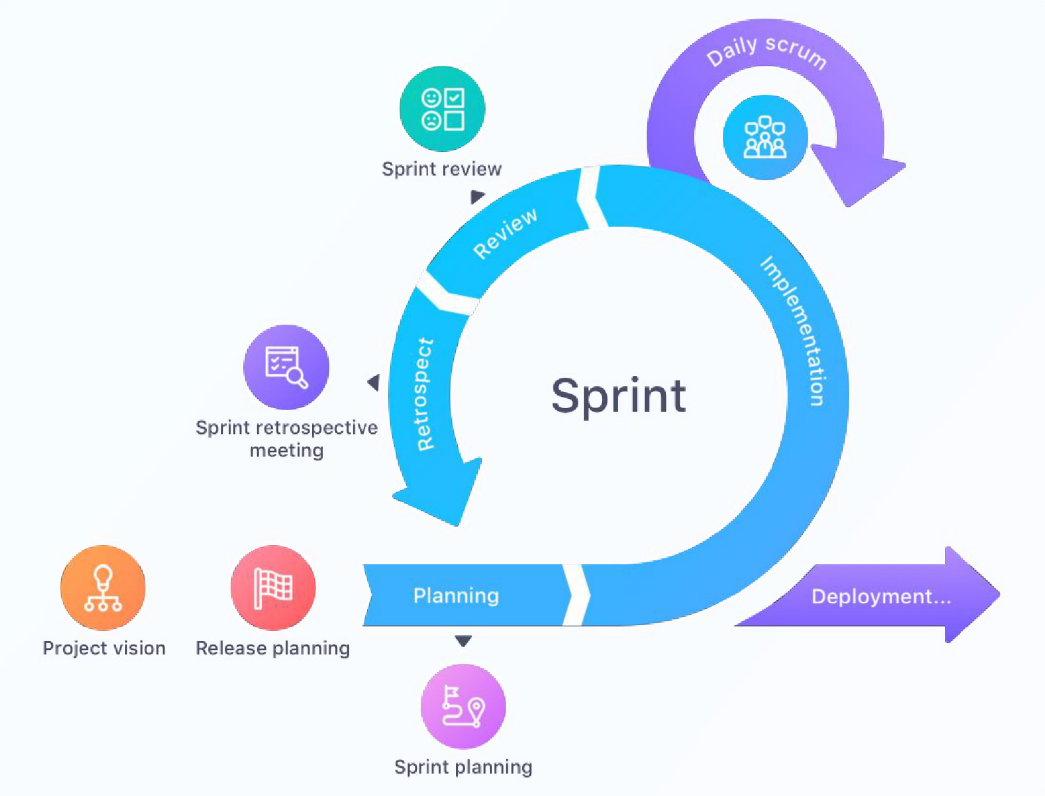
Hlavným cieľom Kanbanu je presunúť každú časť práce efektívne od začiatku až do konca s čo najmenším oneskorením. Využíva preto obmedzenie množstva práce v jednotlivých fázach tak, aby sa dali primerane zvládnuť. To znamená, ak je maximálne množstvo položiek vstave Doing4, a práve sa tam 4 položky schádzajú, nie je možné začať rozpracovávať ďalšiu. Musí sa najskôr nejaká posunúť do ďalšej fáze [3].

Minimalizovanie času priechodu

Posledným princípom je minimalizovanie času priechodu medzi jednotlivými fázami, prípadne minimalizovanie priemerného času potrebného na dokončenie úlohy. Pri použití fyzickej tabule môže byť toto meranie náročnejšie, avšak existujú softwarové nástroje, ktoré nám so sledovaním pomáhajú. Meraním času stráveného v jednotlivých stĺpcoch Kanban tabule a meraním času dokončenia úloh, je možné nájsť príležitosti na zlepšenie procesu [3].

1.6.1 Scrum

Metodiku Scrum je možné využiť aj v iných technologických odvetviach orientovaných na vývoj, nielen pri vývoji softwaru. Scrum má na rozdiel od Lean metodiky striktne definované pravidlá a presne určené procesy. Tieto procesy môžeme vidieť na obrázku č. 8. Na uvedenom obrázku sú vyjadrené jednotlivé procesy a fázy, z ktorých sa skladajú.



Obrázok č. 8: Schéma priebehu Scrum modelu (prevzaté z [13])

Na začiatku projektu sa definuje vízia projektu, určia sa jednotlivé ciele, jednotlivé očakávania ohľadom funkcií a ich času dodania. Následne sa vytvorí produktový backlog, ktorý bude obsahovať všetky funkcionality, ktoré sa na základe komunikácie so zadávateľom projektu zistia. To znamená, že backlog bude obsahovať aj možné rozšírenia produktu [3].

Po vytvorení backlogu sa priradí funkcionalitám priorita na základe produktových požiadaviek a technickej významnosti. Podľa určenia priorít sa backlog rozdelí do jednotlivých iterácií - šprintov. Požiadavky do šprintov sú zvyčajne priraďované tak, aby sa vzájomne neblokovali a dalo sa na nich pracovať paralelne [3].

Dĺžka jednej iterácie, teda šprintu, má rozsah medzi dvoma až štyrmi týždňami. Zároveň by táto dĺžka nemala presiahnuť jeden mesiac. Tento rozsah sa definuje pred začiatkom projektu a nemal by sa meniť počas priebehu projektu. Počas šprintu prebieha vývoj a testovanie požiadaviek, ktoré sa na jeho konci predajú zákazníkovi. Ten môže implementovanú funkcionalitu komentovať, či vyžadovať úpravy, a tým vytvárať nové požiadavky do produktového backlogu. Po ukončení šprintu

sa plánuje šprint ďalší, pričom počas tohto plánovania sa môžu priority jednotlivých požiadaviek meniť [3].

V prípade vytvorenia nových prioritných požiadaviek sa požiadavky s menšou prioritou posunú nižšie v produktovom backlogu. Z neho sa vyberú požiadavky do ďalšieho šprintu a takýmto spôsobom sa postupuje až do implementovania všetkých požiadaviek. Ak sa pridá veľa nových požiadaviek, je možné, že niektoré menej prioritné sa neimplementujú z dôvodov nedodržania rozpočtu alebo dopredu stanoveného času [3].

Scrum ceremónie

Scrum ceremónie sú činnosti, ktoré sa vykonávajú v každom jednom šprinte. Majú za úlohu zabezpečiť správne vykonanie šprintu. Ich dĺžka je časovo ohraničená maximálnou dobou trvania. Činnosti môžu skončiť predčasne, ak naplnia svoj cieľ. Nedochádza teda k plytvaniu času na viac meetingoch ako je potrebné [3].

Každá z ceremónií je navrhnutá tak, aby prispievala k transparentnému prehľadu nad projektom a jeho kontrole. Šprint je súhrnným predpisom činností, ktoré sa majú vykonávať, ale ich konkrétne zavedenie a vykonávanie je už na danej spoločnosti. Scrum má nasledovné ceremónie:

- Plánovanie šprintu
- Denný Scrum
- Zhodnotenie šprintu
- Retrospektíva šprintu [3]

Plánovanie šprintu

Schôdza slúži na prípravu celého tímu na ďalší šprint. Je vyžadovaná účasť všetkých Scrum rolí: vývojový tím, Scrum master a vlastník produktu (produkt owner). Plánovanie prebieha ešte pred začatím ďalšieho šprintu, zvyčajne na konci šprintu predchádzajúceho. Jeho dĺžka sa mení počtom členov tímu, zvyčajne desiatky minút až niekoľko hodín [3].

Vlastník produktu si pripraví priorizovaný list z produktového backlogu a predstaví ho tímu. Položky listu sa tiež nazývajú user stories. Počas schôdze sa diskutuje s vývojovým tímom, čo je potrebné spraviť, aby sa priorizované user stories dokončili

počas šprintu. Odhaduje sa množstvo práce na jednotlivé user story, odhad je zvyčajne v jednotkách osobohodín alebo story pointoch. Po odhade potrebného množstva položiek z produktového backlogu tím navrhne, koľko práce je schopný dodať. User story vybrané z produktového backlogu budú súčasťou šprint backlogu a tím sa zaviaže k ich dodaniu v nasledujúcom šprinte [3].

Po šprint planningu by mal byť každý zúčastnený oboznámený s rozsahom práce. Keďže je súčasťou plánovania aj odhad potrebnej práce na user stories, môže byť plánovanie dlhé. Zvykne sa preto robiť ešte jedna ceremónia, ktorá slúži len na odhadovanie user stories a odľahčí tým plánovanie. Na ňom si tím už len vyberie z odhadnutého produktového backlogu [3].

Denný Scrum

Denný Scrum je krátka schôdzka slúžiaca na informovanie o priebehu šprintu. Zaručuje, že všetci v tíme vedia, čo sa aktuálne deje. Každý člen tímu zodpovie otázky: na čom včera pracoval, na čom bude pracovať dnes a či ho blokuje niečo pri vykonávaní práce [3].

Schôdze prebiehajú na dennej báze a zúčastňuje sa ho vývojový tím, vlastník produktu a Scrum master. Dĺžka je zvyčajne 15 minút. Niekedy sa táto schôdza nazýva aj stand-up, pretože je vykonávaný v stoji. Cieľom je, aby sa schôdza zbytočne nepredlžovala [3].

Predpokladá sa úprimná komunikácia od každého člena schôdze, aby všetci vedeli, ako daný šprint prebieha. Ak niekoho niečo blokuje alebo si nevie rady s danou úlohou, táto schôdza má slúžiť aj na informovanie o takýchto skutočnostiach. Následne sa môže rozhodnúť o pozastavení práce na iných user stories a pomôcť zablokovanému kolegovi, ak je vyriešenie jeho user story prioritnejšie [3].

Zhodnotenie šprintu

Po ukončení šprintu tím prezentuje výsledky svojej práce. Každý člen tímu ukáže funkcionalitu, ktorú vyvinul počas šprintu. Naskytá sa tak priestor na pogratulovanie si k úspešnému šprintu, a tým aj k zvýšeniu morálky. Počas a po skončení prezentácie, môžu dostať spätnú väzbu od kolegov a stakeholderov daného projektu [3].

Medzi účastníkmi meetingu je vývojársky tím, Scrum master, vlastník produktu ale aj stakeholderi a ostatné tímy, ktoré na projekte pracujú. Dĺžka meetingu je závislá na tom, či sa prezentujú všetky nové funkcionality alebo len tie najdôležitejšie, ak je väčšie množstvo prezentujúcich tímov [3].

Retrospektíva šprintu

Retrospektíva sa odohráva na konci šprintu, zvyčajne s dĺžkou jednej hodiny. Účastníci sú členovia vývojového tímu, Scrum master a vlastník produktu. Jedným z agilných princípov je neustále zdokonaľovanie a retrospektíva, čo zabezpečí dosiahnutie zlepšenia [3].

Tím ohodnotí ako daný šprint prebiehal, čo sa podarilo a čo sa nepodarilo. Cieľom je identifikovať a napraviť problémy, ktoré vznikli alebo sa prejavili v priebehu šprintu. Zároveň je možné pochváliť a poďakovať za odvedenú prácu [3].

Po ohodnotení šprintu sa vyberú témy do diskusie a hľadajú sa príčiny ich pôvodu. Zvyčajne sa preberajú vyskytnuté problémy alebo niečo, čo ovplyvnilo výsledok šprintu v negatívnom smere. Scrum master pomáha tímu prísť na riešenie daných problémov a ich príčin. Následné riešenie sa budú členovia tímu snažiť uplatňovať v nasledujúcich šprintoch a ak by nepomohlo, zamyslia sa nad iným spôsobom [3].

Scrum role

Scrum má tri role: Scrum master, vlastník produktu (produkt owner) a vývojový tím. Všetka manažérska zodpovednosť je rozdelená medzi tieto role. Scrum tímy sú samo organizujúce, samé si zvolia ako budú vykonávať danú prácu. Nie sú vedené nikým mimo tímu, a preto je ich zodpovednosťou dodať všetko naplánované v daných šprintoch. V rámci tímu majú všetko čo potrebujú na dokončenie svojej práce, bez závislosti na niekom mimo tímu [3].

Vlastník produktu

Vlastníka produktu je možné definovať ako osobu zodpovednú za organizáciu produktového backlogu, a teda aj za prácu vývojového tímu. Správa produktového backlogu obsahuje činnosti ako:

- Presný popis práce položiek produktového backlogu.

- Usporiadanie jednotlivých položiek tak, aby odrážali víziu, ciele a priority produktu.
- Zaistenie, aby vývojový tím rozumel položkám v produktovom backlogu.
- Zaistenie transparentnosti, na čom bude tím v najbližších šprintoch pracovať. Vlastník je zodpovedná osoba za produktový backlog, a preto každá zmena v backlogu musí ísť cez neho. Organizácia musí rešpektovať jeho rozhodnutia. Vývojový tím sa riadi jeho pokynmi ohľadom práce, ktorú má vykonávať [3].

Vývojový tím

Je zložený z profesionálov, ktorí vyvíjajú softvérový produkt počas šprintu. V organizácii, ktorá používa metodiku Scrum, je viac vývojových tímov, ktorým zadáva prácu. Prípadne má každý tím zodpovednosť za určitú časť projektu [3].

Takýto samoorganizujúci sa tím má všetky schopnosti na dodanie položiek v šprinte. Nikto tímu nediktuje ako má tieto položky implementovať, majú slobodu v rozhodovaní sa, ale zároveň aj zodpovednosť za výsledok [3].

Veľkosť takéhoto tímu musí byť malá na to, aby bol flexibilný a efektívny. Zároveň dostatočne veľký na to, aby vedel dodať hodnotu počas šprintu. Tímy s viac ako deviatimi členmi začínajú mať problém s koordináciou. Zato tímy s menej ako piatimi členmi môžu naraziť na to, že nemajú v tíme všetky potrebné schopnosti [3].

Scrum master

Scrum master je osoba zodpovedná za dodržovanie Scrum techník a pravidiel. Má rolu vedúceho tímu, pričom jeho hlavná úloha je slúžiť ostatným. Pomáha ľuďom mimo Scrum tímu porozumieť, ktoré interakcie s tímom sú prospešné, a ktoré naopak nie sú [3].

Vedie tím k sebaorganizácii a multifunkčnosti. Motivuje ho k lepším výsledkom a odstraňuje prípadné problémy v tíme. Vlastníkovi produktu pomáha s hľadaním efektívnych techník na manažment produktového backlogu. Zaisťuje, aby vlastník produktu vedel ako usporiadať produktový backlog [3].

Mimo tímu slúži celej organizácii tým, že ju vedie a učí ako správne prijať a zaviesť Scrum. Pomáha zamestnancom a zainteresovaným stranám porozumieť Scrumu. Pričom

pracuje s ostatnými Scrum mastermi na zvýšení efektivity aplikácie Scrumu v organizácie [3].

Scrum artefakty

Scrumové artefakty poskytujú kľúčové informácie, ktorých si musia byť vedomé zainteresované strany a Scrum tím. Musia vedieť, kto sa nachádza práve vo vývoji jednotlivých úloh, ako je naplánovaná práca či ďalší postup a taktiež, ktoré jednotlivé úlohy sú na projekte dokončené. Základné Scrumové artefakty sú:

- User story
- Produktový backlog
- Sprint backlog
- Scrum board [3]

User story

User story je krátky jednoduchý popis funkcionality z perspektívy osoby, ktorá danú funkcionalitu potrebuje. Nedefinuje, ako sa má funkcionalita implementovať. Správne napísaná user story obsahuje odpovede na otázky: Kto? Čo? Prečo? Príklad formátu user story: Ja ako <typ užívateľa>, chcem <nejaká funkcionalita>, pretože <nejaký dôvod>. Príklad konkrétnej user story: Ako admin, chcem dostávať pravidelný email o všetkých neaktívnych užívateľoch, aby som ich nemusel hľadať manuálne v databáze [3].

Na základe popísania žiadanej funkcionality bude tím diskutovať o náročnosti implementácii danej požiadavky. Ak by sa zistilo, že je požiadavka veľká, rozloží sa na niekoľko menších požiadaviek. Menšie požiadavky sa lepšie odhadujú a dajú sa rýchlejšie vyhotoviť ako jedna veľká [3].

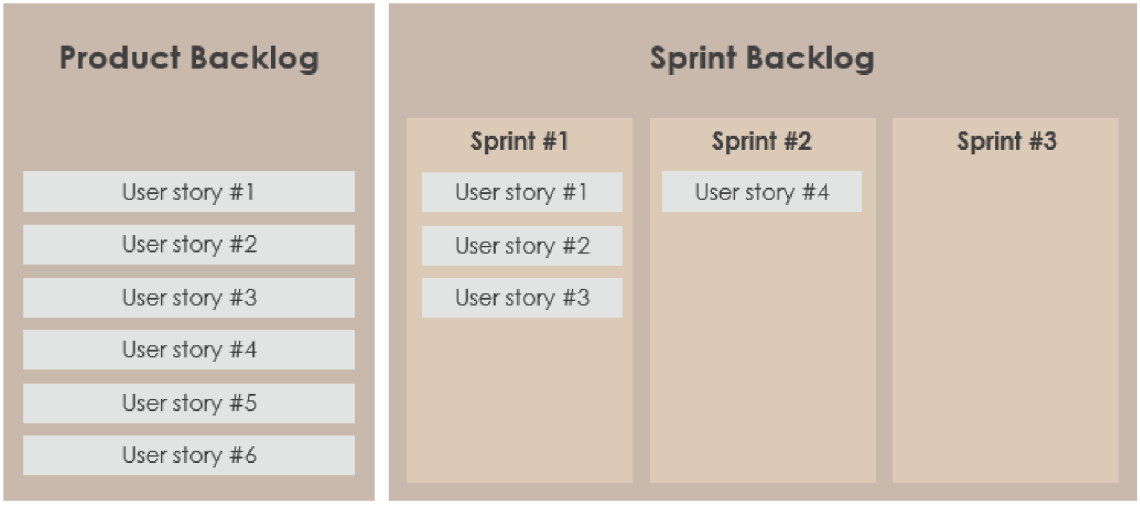
Produktový backlog

Produktový backlog je prioritizovaný list požadovaných funkcionalít na projekt. Tieto funkcionality sú spísané ako user stories s krátkym popisom. Následne k nim Scrum tím a vlastník produktu dopíše viac detailov. Na základe požiadaviek a detailov odhadnú prácu na jednotlivých user stories, ktoré vlastník produktu priorizuje. Z nich sa počas plánovania šprintu vyberú tie, na ktorých sa bude pracovať v nasledujúcom šprinte [3].

Backlog sa počas vývoja projektu môže meniť, user stories pribúdajú alebo ubúdajú. V backlogu sa nachádza nie len nová funkcionalita, ale aj požiadavky na opravu chýb, vyhradený čas na získavanie nových vedomostí a zlepšovanie kvality kódu [3].

Sprint backlog

Backlog šprintu je prioritizovaný list položiek, na ktorých sa bude v konkrétnom šprinte pracovať. Tieto položky sú priradené do jednotlivých šprintov na základe priorít z prouktového backlogu. Schéma ukážky produktového a šprintového backlogu je na obrázku č. 9.

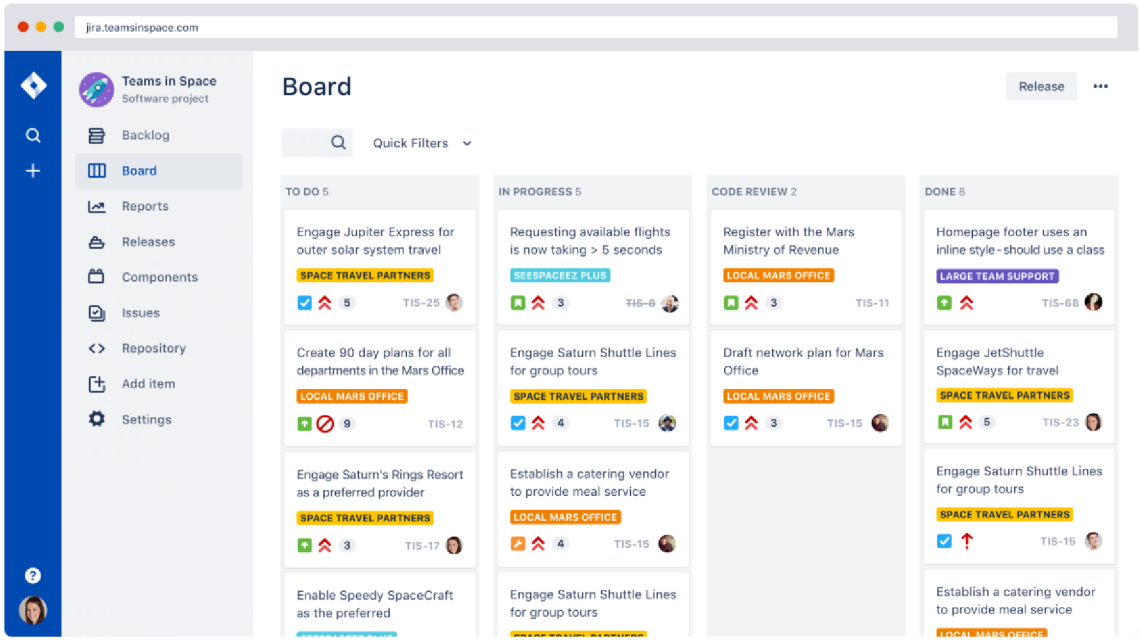


Obrázok č. 9: Schéma produktového a šprintového backlogu (prevzaté z [14])

Scrum board

Scrum board je nástroj na vizuálne zobrazenie priebehu šprintu. Vidieť na ňom backlog šprintu a progres na jednotlivých položkách v ňom. V najjednoduchšej podobe pozostáva z troch stĺpcov: To do, In progress a Done. V prípade softvérového vývoja sa zvyčajne pridávajú stĺpce ako Code review a Testing. Jednotlivé položky na tabuli posúvajú členovia tímu tak, aby zobrazovali reálny stav položky. Tabuľa môže mať fyzickú alebo virtuálnu podobu [3].

Príklad virtuálnej tabule je zobrazený na obrázku č. 10. Táto tabuľa je vytvorená v programe Jira, čo je jeden z najpoužívanejších nástrojov pre zobrazovanie priebehu vývoja v Scrum metodike.



Obrázok č. 10: Scrum tabuľa v programe Jira (prevzaté z [15])

Výhody Scrumu

Proces šprintov umožňuje rozdelenie väčších projektov na menšie časti, ktoré sú v jednotlivých šprintoch dodané. Výsledkom každého šprintu je hodnota pre zainteresované strany. Testovanie prebieha na úrovni jednotlivých user stories, čo znamená, že po dokončení šprintu je všetko v ňom aj otestované. Dáva tak priestor na menenie smeru projektu po každom jednom šprinte [16].

Krátke šprinty umožňujú získanie rýchlej spätnej väzby od zainteresovaných strán. Prípadné zmeny sa ľahšie pridávajú, pretože sa priority položiek v backlogu môžu pri plánovaní šprintu meniť [17].

Nevýhody Scrumu

Môže byť náročné plánovať a organizovať projekt, ktorý niekedy nemá jasnú definíciu. Časté zmeny, časté dodávanie produktu a neistota fungovania výsledného produktu môže zapríčiniť neistotu aj na strane vývojárov. Úspešnosť projektu záleží na schopnosti udržiavať vysokú úroveň komunikácii počas schôdzok a pri riešení problémov. Zároveň sa predpokladá zrelosť, zodpovednosť a odhodlanosť všetkých účastníkov. Ak sa niektoré z týchto predpokladov podarí nenaplniť, môže dochádzať k problémom počas vývoja projektu [16].

2 ANALÝZA SÚČASNÉHO STAVU

V nasledujúcej časti diplomovej práce je predstavená spoločnosť a jej história. Zároveň obsahuje analýzu spôsobu vývoja projektov pre bližšie porozumenie súčasnej situácie. Vyhodnotená analýza objavila miesta, ktoré majú priestor na zlepšenie.

2.1 Predstavenie spoločnosti

XYZ, s.r.o je prevádzkarom dátových centier v Českej republike. Má viac ako desať rokov skúseností v tejto oblasti. Z drobného poskytovateľa hostingu a dátových služieb vyrástla v prevádzkara datacentier s vlastnou cloudovou platformou. V roku 2018 vznikla nová divízia ABC, vyvíjajúca mobilné a webové aplikácie.

2.1.1 Predmet podnikania

XYZ, s.r.o primárne pôsobí v oblasti prevádzkovania dátových centier. Medzi hlavné ponúkané produkty patrí:

- Fyzické servery - dedikované, managed, housing
- Cloud a VPS - cloud hosting, VPS (virtuálne servery) hosting, privátny cloud
- Dátový priestor - zálohovanie, záloha na cloud, návrh zálohovacieho systému, prenájom diskového poľa alebo priestoru.

Zároveň navrhuje riešenia klientom na mieru a pomáha so vzdelávaním prostredníctvom školení. Vytvorením divízie ABC začala spoločnosť ponúkať aj softwarové riešenia. Medzi softvérové produkty patria:

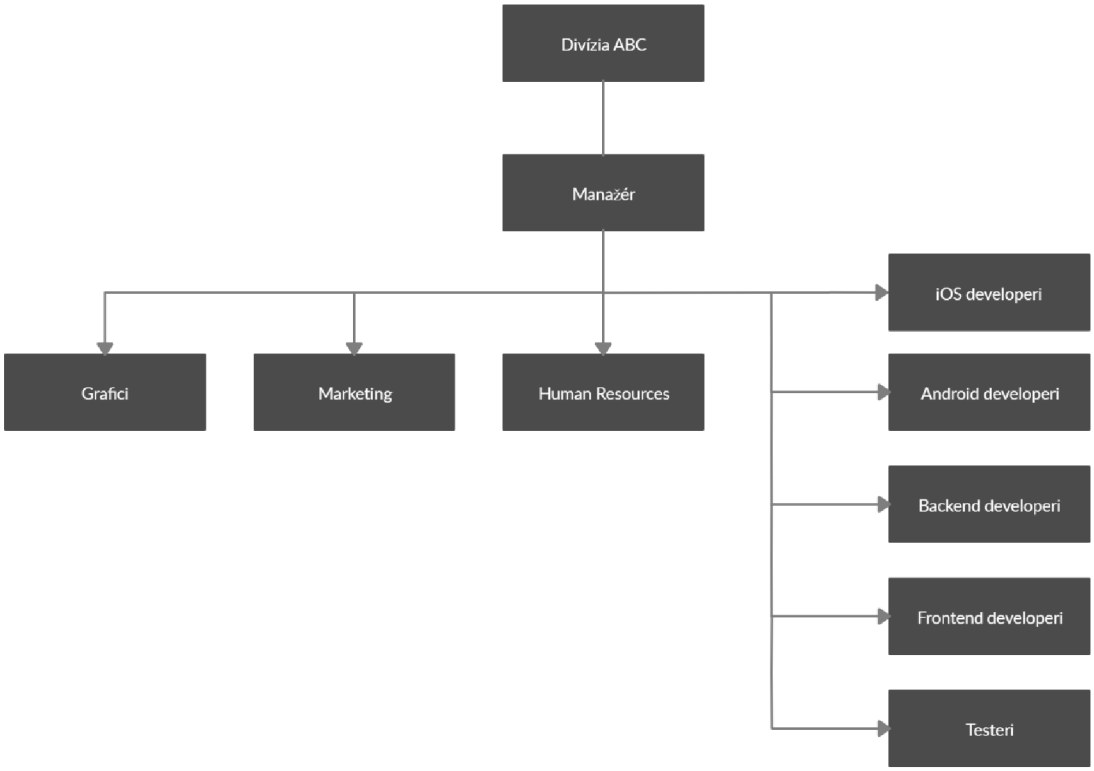
- Mobilné aplikácie - platformy iOS a Android
- Zákazkový vývoj - software a aplikácie na mieru klientom, od vývoja, otestovania až po pomoc s predajom
- SaaS - v spolupráci s partnermi ponúka prenájom hotového softwaru ako službu

2.2 Divízia ABC

2.2.1 Štruktúra divízie

Od založenia divízie sa jej štruktúra veľmi nezmenila. Väčšina častí vývoja rozšírila svoje rady o nových zamestnancov. Pribudli interní grafici, marketéri a rekruiteri, ktorých si predtým divízia požičiavala od spoločnosti.

Za každú vývojársku časť divízie je zodpovedný vedúci tímu. S pribúdajúcim počtom projektov sa časť manažmentu presunula na nich. V divízii je priateľský kolektív a rodinná atmosféra, keďže je ešte pomerne malá. To zároveň prispieva aj k vyššej produktivite samotných zamestnancov.



Obrázok č. 11: Organizačná štruktúra divízie ABC (vlastné spracovanie)

2.2.2 História divízie

Divízia začala s menším počtom ľudí a vývojom prvej internej aplikácie. Uvedenie prvej verzie produktu na trh bolo po pol roku od začiatku vývoja. Produkt bol vydaný pre platformy iOS aj Android súčasne s menšími rozdielmi vo funkcionalite. Release nových verzií sa plánoval každý pol rok, pričom vývoj prebiehal vodopádovým

modelom. Aplikácia používateľov zaujala a dokázala si ich aj udržať. Hlavnou myšlienkou aplikácie bolo streamovanie videa využitím starších zariadení určených na monitorovanie. Vytvorili k nej aj webovú aplikáciu a spolu s ďalšími dvoma produktami tvoria hlavné projekty spoločnosti.

Popri vývoji interných produktov začala divízia realizovať zákazky a to od dizajnu až po propagáciu. Spôsob vývoja sa za celú dobu nezmenil a aj pri zákazkových projektoch sa používa vodopádový model. Postupným rastom divízie na 30 zamestnancov začal aktuálny model vývoja narážať na nedostatky modelu a manažmentu. Časť vývojárov pracovala na interných produktoch, časť na externých a časť na oboch podľa potreby.

Nedostatky sa prejavovali hlavne v komunikácii so zákazníkom a v neskorom dodávaní hotového produktu. Pri menších zákazkách fungoval vodopádový model výborne, pretože bola dobre napísaná špecifikácia a nebolo nutné robiť dodatočné zmeny. Zhoršenie nastalo pri komplexnejších projektoch, ktoré zároveň vyžadujú aj viac času na realizáciu. Pri formulácii požiadaviek dochádza častejšie k prehliadnutiu funkcionalít, ktoré zákazníkovi napadnú až po dizajnovej fáze, prípadne až počas vývoja produktu.

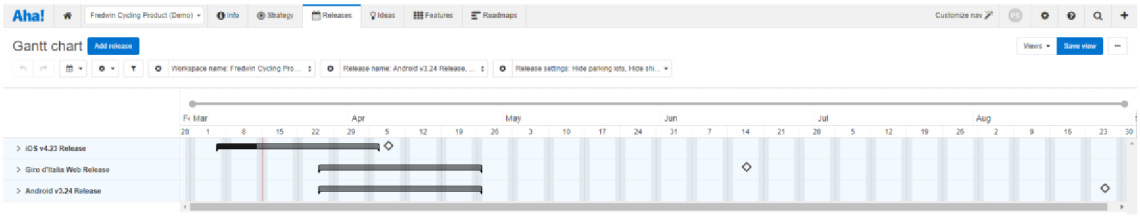
Dôsledkom nízkej frekvencie komunikácie sa informácie o zmene alebo doplnení požiadaviek dostanú príliš neskoro od zákazníka k manažmentu. Pri zmene požiadaviek počas dizajnovej fázy alebo pri začiatku implementácie, je ešte možné zakročiť tak, aby neboli zmeny časovo a finančne nákladné. V horšom prípade zákazník vznesie nové alebo doplňujúce požiadavky až pri dodaní hotového produktu. Čím neskôr vo vývojovom cykle softwaru sa začne nad zmenou uvažovať, tým bude jej zavedenie drahšie. Niekedy je možné novú funkcionalitu zakomponovať do aktuálneho stavu produktu, avšak sú prípady, kedy je lepšie začať od začiatku.

2.2.3 Softwarové nástroje

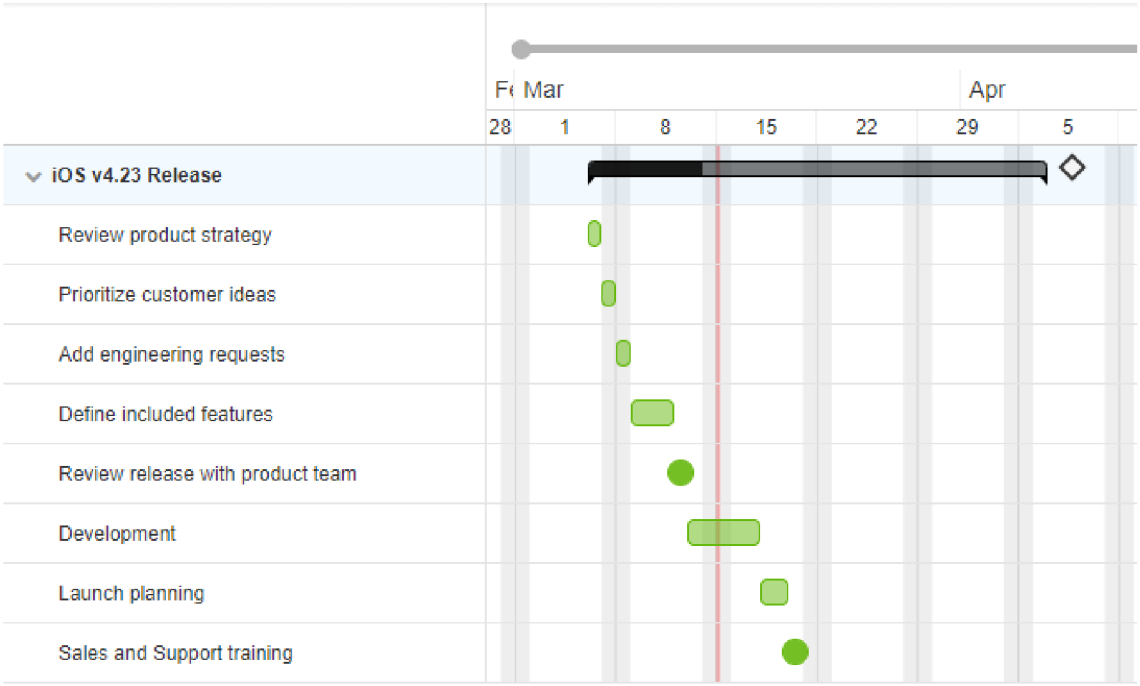
Aha!

Na riadenie projektov používa spoločnosť softvérové riešenie Aha!. Je to webová aplikácia, v ktorej sa dajú vytvárať pracovné plochy a pridávať k nim užívateľov s rôznymi právami. Slúži na vytváranie stratégií, kultivovanie nápadov a manažovanie

projektov. Projekty je možné zobraziť formou Ganttovho diagramu, ako je vidieť na obrázkoch č. 12 a 13. Zároveň je možné naznačiť dátum vydania projektu do produkčného prostredia alebo predania klientovi.



Obrázok č. 12: Ganttov diagram v programe Aha! (vlastné spracovanie podľa voľnej verzie [18])



Obrázok č. 13: Zobrazenie pod úloh Ganttovho diagramu v programe Aha! (vlastné spracovanie podľa voľnej verzie [18])

Invision

Pre návrhy grafiky do jednotlivých projektov používa divízia nástroj Invision. Majú doňho prístup všetci stakeholderi, dizajnéri, developeri a testeri. Pomocou nástroja Invision sa vytvárajú wireframi jednotlivých obrazoviek aplikácií. Umožňuje zobraziť nie len wireframi ale aj prototypy dizajnu obrazoviek a prechody medzi nimi. Každý má možnosť vzniesť pripomienky a napísať komentár k týmto návrhom. Komentáre sa následne preberú, dizajn sa prípadne upraví, nahrá znovu do Invisionu a všetci majú k dispozícii najnovšiu verziu.

Jira

Napriek vodopádovému spôsobu vývoja používa divízia nástroj Jira. Väčšinou sa tento nástroj používa pri agilne riadených projektoch, ale je možné ho využívať aj pri tradičných metodikách. Jeho použitie bolo zvolené kvôli tomu, že spoločnosť tento nástroj už používa. Každý vývojársky tím má založený svoj vlastný projekt a upravenú Jira tabuľu. Zároveň testeri zapisujú pod jednotlivé projekty nájdené chyby podľa toho, či sa jedná o chybu na backende, frontende alebo v jednotlivých aplikáciách. Zákaznícka podpora zapisuje chyby nahlásené od zákazníkov do Jira ticketov, nazývaných aj Jira problémy či vecí (z angl. *jiraissue*), na určenú Jira tabuľu. Na ňom sú tieto tickety označené príslušnou "nálepkou", ktorá ich triedi na základe toho, kto má daný problém vyriešiť. Zároveň je na danej Jira tabuli vidieť postup testovania a potvrdenie zákazníkom nájdenej chyby.

GitHub

GitHub je verzovací systém používaný developermi pre správu zdrojového kódu. Slúži im na vzájomnú kontrolu zavádzaných zmien, obmedzenie prístupu k jednotlivým projektom, prehľadnej histórii zmien a riešeniu konfliktov v zdrojovom kóde.

Skype

Divízia, ale aj celá spoločnosť používa na komunikáciu aplikáciu Skype. Je to oficiálny nástroj, a preto sa používa firemný Skype účet. Väčšina každodennej komunikácie však prebieha offline, keďže je celá divízia v jednej budove na rovnakom poschodí. Namiesto zdĺhavého vysvetľovania a písania si, porozprávajú sa o danej problematike osobne.

Skype je využívaný najmä na rýchle a krátke správy, prípadne zamestnancami, ktorí komunikujú mimo divíziu. Pokročenie vývoja v interných projektoch je prezentované formou video hovoru so všetkými stakeholdermi. Developer alebo vedúci tímu prezentuje funkcionalitu v rámci dosiahnutého míľnika.

2.2.4 Proces riadenia projektu

Projekty v divízii má na starosti hlavne manažér a vedúci tímov. Manažér vedie prvotnú komunikáciu so zákazníkom, zisťuje jeho požiadavky a potreby. Keď si ujasnia požadovanú funkcionalitu aplikácie či softvéru, diskutuje ju s vedúcimi tímov, ktorých

by sa realizácia následne týkala. Odhadnú pracnosť a čas potrebný na daný projekt a následné záležitosti vybavuje znovu manažér. Vedúci tímov preberajú aj časť role produktového manažéra za svoju časť projektu. Môžu sami na danej časti pracovať alebo len delegujú prácu v rámci tímu. Developer, ktorý danú časť projektu implementuje, následne informuje o pokroku svojho vedúceho. Vedúci tímov komunikujú s manažérom, prípadne priamo so zákazníkom a bližšie informácie transformujú do zadaní pre developerov.

Predprojektová fáza

Predprojektová fáza slúži na získanie čo najviac informácií o potrebách a problémoch, ktoré má softvérová aplikácia vyriešiť. Je to prvá fáza väčšiny projektov a nie je to inak ani v prípade vodopádového vývoja. Jej výstupom je dokument špecifikácie požiadaviek na produkt. Cieľom je definovať požiadavky čo najjasnejšie a najdetailnejšie ako je to možné.

V divízii má túto časť na zodpovednosti manažér a biznisový analytik, ktorý je zamestnancom spoločnosti. Prvá osobná komunikácia prebieha medzi klientom, analytikom a manažérom. Klient je pozvaný na osobné stretnutie až po internom rozhodnutí, či má divízia kapacitu daný projekt realizovať. Na stretnutí poskytne klient požiadavky na softvér. Analytik s manažérom sa pýtajú doplňujúce otázky, aby odstránili pochybnosti ako na svojej strane, tak na strane klienta. Po niekoľkých stretnutiach si obe strany spresnia požiadavky na funkcionalitu, bezpečnosť, použiteľnosť, prípadne dodržanie právnych predpisov. Doplnia a finalizujú dokument so špecifikáciou, podpíšu ho a následne ho použijú v ďalších fázach. Jedným z častých problémov, ktoré sa vyskytujú pri tomto modeli vývoja, je zanedbanie predprojektovej fázy. Ak sa objaví nejasnosť alebo sa vyskytne zmena v neskorších fázach, môže to mať dopad na výsledok celého projektu.

Dizajnová fáza

Na základe podpísanej špecifikácie projektu sa môže začať pracovať na dizajne produktu. Manažér s analytikom prezentujú klientovu predstavu o produkte vedúcim realizačných tímov. Hlavná časť diskusie je zameraná na to, či je možné dané funkcie implementovať na oboch platformách (Android a iOS) rovnako, prípadne aj ako webovú aplikáciu. Snažia sa odhaliť prípadné vizuálne rozdiely funkcionalít medzi

platformami. Ak sa jedná o aplikáciu využívajúcu funkcionalitu zariadení, zisťujú limitácie jednotlivých platforiem a ich verzií. Backendová časť je realizovaná v programovacom jazyku Java a prípadná frontendová časť v Reacte. Vytvorí sa technický dizajn a rozhodne sa o developeroch z jednotlivých tímov, ktorí budú na projekte pracovať. Môžu to byť aj vedúci tímov, avšak tí častejšie len vypomáhajú, prípadne rozbiehajú nový projekt.

Dizajnér navrhne wireframi a jednotlivé obrazovky mobilnej či webovej aplikácie. Pri ich vytváraní sa zároveň zamýšľa aj z perspektívy koncového užívateľa. Snaží sa vcítiť do používateľa, pre ktorého je aplikácia vytváraná, aby vytvoril dizajn, ktorý je elegantný a zároveň nie je komplexný. Návrhy pre jednotlivé platformy a webové prehliadače sa ešte pred implementáciou musia prebrať s vedúcimi tímov a nechať schváliť manažérom a klientom.

Implementačná fáza

V implementačnej fáze začnú developeri programovať požadovaný produkt na základe produktových požiadaviek a dizajnu, ktorý bol vytvorený vo fáze predtým. Vodopádový model sa spolieha na to, že sú už všetky potrebné informácie zdokumentované a nebudú sa meniť. Vývojári pracujúci na projekte si dôkladne prezrú navrhnuté wireframi a obrazovky aplikácie. Začnú vytvárať hlavnú kostru, ktorá zatiaľ nepotrebuje dáta z backendu. V tejto fáze je dôležitá komunikácia medzi jednotlivými developermi, ktorým bol projekt priradený. Nastávajú však situácie, kedy sa informácie predávajú cez viac ľudí, než by bolo potrebné, čím sa stráca prehľadnosť alebo úplne celá informácia.

Backendový developer začne s prípravou endpointov a návrhom databázy, do ktorej sa budú ukladať dáta z aplikácie. Záleží od produktu, či budú dáta primárne držané v aplikácii na zariadení alebo na serveri. Každý vývojár si vytvára svoju časť celkového produktu, na základe dohodnutých míľnikov. Pri ich dosiahnutí sa jednotlivé časti buď vzájomne integrujú alebo sa pokračuje v samostatnom vývoji. V zmysle integrácie sa myslí komunikácia aplikácií medzi platformami a backendom.

Vedúci tímov programujú, ale zároveň sú aj produktovými manažérmi. Pri počte developerov a projektov nie je možné, aby pracovali na všetkých. To je dôvodom, že niektoré projekty len manažujú. Nemajú teda úplný prehľad nad stavom kódu,

za ten zodpovedajú jednotliví vývojári. V pravidelných intervaloch sa spolu stretávajú a zisťujú aktuálny stav vývoja. Pomáhajú s riešeniami prípadných problémov, ktoré projekt brzdia. Míľniky sú stanovené v jeden až dvoj mesačných intervaloch, kedy sa v prípade ich dosiahnutia prezentujú pokroky a manažujú možné riziká, ktoré by posunuli dátum dokončenia a odovzdania projektu.

Fáza testovania

Po implementácii funkcionalít, prichádza na radu ich testovanie. Vzhľadom na vyšší počet projektov ako je testerov, je nutné rozdelenie práce. V bežnom priebehu vývoja projektu má každý na starosti niekoľko aplikácií, ktoré testuje, pričom nemá detailný prehľad o iných. Tento prístup funguje do bodu, kým nie je niekto na dovolenke alebo práce neschopný. Aplikácie, ktoré výhradne testoval si musia rozdeliť zvyšní testeri, ktorí nemajú až tak hlbokú znalosť problematiky. To môže spôsobiť zavedenie a neodhalenie chýb, ktorým sa dalo predísť.

Vedúci testerského tímu priraďuje prácu na projektoch, manažuje proces testovania a referuje priebežné výsledky manažérovi. Sám má na starosti niektoré aplikácie, pri ostatných si udržiava prehľad nad postupom práce, počtom chýb a ich závažnosť. Výsledkom predchádzajúcich fáz je dokumentácia požiadaviek, grafický a technický dizajn - to čo treba naprogramovať s použitím konkrétnych technológií.

Testeri majú k dispozícii daný technický dizajn a požadované špecifikácie od klienta. Na základe nich môžu začať dopredu pripravovať testovacie scenáre, ktorými overia správnu funkciu hotového produktu. Nikde však nie je poriadne popísaný spôsob testovania menších častí, ktoré sa postupne vyvíjajú medzi jednotlivými míľnikmi. Tester musí na základe dielčej implementovanej funkcionality vymyslieť, ako ju čo najlepšie otestovať. Kým má testovaná časť grafické rozhranie, pomocou ktorého si vie overiť správnosť testu, je výsledok testu dostatočne vierohodný. Pri nutnosti testovania čisto backendu, môže niektorým chýbať hlbšia znalosť a potrebujú pomoc s použitím nástrojov na testovanie od developerov. Komunikácia medzi developerom a testerom je dôležitá. Základom dobrého testu je nezávislé overenie funkcií vzhľadom na informácie získané od developera.

Výsledkom tejto fázy je overenie akceptačných kritérií v dokumente so špecifikáciou funkcionalít produktu, spísaním nájdených chýb a výškou ich rizika pre vydanie produktu na trh.

Na základe týchto informácií a časového plánu sa rozhodne, ktoré chyby je potrebné opraviť. Opravovanie a opakované testovanie prebieha do bodu, kým sú nájdené chyby natoľko závažné, aby ovplyvnili plynulé vydanie produktu. Otestovaný produkt sa predá klientovi na schválenie, po ktorom sa presunie do prevádzky.

Ukončenie projektu

Predposlednou fázou vodopádového modelu je nasadenie systému do klientovho prostredia alebo vydanie na trh. Po otestovaní produktu a akceptovaní z klientovej strany sa naplánuje vydanie aplikácii na trh, či spustenie webovej aplikácie na požadovanej doméne. Pripraví sa produkčné prostredie, ktoré môže byť spravované v rámci serverov spoločnosti alebo prenesené na servery klienta. Divízia ponúka možnosť propagácie nových aj stálych aplikácií zverejnených v Apple store alebo v Google play. Používa k tomu prostriedky a komunikačné kanály, ktoré využíva aj na propagáciu svojich interných projektov.

Údržba produktu

Úspešné otestovanie a prijatie projektu klientom nemusí byť vždy koniec spolupráce. Klient sa môže rozhodnúť pre použitie serverov spoločnosti. Divízia ponúka možnosť využitia support tímu v prvých mesiacoch od vydania. Zároveň s tým ponúka opravenie všetkých nahlásených chýb, ktoré sa prejavia až v produkčnom prostredí pri používaní aplikácie väčším množstvom koncových užívateľov.

Práca nad rámec špecifikácie

Po ukončení implementácie a testovania sa odovzdá produkt klientovi na akceptačné testovanie. Keď klient uvidí hotový produkt, často k nemu začne vznášať pripomienky a komentáre typu: “ešte by som pridal takúto funkcionalitu”, či “myslel som si, že to tam bude” , “toto tak nemá fungovať” alebo “predstavoval som si túto časť inak”. Použitím vodopádového modelu pri komplexnejších projektoch táto situácia nastáva často. Pri jednoduchších a lepšie špecifikovaných projektoch to problém nebýva. Príčina tohoto problému je, že klient zistí až počas priebehu implementácie projektu,

že by chcel niečo zmeniť alebo pridať. Pred začatím projektu sa analyzovali klientove požiadavky na funkcionalitu, odhadla sa pracnosť a čas potrebný na realizáciu. Nie je možné zavádzať zmeny, na ktoré sa nevyhradil čas.

Môže nastať aj situácia, že je niečo implementované inak, ako by si klient predstavoval. Preto je dôležitá čo najpodrobnejšia špecifikácia, ktorá by sa nemala podceňovať. V takýchto prípadoch rozhoduje práve tá. Považovať niečo za chybu začneme až vtedy, ak je to v špecifikácii explicitne napísané inak. Ak tam nie je funkcionalita vôbec spomenutá, alebo je opísaná inak, tak to nie je chyba v implementácii ale v špecifikácii. Divízia je ešte pomerne mladá, a preto častokrát v záujme zachovania dobrých obchodných a partnerských vzťahov rieši prevzatie časti nákladov. O to akú veľkú časť, väčšinou záleží na konkrétnom klientovi a spolupráci s ním. Pri zákazkovom vývoji projektov záleží z dlhodobého hľadiska aj na spokojnosti klienta.

Komunikácia počas projektu a zdieľanie skúseností

Všetky tímy majú svoju miestnosť, v ktorej sedia len členovia daného tímu. Počas vývoja projektov spolu komunikujú na dennej báze, ale čo sa týka zdieľania skúseností, toto väčšinou prebieha formou otázka-odpoveď a len vtedy, keď potrebuje niekto s niečím poradiť. Seniornejší kolegovia odporučia návrhový vzor na vyriešenie konkrétneho problému alebo sami implementujú základy nového projektu, na ktorom následne pracuje juniornejší kolega. Produktovú perspektívu ohľadom jednotlivých projektov poznajú okrem manažéra iba vedúci tímov. Ostatní vývojári ju zistia až na prvom stretnutí s vedúcim ohľadom nového projektu, na ktorom budú pracovať. Následne získavajú ďalšie informácie na základe dostupných dokumentov, špecifikácií a dizajnov. V prípade nejasností sa pýtajú svojho vedúceho, ktorý buď odpoveď vie alebo musí rozhodnúť manažér. Komunikácia medzi tímami prebieha len vtedy, keď je potrebné zosúladiť komunikáciu a integráciu projektových častí ako napríklad frontend a backend. Vyskytuje sa tu len minimálne zdieľanie skúseností a znalostí, čo je niekedy premrhaná príležitosť.

2.3 SWOT analýza divízie

Pomocou SWOT analýzy sa zhrnú skutočnosti vyplývajúce z analýzy riadenia projektov. Rozdelíme ich na do štyroch kategórií: silné stránky, slabé stránky, príležitosti a hrozby. Vnútorné prostredie organizácie je charakterizované silnými

a slabými stránkami. Vonkajšie prostredie prostredníctvom príležitostí a hrozieb. Vnútorné prostredie je možné ovplyvniť, pričom vonkajšie veľmi ťažko.

2.3.1 Silné stránky

- Lojalita zamestnancov
- Skúsení a kvalifikovaní zamestnanci
- Dobrá povesť spoločnosti
- Zázemie spoločnosti
- Ziskové interné projekty
- Ponuka komplexného riešenia
- Dôkladné testovanie

2.3.2 Slabé stránky

- Rozdelenie rolí a kompetencií
- Komunikácia so zákazníkom počas implementácie
- Komunikácia medzi zamestnancami
- Relevancia produktu od návrhu po vydanie na trh
- Reakcia na zmeny
- Neúplnosť špecifikácie pri komplexnejších projektoch
- Zdieľanie skúseností a znalostí

2.3.3 Príležitosti

- Záujem nových zákazníkov
- Optimalizácia riadenia IT projektov
- Zlepšenie komunikácie so zákazníkom
- Zapojenie zákazníka počas implementácie

2.3.4 Hrozby

- Nedostatky na trhu práce
- Lepšia ponuka služieb zo strany konkurencie
- Zmena ekonomickej situácie

2.3.5 Informácie vyplývajúce zo SWOT analýzy

Zo SWOT analýzy vyplýva, že divízia disponuje kvalitným zázemím spoločnosti, čo jej pomáha šíriť povedomie o sebe na trhu. Medzi jej silné stránky patria skúsení, kvalifikovaní a lojálni zamestnanci, ktorí odvádzajú profesionálnu prácu. To je vidieť na implementácii a kvalite odovzdávaného produktu. Podlieha dôkladnému priebežnému testovaniu počas vývoja a následne finálnemu testu po ukončení implementácie. Klientovi sa odovzdáva odladený produkt na akceptačné testovanie, pri ktorom by nemal nájsť žiadne chyby. Divízia ponúka komplexné riešenie projektu od jeho návrhu, až po jeho správu po ukončení vývoja. Táto možnosť môže byť výhodná pre menšie firmy, ktoré nemajú prostriedky na vlastný vývoj aplikácie alebo na jej správu po vydaní.

Divízia má veľký priestor na zlepšenie v rámci komunikácie so zákazníkom a medzi jednotlivými zamestnancami. Počiatočné stretnutia so zákazníkom sú frekventované, pretože je potrebné spísať špecifikáciu a ujasniť si grafický dizajn produktu. Keď sú tieto fázy projektu ukončené, komunikácia s klientom upadáva kvôli predpokladu, že sú všetky potrebné informácie už zdokumentované. Implementácia a testovanie prebieha niekoľko mesiacov, po ktorých klient uvidí výsledný produkt. Počas týchto mesiacov sa jeho vízia a potreby daného produktu mohli zmeniť, a preto sa stáva, že by chcel nejakú funkcionalitu pridať alebo zmeniť. Vodopádový model nie je na takéto zmeny pripravený, čo zvyčajne spôsobí predĺženie projektu a zvýšenie nákladov. Zároveň môže nastať situácia, kedy vyvíjaný produkt stratí na atraktivite, ktorú mal ešte v čase vývoja alebo pôvodného dátumu vydania. To môže viesť k strate ziskov klienta, čo nevrhá na divíziu dobré svetlo.

Zamestnanci medzi sebou komunikujú na dennej báze, avšak rozdelenie rolí a kompetencií sa niekedy môže zdať chaotické. Manažér je prioritná osoba, ktorá komunikuje s klientmi počas celého projektu. Vo fáze dizajnu je možné, že sa pri stretnutiach zapojí aj dizajnér, aby pomohol s realizáciou klientových predstáv. Postupným zväčšovaním sa divízie sa vedúcim vývojárskych tímov pridali zodpovednosti na sledovanie pokroku projektov. Ten pravidelne zisťujú od kolegov, ktorí na jednotlivých projektoch pracujú a pokroky prezentujú manažérovi počas pravidelných stretnutí. Takýmto spôsobom komunikuje so štyrmi vedúcimi.

Keďže sa vyvíja viac projektov zároveň a každý je v inej fáze vývoja, vedúci tímov majú niekedy problém udržať si komplexný prehľad nad stavom projektov. Zároveň vývojári, ktorí pracujú na projekte komunikujú medzi sebou priamo a až keď je potreba, tak zapájajú do implementačných diskusií svojich vedúcich. S vedúcim prevažne komunikujú až pokrok, ktorý bol na projekte vykonaný.

O ponuku služieb divízie je čoraz väčší záujem medzi novými zákazníkmi, ak by sa však zmenila ekonomická situácia, mohlo by dôjsť k výraznému poklesu dopytu. Zároveň rýchlim rastom nových príležitostí môže divízia prísť do bodu, kedy nebude schopná prijímať nové projekty z nedostatku ľudských zdrojov. Riadenie komplexnejších projektov použitím vodopádového modelu sa začína ukazovať ako neefektívne, čo môže byť aj dôsledkom spôsobu manažovania týchto projektov viacerými ľuďmi. Týmito skutočnosťami sa vytvorila príležitosť pre optimalizáciu súčasného stavu riadenia IT projektov.

2.4 Zhrnutie analytickej časti

Divízia vidí nárast záujmu nových klientov, s čím zároveň pribúdajú aj komplexnejšie projekty. Z analýzy divízie vyplýva, že práve pri týchto komplexnejších projektoch sa aktuálna metóda riadenia projektov javí ako neefektívna. Vodopádový model nie je určený na prispôsobenie sa zmenám a flexibilné reagovanie na ne. Väčšina potrebných zmien je tiež spôsobená slabou komunikáciou so zadávateľom špecifikácie po jej dokumentovaní. Vývoj prebieha podľa špecifikácie a dizajnu, ktorý sa na začiatku so zákazníkom dohodol a až po jeho dokončení sa zákazníkovi odovzdáva hotový produkt. Ďalším problémom, ktorý je potrebné vyriešiť, je presné definovanie rolí. S pribúdajúcim počtom projektov sa zodpovednosti za kontrolu priebehu projektov presunuli na vedúcich tímov, pričom ani oni nie sú vždy schopní mať úplný prehľad.

Nedostatok pracovníkov v oblasti produktového manažmentu je riešiteľný jedine náborom nových zamestnancov, prevzatím špecialistov z inej časti firmy alebo preškolením niekoho z divízie. Väčšinu zvyšných zistení z analýzy je možné vyriešiť zavedením agilnej metodiky Scrum. Optimalizuje sa proces vývoja produktu, v rámci Scrum tímu má každý presne určenú svoju rolu a komunikácia v tíme je podporovaná, až priam vyžadovaná prostredníctvom Scrum meetingov. V Scrume sa implementuje produkt iteratívne, a preto je možné reagovať na zmeny flexibilne.

Zároveň sa zefektívni komunikácia s klientom, bude pravidelná, keďže sa bude účastniť prezentácie dokončenej práce po každej iterácii. To by malo eliminovať stratu financií a času vynaloženého na zmenu výsledného produktu po jeho odovzdaní zákazníkovi.

3 NÁVRH RIEŠENÍ A PRÍNOS NÁVRHOV RIEŠENIA

Na základe vykonanej analýzy súčasného stavu pre divíziu ABC je v tejto časti práce navrhnuté a detailne popísané zefektívnenie vedenia projektov zavedením agilnej metodiky Scrum. Je opísané zloženie Scrum tímu, priebeh šprintov a jednotlivých ceremónií. Následne sa uvedú možné podporné nástroje a techniky pre meranie úspešnosti projektov, ktoré sú vedené agilným spôsobom. Záverom je spracované ekonomické zhodnotenie a prínosy zavedenia návrhu pre divíziu.

Prechodom na agilný spôsob riadenia bude možné odstrániť väčšinu negatívnych pozorovaní z analýzy, ktoré sa začali prejavovať pri aktuálnom vedení projektov. Medzi tieto problematické oblasti patrí:

- Slabá komunikácia so zákazníkom a taktiež medzi jednotlivými členmi tímu
- Málo priestoru pre prácu s meniacimi sa požiadavkami, ktoré zákazník vnesie až počas vývoja
- Neúplná špecifikácia požiadaviek
- Testovanie až po dokončení vývoja
- Neskoré odhalenie chýb
- Zdieľanie skúseností a znalostí

3.1 Zavedenie agilného prístupu

Zavedením agilného prístupu je potrebné definovať jednotlivé role: Scrum master, Product owner a členovia vývojového tímu. Vývojový tím je zložený z jednotlivých členov, z ktorých má každý rôzne znalosti potrebné pre úspešné dokončenie projektu. Product owner rozumie potrebám zákazníka a zároveň hodnote, ktorú mu vývojový tím v každej iterácii prináša. Scrum master to drží celé pohromade, pomáha celému tímu, aby dobre spolupracoval a dodržoval zásady Scrumu. Pomáha so vzdelávaním členov tímu a všetkých v organizácii ohľadom Scrumu a iných soft skill oblastí.

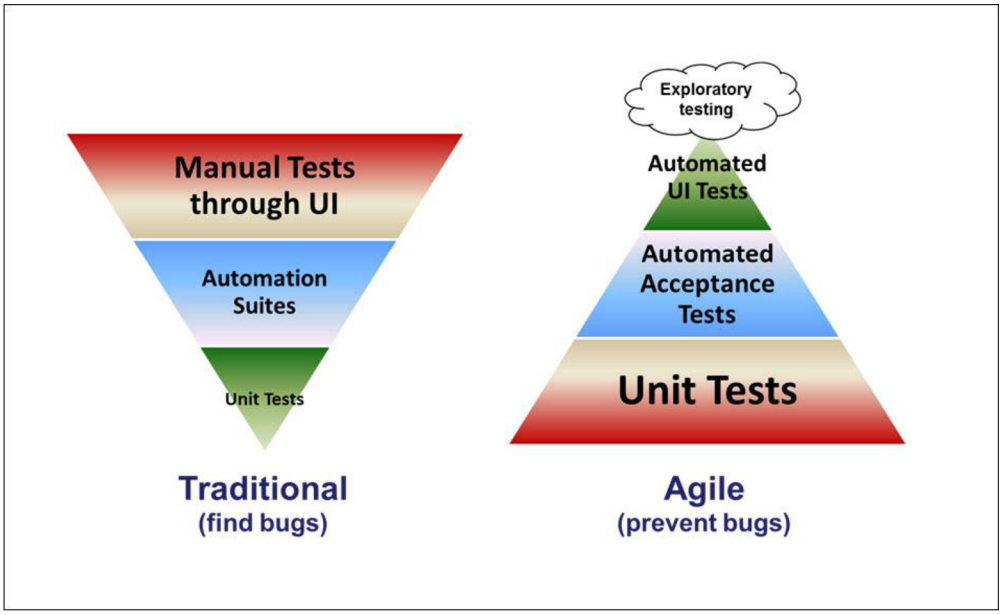
3.1.1 Definovanie rolí

Vývojový tím

V divízii bude tím zložení zo siedmich členov, pričom má každý z nich svoju rolu. Tím sa skladá z dvoch QA inžinierov (testerov), troch backend programátorov a dvoch frontendprogramátorov.

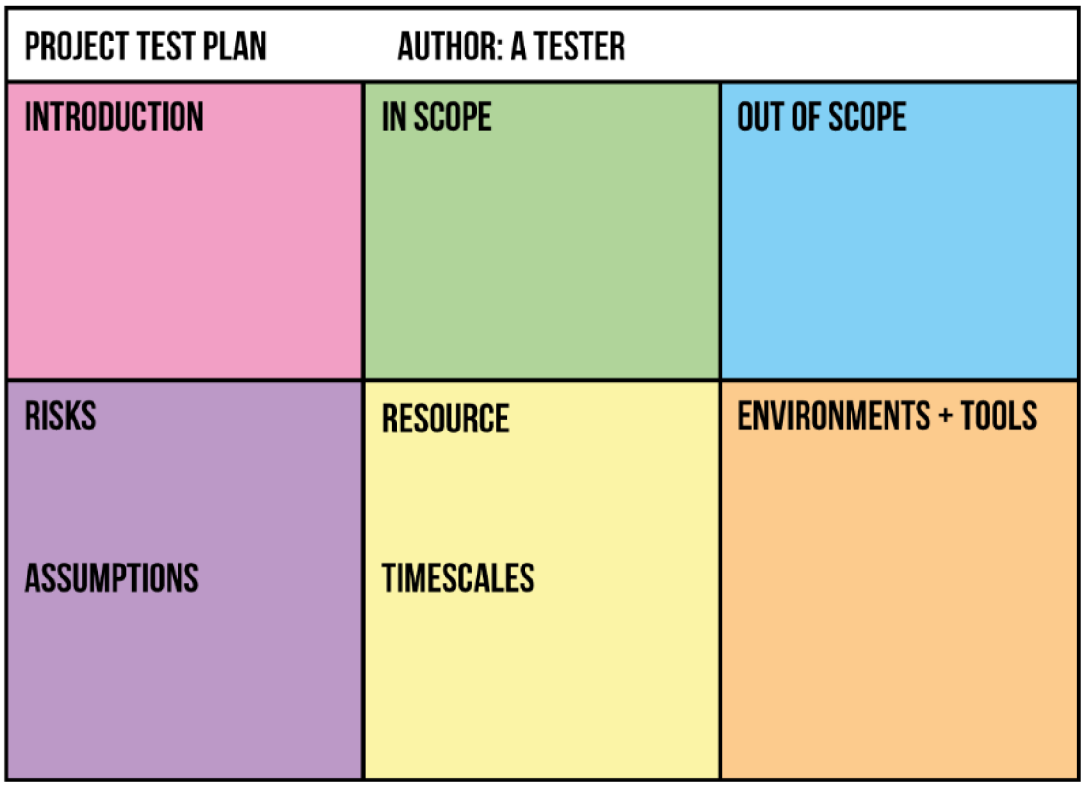
QA inžinieri pomáhajú Product ownerovi pri zostavovaní požiadaviek od zákazníka, zapájajú sa pri produktovom dizajne ale aj technickom dizajne, ktorý vytvárajú developeri na základe produktového. Postup ich práce v tomto smere začína analýzou produktu, jeho funkcionalít a výsledných akceptačných kritérií spísaných v požiadavkách na produkt. Na základe týchto informácií začnú vytvárať testovaciu stratégiu, dokument, ktorý bude definovať rozsah testovania - komponenty systému ktoré budú a nebudú v rozsahu.

QA inžinier vyberie typy testov, ktoré budú použité. Ako je znázornené na obrázku č. 14, pri agilnom vývoji sa kladie dôraz na prevenciu chýb pomocou väčšej sady automatizovaných testov. Unit testy implementuje developer, aby odhalil chyby priamo pri implementácii. Automatizované testy píšu QA inžinieri, ale môžu aj samotní developeri.



Obrázok č. 14: Testovacia pyramída pri tradičných a agilných metodikách (prevzaté z [19])

Následne definuje QA inžinier cieľ testu a testovacie kritériá, ktoré musia byť splnené, aby sa mohol plán považovať za splnený. Definuje zdroje potrebné na realizovanie plánu, prostredie, na ktorom sa bude test vykonávať. Odhadne približnú dĺžku na základe svojich skúseností s podobnými projektmi. Výsledkom bude test plán, ktorý môže mať pri komplexnejších projektoch formu viacstranového dokumentu. Ten je síce podrobný, ale nie vždy má naň manažment čas, preto je lepšie použiť prehľadnejšie riešenie vo forme jednostranového testovacieho plánu, ako je znázornené na obrázku č. 15.



Obrázok č. 15: Test plán na jednu stranu (prevzaté z [20])

QA inžinieri testujú dokončenú funkcionalitu v šprintoch, dokumentujú finálne testovanie po ukončení projektu a pomáhajú zodpovedať otázky zákazníka na hotový produkt.

Developeri v rámci Scrum tímu definujú technické požiadavky, pričom vychádzajú z produktových. Vytvoria technický dizajn, kde opíšu technické riešenie na základe predchádzajúcich požiadaviek (produktové a technické). Tento technický dizajn sa následne implementuje počas jednotlivých iterácií, nazývaných šprinty. Developeri implementujú jednotlivé tikety v šprinte a po ich dokončení dajú svoj kód na review

kolegom. Takýmto spôsobom si po sebe kontrolujú, či porozumeli správne zadaniu úlohy a či pokryli testami časť, čo práve implementovali. Ak niektorý developer vykonávajúci review zistí, že je možné daný problém vyriešiť efektívnejšie, navrhne svoje riešenie, ktoré sa po diskusii môže ale nemusí použiť. Všetky spomínané postupy majú za cieľ udržovať kvalitu kódu, dopredu identifikovať problémové oblasti a pripraviť sa na ne. Počas prezerania kódu a vznášaním pripomienok si developeri vymieňajú skúsenosti a znalosti.

Product owner

V rámci Scrum tímu je určený jeden Product owner, ktorý má na starosti komunikáciu so zákazníkom a ostatnými stakeholdermi. Definuje ich požiadavky spolu s QA inžinierom, na základe ktorých vytvára produktový dizajn. Ten sa diskutuje so stakeholdermi a po jeho schválení vytvára projektový Backlog - zoznam úloh, ktoré musia byť implementované. Celý backlog zoradí podľa priority, podľa ktorej sa budú jednotlivé úlohy vyberať do nasledujúceho šprintu. Backlog by mal vizualizovať poradie krokov, ktoré budú vykonané pre dokončenie projektu.

Produkt owner neriadi tím, neprikazuje im, čo majú alebo nemajú robiť. Jeho úlohou je podpora počas vývoja zodpovedaním otázok ohľadom funkcionality produktu. Objasňuje bližšie navrhnutú funkcionalitu, ak nie je jej popis dostatočne jasný. Rozhoduje v oblasti produktových otázok, ak sa vyskytne problém alebo možnosť viacerých variant riešenia. Príkladom môže byť výber dizajnu alebo rozmiestnenie komponentu, ak neboli vopred určené. Zastupuje záujmy klienta a preto musí vedieť, čo presne za výsledok klient očakáva. Ak sú okrem klienta zainteresovaní aj iní stakeholderi, musí mať na mysli aj ich záujmy. Na začiatku každej iterácie, šprintu, stanovuje cieľ pre danú iteráciu, prioritu, ktorá by mala byť doručená, ak by sa aj nič iného nepodarilo. Na konci každého šprintu a počas plánovania ďalšieho môže zmeniť poradie požiadaviek v backlogu na podnet stakeholderov.

Scrum Master

V divízii ABC je Scrum master zodpovedný za propagáciu a podporu Scrumu, pomáha každému porozumieť teórii Scrumu, jeho praktikám, pravidlám a hodnotám. Je dôležitým členom v zabehnutých tímoch a v nových je priam nevyhnutný. Aj keď tím neriadi, pomáha mu v dosiahnutí stanovených cieľov, s dodržiavaním

procesov a riešením prípadných problémov. Problémy môžu vzniknúť priamo v rámci tímu, ako napríklad komunikácia medzi jednotlivými členmi, prípadne nezhody medzi členmi vývojového tímu a Product ownera počas plánovania. Problémy externého charakteru môžu ovplyvniť alebo blokovať pokrok tímu počas šprintu. Scrum master pomáha tieto situácie a problémy identifikovať, analyzovať a následné riešenie vykoná buď sám, alebo poverí niekoho z tímu. V Scrumových tímoch je vysoká dôvera medzi jednotlivými členmi, a preto sa môžu na seba spoľahnúť, že každý svoju úlohu splní ako najlepšie vie.

Mimo tímu pomáha ľuďom porozumieť, aké interakcie s ním sú užitočné a aké majú negatívny dopad. Následne podporuje zmenu týchto interakcií tak, aby sa maximalizovala hodnota vytvorená Scrum tímom. Riadi všetky Scrum ceremónie a schôdzky so zákazníkom, kde podľa potreby pôsobí ako moderátor alebo zapisovateľ. Má na starosti plánovanie týchto stretnutí, stará sa, aby boli produktívne a spĺňali svoj účel. Scrum master usmerňuje členov tímu na správnu cestu, pokiaľ si nie sú istí v niečom súvisiacom so Scrumom.

Product ownerovi pomáha s lepším porozumením a komunikáciou hodnoty. Konzultuje s ním usporiadanie Backlogu a hľadá efektívne techniky na jeho manažovanie. Radí pri plánovaní práce na ďalší šprint vzhľadom na jej množstvo, náročnosť a rýchlosť práce tímu.

Podporuje vývojový tím pri samo organizovaní pozorovaním jeho postupov a následným navrhovaním zmien pre zvýšenie produktivity. Pomáha pochopiť definíciu „Hotovo“ všetkým členom - hotovo pre vývojára môže znamenať, že niečo implementoval, ale aby to bolo naozaj hotové, musí to byť aj otestované. Stará sa aj o rozvoj soft skillov jednotlivých členov tímu formou interných školení.

3.1.2 Scrum artefakty

Scrum artefakty poskytujú kľúčové informácie Scrum tímu a stakeholderom ohľadom vyvíjaného produktu, plánovaných aktivít a aktivitách, ktoré už boli dokončené.

Product Backlog

Vytvorenie produktového Backlogu má na starosti hlavne Product owner. Po úvodných stretnutiach so zákazníkom prevedie jeho požiadavky do jednotlivých úloh.

S rozpadnutím projektu a požiadaviek na menšie úlohy mu môžu pomôcť aj členovia vývojového tímu, prípadne Scrum master. Avšak jeho zostavenie je stále zodpovednosťou Product ownera. Product Backlog obsahuje všetky úlohy, ktoré sa musia splniť, aby bol produkt dokončený. Po začatí projektu sa môžu úlohy pridávať alebo odoberať v závislosti na zmeny v požiadavkách stakeholderov.

Sprint Backlog

Vytvorenie sprint Backlogu prebieha na plánovacom meetingu, súčasťou ktorého je celý Scrum tím. Vyberú sa úlohy z produkt Backlogu, ktoré majú byť splnené v rámci nasledujúceho šprintu. Po začatí šprintu môže už len tím meniť sprint Backlog. Developeri pridávajú nové úlohy väčšinou kvôli výskytu nových požiadaviek alebo novým zisteniam počas práce na úlohách v šprinte.

User story a story pointy

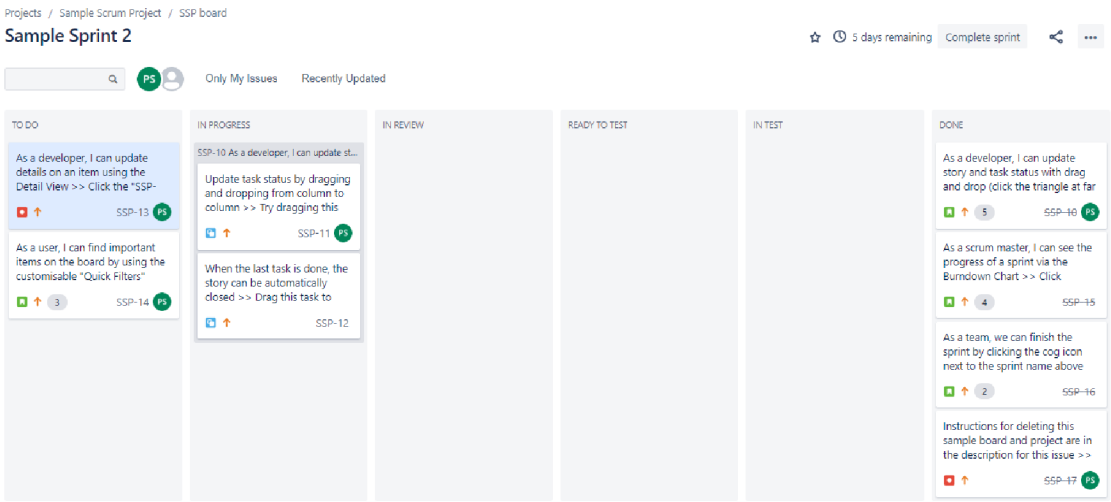
Namiesto použitia bežného popisu softwarových požiadaviek sa použijú user stories. Tie dávajú koncového užívateľa do centra diania, sú napísané v netechnickom jazyku a slúžia na poskytnutie kontextu pre vývojový tím. Po jej prečítaní by malo byť tímu jasné, prečo vytvárajú to, čo vytvárajú a akú to prináša hodnotu koncovému užívateľovi, či stakeholderovi. User story má zvyčajne nasledovný tvar: Ja ako <typ užívateľa>, chcem <nejakáfunkcionalita>, pretože <nejaký dôvod>. Ako admin, chcem dostávať pravidelný email o všetkých neaktívnych užívateľoch, aby som ich nemusel hľadať manuálne v databázi.

User story popisuje funkcionalitu z pohľadu osoby (napríklad admina), to akým spôsobom sa implementuje, vymyslí už vývojový tím. Zároveň musí odhadnúť pracnosť a zaznačiť ju ku každej úlohe. Namiesto klasického časového odhadu sa použijú tzv. story pointy. Použijeme čísla fibonacciho postupnosti: 0, 1, 2, 3, 5, 8, 13, 21, atď. Toto číslo reprezentuje náročnosť práce na danej úlohe. Výhodou použitia je relatívne ohodnotenie úloh, na ktorom sa podieľa celý vývojový tím. Relatívny odhad odstraňuje emocionálne pripútanie, ktoré môžu ľudia mať napríklad k dátumom. Dátum ukončenia celého projektu Product owner vie, ale nie je odzrkadlený priamo v jednotlivých úlohách. Pri riešení úlohy človek nepracuje pod časovým tlakom, ktorý by mohol byť spôsobený časovým odhadom. Keď si tím zvykne na tento spôsob odhadovania a na relatívnu hodnotu jedného story pointu, bude odhadovanie rýchlejšie bez dlhších

diskusií. Keďže sa odhadovania zúčastňuje celý tím (Product owner s vývojovým tímom), je v odhade zahrnutý názor všetkých strán, ktoré budú na danej úlohe pracovať.

Scrum tabula

Scrum tabula je virtuálna alebo fyzická tabula, ktorá vizuálne znázorňuje pokrok na úlohách v šprinte v reálnom čase. Obsahuje šesť stĺpcov, z ktorých každý odzrkadľuje stav úlohy, v ktorom sa nachádza: To do, In Progress, In Review, Ready to test, In test, Done. Začiatkom šprintu sú všetky naplánované úlohy pre daný šprint v stĺpčeku To do. Tieto úlohy boli navrhnuté Produkt ownerom na základe ich priorít a odsúhlasené tímom v rámci plánovania šprintu. Vývojári si jednotlivé úlohy, označované aj ako tickety, postupne priradia na seba a posunú o stĺpček doprava - do In progress. Tento priebeh a Scrum tabula je zobrazená na obrázku č. 16.



Obrázok č. 16: Scrum tabuľa v programe Jira (vlastné spracovanie podľa voľnej verzie [15])

Takýmto spôsobom sa presúva úloha po tabuli až sa dostane do posledného stĺpca Done. To znamená, že úloha prešla implementáciou a testovaním. V stĺpčeku Done sa nachádzajú všetky dokončené úlohy v danom šprinte. Okrem práce na novej funkcionalite sa počas šprintu pracuje aj na odstránení nájdených chýb, bežných úlohách a odstránení technického dlhu, ktorý postupom času vzniká.

3.1.3 Priebeh šprintu a meetingov

So zavedením agilnej metódy Scrum sa zavedú aj nové meetingy, ktoré k nej patria. V nasledujúcich podkapitolách bude popísaná ich presná náplň, čas konania v rámci šprintov a ich kľúčové výstupy. Taktiež bude opísaný priebeh šprintu v divízii.

Šprint

Každý šprint bude prebiehať dva týždne, teda desať pracovných dní. Je možné ho v súlade so Scrum pravidlami predĺžiť na tri až štyri týždne, avšak štyri týždne sú už veľmi dlhá doba pre efektívne reagovanie na zmeny. Pred začiatkom každého projektu, a teda aj pred jeho prvým plánovaním prebehne Backlog refinement meeting, kde sa ohodnotia jednotlivé úlohy story pointami minimálne pre prvý šprint. Je vhodné ohodnotiť o niečo viac úloh, keby bolo jednu z nich potrebné pridať do prebiehajúceho šprintu. Zvyšné úlohy sa odhadnú počas rovnakého meetingu, ktorý bude pravidelne prebiehať ako súčasť šprintov.

Pred začiatkom každého nového šprintu prebehne v piatok retrospektíva práce končiaceho šprintu a plánovanie pre nový šprint. Týmto spôsobom bude možné poznatky získané z retrospektívy plynulo zakomponovať pri nasledujúcom plánovaní bez väčšieho časového oneskorenia. Zabráni sa tak strate dôležitých informácií pre zlepšenie celkového procesu. Šprint začne v pondelok, kedy developeri začnú implementovať jednotlivé úlohy a QA inžinieri vykonajú regresné testovanie. Ďalšie dni začnú pripravovať test plány k jednotlivým čiastkovým úlohám, čím sú nútení sa zamyslieť nad vyvíjanou funkcionalitou skôr, ako ju začnú testovať.

V priebehu druhého dňa začnú QA inžinieri pripravovať automatizované testy pre úlohy, ktoré aktuálne developeri implementujú. V ideálnom prípade budú testy pripravené skôr alebo zároveň s ukončením vývoja funkcionalít, aby sa daný lístok kvôli tomu nezdržal. Nastanú aj situácie, kedy sa začne na automatizovaných testoch pracovať až po ukončení vývoja funkcionality z dôvodu zaneprázdnenosti QA inžinierov. V takýchto prípadoch si tieto testy napíšu developeri sami alebo vzájomne. Tieto testy majú pokryť implementovanú biznis logiku, jej správne scenáre s očakávaným správaním. Zároveň sa vyberú aj scenáre, ktoré budú testovať nejakú formu zlyhania - test bude odhaľovať chybu tak, že nespadne aj keď by mal. Tento druhý typ testov sa väčšinou pridá, až keď sa daná chyba vyskytla.

Jednou z pre rekvizít pre umiestnenie úlohy zo stĺpca In review do Ready to Test je úspešné prejdenie všetkých testov. To znamená unit testy, napísané developerom a automatizované testy pripravené členom vývojového tímu. Automatizované testy sú pravidelne spúšťané, aby priebežne odhaľovali zavedené chyby skôr, ako sa dostanú k zákazníkom, prípadne QA inžinierom na testovanie. Zároveň sú využívané ako regresné testovanie pri projektoch, ktoré sú dobre pokryté hneď od začiatku.

Keď je nová funkcionalita pripravená na testovanie, prebehne ešte pred samotným testovaním tzv. showcase. Ten prebieha za prítomnosti developera, ktorý danú úlohu implementoval, QA inžiniera, ktorý ju bude testovať a vo vybraných prípadoch je súčasťou aj Product owner. Developer prezentuje výsledok úlohy, čo ho núti nepreskočiť otestovanie si výsledku po sebe. Product owner hodnotí výsledok z pohľadu stakeholderov. Môže prebehnúť diskusia, počas ktorej môžu prísť na prípad použitia, ktorý im predtým nenapadol a nie je zahrnutý ani v popise úlohy. Ak sa nič závažného neobjaví, môže začať fáza testovania danej úlohy.

Každú stredu v šprinte je Backlog refinement meeting, na ktorom sa stretne celý tím a odhaduje pracnosť zvyšných úloh pre daný projekt. Ak tím pracuje na viacerých projektoch naraz, odhaduje úlohy pre projekt, ktorý to aktuálne vyžaduje. Každý deň sa koná daily stand-up, kde sa všetci členovia oboznámia s pokrokom ostatných členov a prezentujú svoj vlastný.

Backlog refinement

Backlog refinement meeting slúži na odhadovanie náročnosti úloh za prítomnosti celého Scrum tímu. Product owner ešte pred meetingom priorizuje backlog. Na meetingu predstaví jednotlivé user stories pripravené na ohodnotenie story pointami a prečíta poznačené produktové požiadavky. Úlohy sa začnú prechádzať postupne, tím začne klásť doplňujúce otázky pre lepšie pochopenie danej user story a požiadaviek. Po zodpovedaní všetkých otázok má celý tím lepšiu predstavu o práci, ktorá sa bude musieť vykonať a to od implementácie až po otestovanie.

Každý člen vývojového tímu si premyslí, aké číslo (odhad story pointov) by pre daný lístok(úlohu) dal. Následne sa použijú buď karty s číslami alebo prsty na rukách, tie sú tiež do čísla 8 použiteľné. Scrum master odráta od čísla tri do jeden a každý z vývojového tímu ukáže všetkým svoj odhad, či už formou karty alebo prstov.

Ak sa tím zhodol, dané číslo story pointov sa zapíše. To sa dlhodobou spoluprácou členov s rovnakým chápaním relatívneho rozsahu jednotlivých story pointov stáva pomerne často. Taktiež je časté, že sa nezhodnú v čísle o jeden stupeň, väčšinou medzi číslami 3, 5 a 8. Každý vidí riziko subjektívne, niekto väčšie, niekto menšie, a práve preto robí odhad celý tím. V prípade rozdielnych názorov sa začne krátka diskusia, kde si navzájom vysvetlia postup, akým dospeli k danému číslu. Ak by sa diskusia začala dostávať veľmi do detailu a začali by vznikať otázky, na ktoré by ani Product owner nemal odpovede, mohla by sa úloha odložiť bokom a Product owner by bol poverený zistením odpovedí. Diskusia, ktorá viedla k úspešnému vyriešeniu otázok, ktoré v nej vznikli, vedie k ďalšiemu hlasovaniu. Pri rozhodovaní sa medzi menšími čísla ako 2 a 3, alebo ak väčšina hlasovala rovnako, je možné po presvedčení posledného člena rovno zapísať story pointy bez dodatočného hlasovania.

Odhadovanie komplexnejších úloh môže mať vyšší výsledok, ako napríklad 13 či 20. Vyššie číslo značí vyššiu mieru neistoty, rizika, či nejasností. Je dobré znovu zvážiť, či sa daná úloha nedá rozdeliť na niekoľko menších, ktoré sa dajú lepšie odhadnúť. Odhadovanie celého projektu hneď pri začiatku môže mať zmysel, ak je relatívne malý. Pri väčších projektoch je vhodné odhadovať jednotlivé úlohy v predstihu šprint alebo dva, pretože sa môžu okolnosti a priority zmeniť. To môže spôsobiť, že niektoré funkcionality nebudú implementované vôbec alebo sa zmení ich popis natoľko, že bude potrebné vykonať ich odhad znovu. Pre potreby odhadnutia celého projektu z hľadiska plánovania je možné bez väčšej diskusie prejsť celý backlog projektu, ohodnotiť ho na základe väčšiny a keď príde na plánovanie daného ticketu do šprintu verifikovať odhad.

Tento meeting prebieha pravidelne každú stredu daného šprintu. V prípade divízie ABC však ešte žiadny šprint neprebehol, a preto je potrebné, aby tento meeting prebehol ešte pred začiatkom prvého. Vzhľadom na prvé odhady tímu ako celku bude pravdepodobne dochádzať k nadhodnocovaniu alebo podhodnocovaniu. To sa prejaví na priemernom počte dodaných story pointov po niekoľkých šprintoch. Postupom času sa táto hodnota ustáli a bude podľa nej možné plánovať. Keďže sa v šprintoch nenachádza len funkcionalita, na ktorej sa pracuje, je dobré odhadovať všetko, čo sa v šprinte nachádza. Tým sa zviditeľnia prípadné problémy, ak by tím nedodával, čo

si naplánoval. Zároveň je vhodné používať predchádzajúce ohodnotené tickety ako referenčné pre budúce odhady.

Sprint planning

Stretnutie zamerané na plánovanie práce do nasledujúceho šprintu sa opakuje pravidelne každý piatok na konci predchádzajúceho šprintu. Predpokladom k dobrému a rýchlemu plánovaniu je sprioritizovaný backlog s priradenými odhadmi. Plánovania sa účastní celý Scrum tím s tým, že je možné pripravený backlog diskutovať v menšom počte ešte tesne pred týmto meetingom. Takejto diskusie je zvyčajne účastníkom Product owner, Scrum master a jeden z developerov. Môže to urýchliť priebeh planningu s celým tímom.

Začiatkom stretnutia predstaví Product owner svoju predstavu pre ďalší šprint. Mala by to byť kombinácia novej funkcionality, opravenie chýb, ktoré sa neodhalili počas vývoja a odstraňovanie technického dlhu, ktorý spôsobuje alebo môže spôsobiť problémy s implementáciou budúcej funkcionality. Každý tím a firma si musí nájsť svoj ideálny pomer, ja navrhujem percentuálny pomer 65:20:15 pre funkcionalitu, odstránenie chýb a odstránenie technického dlhu. Je potrebné každý planning tento pomer sledovať a ak sa zistí, že sa tento pomer pre ďalší šprint nedodrží, je potrebné zistiť príčinu. Môže byť vyvíjaný tlak zo strany Product ownera na väčší vývoj nových funkcií, v lepšom prípade nie je veľa technického dlhu a chýb na odstraňovanie. Pri plánovaní sa vývojový tím zaväzuje, že naplánované lístočky dodá, a preto je potrebné dôjsť ku kompromisu. Product owner stanový cieľ šprintu, čo zvyčajne znamená, že ak by mala byť dodaná len jedna vec, má to byť daný cieľ.

Môžu a pravdepodobne nastanú situácie, kedy tím nedodá všetko, k čomu sa zaviazal. Určité veci, čo boli naplánované, sú už rozrobené a bola by škoda, ak by sa nedokončili. Pri ďalšom plánovaní sú teda ako prvé pridané do nasledujúceho šprintu. Je možné niektoré tickety odobrať, napríklad, ak sa práca na nich ešte nezačala a je potrebné dodať niečo prioritnejšie. Každým šprintom tím dodáva nejaký počet story pointov. Keďže sa odhaduje ideálne všetko, je jednoduché plánovať pre ďalší šprint. V prípade, že by tím nedodával plánované tickety z dôvodu externého vplyvu na jeho prácu, je vhodné uvažovať na vyčlenení dočasného bufferu - počtu story pointov, ktoré sa nenaplánujú na nič konkrétne, pretože sa môže objaviť niečo externého.

Product owner využíva okrem Jiri aj webovú aplikáciu Aha!. V nej upravuje očakávané dodanie jednotlivých funkcionalít pri začiatku projektu a pri každej zmene, ktorá môže nastať po konci každého šprintu. Zároveň zaznačuje míľniky ako dokončenie minimálneho životaschopného produktu, či iné fázy vývoja. Duplicita týchto údajov v dvoch webových aplikáciách nastáva kvôli manažmentu divízie a manažmentu firmy.

V prípade divízie ABC sa bude konať prvé plánovanie v čase, keď žiadny šprint ešte neprebehol. Musí sa uskutočniť až po prvom backlog refinement meetingu, aby boli už nejaké user stories odhadnuté. Nasledujúce stretnutia budú každý druhý piatok na konci šprintu a budú prebiehať tak, ako bolo v tejto časti opísané.

Daily Stand-up

Daily stand-up je meeting, ktorý sa opakuje každý deň až na posledný deň v šprinte. Účasní sa ho Scrum master, Product owner a celý vývojový tím. Tieto stretnutia sa konajú o jedenástej v miestnosti s televízorom. Ten je potrebný na zdieľanie obrazu z notebooku, keďže Scrumová tabula je vo virtuálnej forme v programe Jira. Scrum master vedie stretnutie, prechádza jednotlivé položky z prava do ľava. Zvykom je, že sa prechádza po jednotlivých členoch tímu a tí hovoria na čom pracovali, na čom budú pracovať a čo ich prípadne brzdí. Pri prechádzaní položiek z prava do ľava, čiže sa začne tými čo sú najbližšie k dokončeniu, sa tiež dostane rad na každého, kto má aktuálne niečo rozpracované. Tí, ktorí nie sú priradení na žiadnom lístočku na Scrum tabuli hovoria až po prejdení všetkých lístočkov. Tento spôsob vedenia stand-upov zabezpečuje, že sa žiadny lístoček nevynechá. Lístočky, na ktorých sa ešte nezačalo pracovať sa môžu v závislosti na dni v šprinte prebrať tiež, Product ownera môže zaujímať, či sa stihnú dokončiť alebo aspoň rozpracovať.

Ak je niekto z členov tímu niečím blokovaný priebežne to už s niekým rieši, ale stand-up je to miesto, kde by o tom mal informovať aj zvyšok tímu. Je to meeting určený na synchronizáciu, aby každý vedel, ako daný šprint postupuje. Zároveň môžu na stand-upe zistiť, že si vedia jednotliví členovia navzájom pomôcť, niekto z tímu môže mať viac kľúčových informácií pre riešenie problému. Prvý deň šprintu sa na stand-upe diskutuje rozpracovaná práca, prípadné chyby zistené automatizovanými testami cez víkend alebo počas regresného testovania. Scrum master dohliada na to, aby sa nezachádzalo príliš do detailu pri jednotlivých diskusiách. Product owner

dostáva tiež slovo, aby informoval tím o tom, ako sa projekt vyvíja z produktovej strany, či mal zákazník nejaké nové požiadavky alebo či zistil odpovede na otázky, ktoré tím vzniesol na niektoré tickety pri backlog refinement meetingu.

Bežná dĺžka tohoto stretnutia je 15 minút a nemalo by byť dlhšie. V prípade, že chce tím niečo prebrať podrobnejšie, je vhodné najprv ukončiť stand-up a následne zostanú v meetingovej miestnosti len tí, ktorí tam potrebujú ostať. Situácia, keď je celý tím v jednej miestnosti sa môže využiť aj na showcase vytvorenej funkcionality pred všetkými. Viac očí viac vidí a v prípadoch komplexnejšej úlohy sa môžu takýmto spôsobom odhaliť skryté problémy rýchlejšie. Môže nastať aj opačná varianta, keď nebude mať nikto žiadny problém na diskutovanie, všetci budú mať na čom pracovať a meeting sa ukončí skôr.

V posledný deň šprintu stand-up nie je, pretože je v tom čase retrospektíva daného šprintu. To zároveň znamená, že je potrebné dokončiť všetku prácu deň predtým, do štvrtka poobede a v piatok už len doladiť posledné veci. Posledný deň šprintu sa všetky zmeny vykonané v aktuálnom šprinte presunú z vývojového prostredia (ďalej len DEV, z angl. *development*) na prostredie zabezpečenia kvality (ďalej len QA, z angl. *quality assurance*), na ktorom bude prebiehať regresia prvý deň nasledujúceho šprintu. Namiesto stand-upu prebehne o jedenástej retrospektíva šprintu, o tretej poobede sprint review meeting a o štvrtej sprint planning.

Sprint review

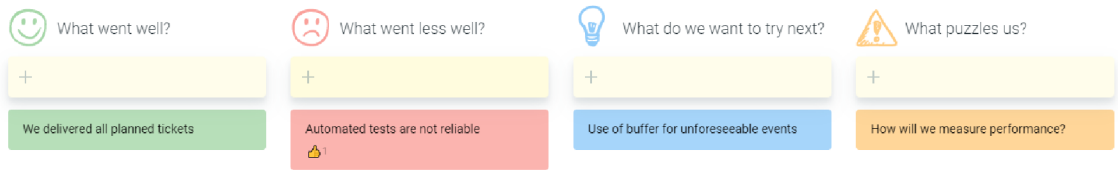
Koniec každého šprintu je v piatok o druhej. Keďže sú šprinty dlhé dva týždne, tak sa každý druhý týždeň koná sprint review. Tento meeting je špeciálny v tom, že sa zúčastní aj zákazník, prípadne ostatní stakeholderi. Vývojový tím predvedie implementovanú funkcionalitu v danom šprinte. Užívateľ má k dispozícii novú verziu hodinu predtým, takže si ju môže v prípade záujmu dopredu vyskúšať. Po ukončení prezentácie nových funkcií vznášajú stakeholderi a zákazník pripomienky, či doplňujúce otázky. Po ich zodpovedaní a uzavretí daného šprintu je priestor pre zmeny priorít lístkov v backlogu, pridanie nových alebo odobranie už existujúcich. Na základe zmien v tomto meetingu sa môže zmeniť poradie a priorita, ktorá ovplyvní aj plánovanie ďalšieho šprintu.

Scrum master vedie celý meeting, ktorý má dĺžku maximálne 45 minút. Miesto stretnutia je priamo fyzicky v divízii alebo formou video hovoru a zdieľaním plochy. Je potrebné na takýto vzdialený spôsob práce pripraviť aj klienta, ak na to nie je zvyknutý. Zároveň sa od neho čaká aktívna účasť a komunikácia a to nie len počas daného meetingu.

Sprint retrospective

Každý druhý piatok na konci šprintu nahradí retrospektíva stand-up meeting. Scrum master toto stretnutie vedie a ostatní členovia Scrum tímu sú aktívnymi účastníkmi. Na tomto meetingu sa zozbierajú myšlienky a názory jednotlivých členov na končiaci šprint.

Stretnutie sa koná v miestnosti s tabulou, rozdajú sa lepiace lístky všetkým zúčastneným. Podľa aktuálnej formy retrospektívy má každý päť minút na spísanie myšlienok na lístky- jednu myšlienku na jeden lístok. Každý člen tímu prejde po jednom k tabuli, rozmiestni svoje lístočky na tabulu a každý v krátkosti predstaví. Môžu nastať situácie, kedy nie je možné stretnutie všetkých fyzicky v jednej miestnosti a je potrebné použiť online nástroje ako napríklad TeamRetro [21]. Na obrázku č.17 je zobrazené, ako môže takáto retrospektíva vyzerať online.



Obrázok č. 17: Retrospektíva vo webovej aplikácií TeamRetro (vlastné spracovanie podľa voľnej verzie [21])

Vybraný formát retrospektívy je vhodný pre prvé retrospektívy, keďže vytvára jasný priestor pre návrhy na zlepšenie a vyjadrenie nejasností. Každý formát nejakým spôsobom vymedzuje priestor pre udalosti čo sa podarili a tie, čo sa nepodarili. Scrum master vyberá variantu, ktorá sa bude pri retrospektíve používať.

Po doplnení lístkov od každého zúčastneného určí Scrum master počet hlasov na osobu. Zároveň sa lístky s rovnakým významom spoja do jednej skupiny, aby sa hlasy nerozdelili medzi rovnakú tému, ale boli pokope. Zúčastnené osoby následne priradia svoje hlasy k jednotlivým lístočkom na základe toho, o čom by radi diskutovali. Hlasy

sa pri lístkoch sčítajú a zoradia sa jednotlivé témy na následnú diskusiu od najväčšieho počtu po najmenší. Ak by mali nejaké témy rovnaký počet hlasov, tím si určí, čo je preňho dôležitejšie. Na stretnutie je vyčlenených 50 minút, tento čas si môže tím v prípade potreby ľubovoľne upraviť. Je samozrejmé, že sa nepodarí vždy prejsť všetky témy, za ktoré sa zahlasuje. Tie ktoré nebudú zodpovedané, ale autorovi danej témy záleží na jej vyriešení môže vyhľadať pomoc Scrum mastra mimo retrospektívy.

Začiatok diskusie začne predstavením témy autorom lístočku. Scrum master prevezme rolu zapisovateľa a bude sa snažiť z diskusie vyberať identifikáciu problému a možné riešenia. Tieto poznámky píše priamo na tabulu, alebo v prípade online stretnutia u seba lokálne a následne ich za zdieľa celému tímu. Z poznámok sa vytvorí identifikácia problému, návrhy na možné riešenie a čo je najdôležitejšie, vyberie sa osoba, ktorá má zmenu alebo vyriešenie problému viesť. Tým sa zabezpečí, že na tom niekto bude pracovať a bude informovať tím o pokroku. Keďže je tento meeting ešte pred plánovaním, je možné zahrnúť výstupy z retrospektívy priamo do nasledujúceho šprintu.

3.2 Použitie softwarových nástrojov

V rámci riadenia projektov vedených agilnou metódou, je vhodné použiť nástroje vyvinuté pre ich podporu. Divízia ABC už väčšinu nástrojov používa, takže nebude musieť prechádzať veľkým preškolením. Hlavný z nich je Jira, ktorý budú využívať všetci členovia Scrum tímu. Druhým je Aha!, ktorý bude využívať prevažne Product owner. TeamRetro bude využívané v prípadoch, kedy bude väčší počet členov dostupných len vzdialene.

3.2.1 Jira

Webovú aplikáciu Jira už divízia používa aj napriek tomu, že ju nevyužívala pre agilnú metodiku vývoja. Je to však zavedený nástroj pre celú firmu, a tak ho prebrali. Vo firme ho mimo iné využívajú na vytváranie chýb, ktoré prichádzajú od zákazníkov a koncových užívateľov. Pre divíziu to teda znamená, že pri prechode na agilný vývoj nepotrebuje investovať do nového nástroja, pretože už používa jeden z najpopulárnejších.

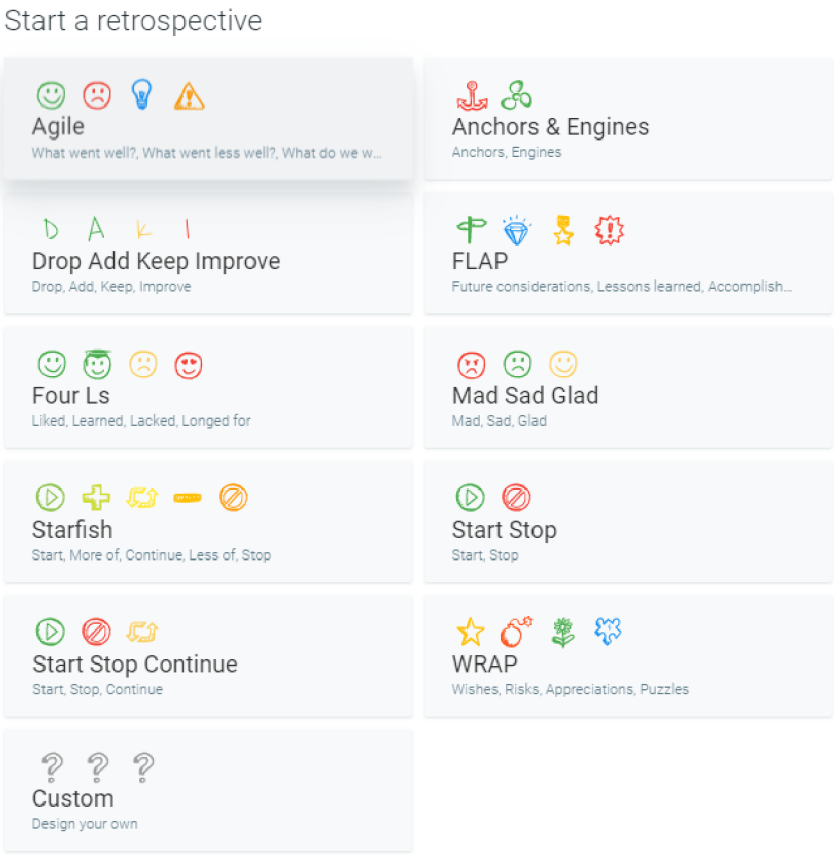
Jira obsahuje všetky artefakty opísané v rámci tejto kapitoly, od produktového backlogu až po burndown grafy. Práve kvôli všetkým funkciám prispôsobeným agilnému vývoju je tento nástroj tak populárny. Hlavné časti, ktoré sa pri vývoji a riadení použijú, sú už spomínané artefakty. Scrum tabulu bude využívať celý Scrum tím, každý deň. Počas meetingov im bude slúžiť ako oporný bod pri diskusiách, developerom pre vizualizáciu voľných ticketov na prácu, QA inžinierom zobrazí tickety pripravené na otestovanie, Product ownerom priebeh a pokrok šprintu. Zároveň ponúka rôzne nástroje na meranie pokroku, plánovanie projektu, delenie na verzie ci zhlukovanie jednotlivých úloh pod rovnakým označením.

3.2.2 Aha!

Webový nástroj Aha! je používaný projektovými manažérmi v rámci spoločnosti. Vedú sa ním projekty a portfóliá projektov tradičným štýlom projektového manažmentu. V divízii je tento nástroj využívaný manažérom a bude využívaný Product ownerom. Ten bude mať na zodpovednosť aktualizovanie pokroku projektov Scrum tímu, ich plánované vydanie na trh alebo odovzdanie zákazníkovi. Zároveň vytvorí míľniky, ktoré bude priebežne kontrolovať.

3.2.3 TeamRetro

Tento softwarový nástroj bude používaný na vedenie retrospektívy vzdialene. Scrum master má na výber z desiatich predpripravených variant, ako môže plocha retrospektívy vyzerať. Zároveň má možnosť vytvoriť si vlastnú, zobrazenie výberu je na obrázku č. 20. Ak by sa tento meeting odohrával fyzicky v zasadacej miestnosti, Scrum master by dané rozdelenie nakreslil na tabulu a každému zúčastnenému by dal lepiace lístočky. Priebeh retrospektívy je vysvetlený v časti meetingov. Zároveň je možné použiť tento nástroj aj na kontrolu stavu tímu. V tomto prípade je možné použiť aplikáciu dopredu pred meetingom a na ňom len diskutovať o výsledkoch.



Obrázok č. 18: Všetky možnosti vytvorenia retrospektívy vo webovej aplikácii TeamRetro (vlastné spracovanie podľa voľnej verzie [21])

3.3 Nábor nových pracovníkov

Prechodom na agilnú metodiku Scrum sa vytvoria nové pracovné pozície, ktoré predtým divízia ABC nepotrebovala. Je ňou pozícia Scrum mastra a Product ownera. V aktuálnom stave je v roli Product ownera manažér divízie, ktorý je za všetky projekty zodpovedný. O túto rolu sa delí zároveň s vedúcimi tímov, ktorí zodpovedajú za ich implementovanú časť projektu. V Scrum tíme je jeden Product owner zodpovedný za projekt, prípadne projekty. Preto je potrebné obsadiť túto pozíciu jedným skúseným človekom a nemať ju rozdelenú po viacerých ľuďoch. Vhodného kandidáta môže divízia vybrať zo svojich radov, prípadne zo zamestnancov spoločnosti. Ak by sa jej to nepodarilo, je potrebné najať vhodného kandidáta mimo nej.

Medzi hlavné požiadavky na kandidáta patrí komunikácia a vyjednávanie. Uplatní ich v rámci tímu, so zákazníkom a s ostatnými stakeholdermi. Musí byť schopný

vykonávať produktové rozhodnutia, pričom bude brať ohľad na všetkých stakeholderov. Mal by rozumieť životnému cyklu vývoja softwaru a analyticky myslieť, keďže bude konzultovať so zákazníkom požiadavky na produkt.

Divízia potrebuje obsadiť aj pozíciu Scrum master. V tomto prípade musí daného kandidáta najať, pretože táto rola a ani jej podobná v rámci spoločnosti neexistuje. Scrum master by mal byť skúsený s niekoľkoročnou praxou, keďže bude zavádzať a dohliadať na vykonávanie Scrum procesov. Bude navrhovať spôsoby pre zlepšenie procesov a pomáhať novému Scrum tímu s prechodom na agilný spôsob vývoja a vedenia projektov. Bude radiť a vzdelávať Product ownera v oblasti vylepšovania a spravovania backlogu. Tímu pomôže odstraňovaním prekážok, ktoré sa budú v rámci prechodu objavovať.

Posledný kandidát, ktorého by som zvážil najať je na pozíciu QA inžinier. V novom Scrum tíme budú na tejto pozícii dve osoby. Jedna z nich musí mať predchádzajúcu skúsenosť s prácou v Scrum tíme, aby sa predpokladaný prínos zavedenia kvality prejavil. Náplň práce v Scrum tíme sa bude čiastočne líšiť od aktuálnej práce testera. Od QA inžiniera sa očakáva implementácia ním automatizovaných testov, spolupráca na spisovaní produktových požiadaviek a vytváranie test plánov v rámci technického dizajnu. Na túto pozíciu je možné vybrať aj pracovníka z radov divízie, avšak je potreba nájsť ešte jedného.

3.4 Metriky na meranie úspechu zavedenia agilného prístupu

Po zavedení novej metodiky je dôležité zistiť, ako daný tím napreduje. Je potrebné sledovať efektivitu tímu, či sa mu darí dodávať všetko, čo si naplánuje. Zároveň je vhodné zistiť, ako sa jednotlivým členom tímu pracuje, čo si o novom spôsobe myslia, čo by zlepšili alebo zmenili. Pre tieto potreby môžeme sledovať niekoľko metrík.

3.4.1 Kontrola stavu tímu

Kontrolu stavu tímu je vhodné realizovať na konci kvartálu, respektíve po dlhšom časovom úseku, keďže jeden šprint trvá minimálne dva týždne. Vytvorí sa stretnutie, ktorého sa zúčastní celý Scrum tím. Malo by mať maximálne hodinu, je však možné, že sa niekedy predĺži alebo skráti. Cieľom tohoto stretnutia je zistiť odpovede

na jedenásť otázok, respektíve oblastí. Následne sa zozbiera spätná väzba, na základe ktorej je možné odhaliť na prvý pohľad skryté problémy. Oblasti, na ktoré sa získavajú odpovede sú:

- Doručovanie hodnoty stakeholderom
- Jednoduchosť zverejnenia (novej verzii)
- Zábava
- Zdravie kódu
- Vzdelávanie
- Misia
- Figúrky alebo hráči
- Rýchlosť
- Vhodný proces
- Podpora (externá, nie v rámci tímu)
- Tímová spolupráca

V prípade fyzického stretnutia sa používajú kartičky so semaforom, či farbami semaforu a znázorňujú tri stupne - kladnú, neutrálnu a zápornú. Zároveň je užitočné zaznačiť aj trend, či sa podľa jednotlivých členov daná situácia zlepšuje, nemení alebo zhoršuje. Podobne ako pri šprint retrospektíve je tím vyzvaný Scrum masterom, aby si rozmyslel svoju odpoveď na danú oblasť. Celý tím následne zdvihne kartičku semaforu s farbou, ktorá vyjadruje jeho odpoveď. Pre tú istú oblasť si znovu každý rozmyslí, aký je podľa neho trend a po vyzvaní Scrum mastrom znovu zdvihne príslušnú kartičku semaforu. Následne sa začne Scrum master pýtať na dôvody, ak sa odpovede členov tímu príliš líšili alebo ak boli príliš negatívne. Odpovede si zapíše a uloží ich ako komentáre k daným oblastiam a výsledkom.

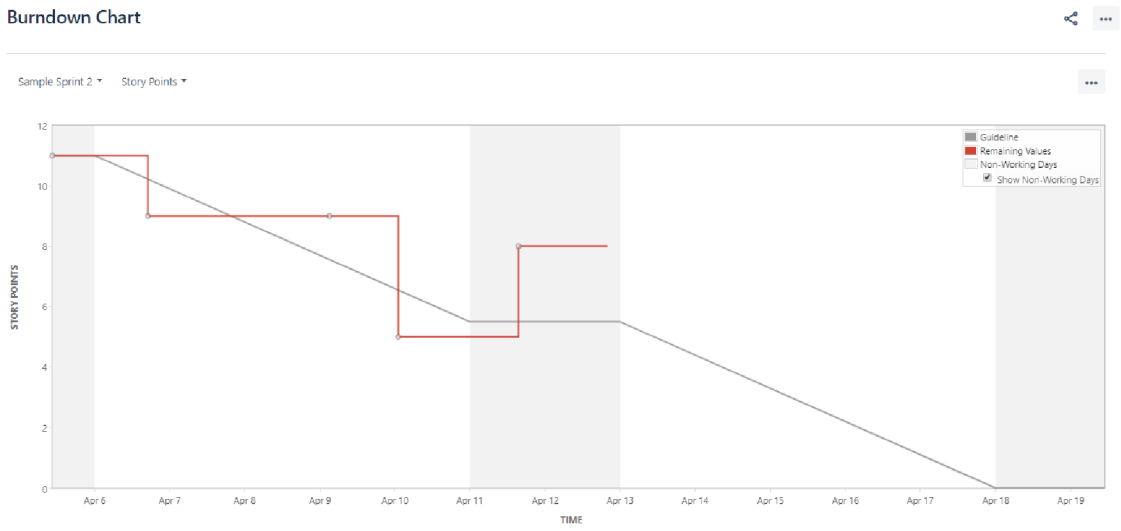
Túto spätnú väzbu si následne môžu prezrieť kompetentné osoby, ktoré môžu jej negatívne časti zlepšiť. Zároveň tak majú prehľad o tom, čo si daný tím myslí a môžu vykonať konkrétne kroky pre odstránenie nedostatkov alebo optimalizáciu procesov. Použitie tohto postupu nie je obmedzené len na Scrum tímy, je možné to zaviesť pre všetky tímy, avšak v Scrum tíme sa kladie väčší dôraz na kooperáciu a samo organizáciu.

Obrázok č. 19: Kontrola stavu tímu vo webovej aplikácií TeamRetro (vlastné spracovanie podľa voľnej verzie [21])

Pre situácie, kedy nie je väčšina tímu fyzicky v kancelárii, je možné robiť kontrolu stavu tímu pomocou video hovoru. Ten môže byť doplnený o rovnaký online nástroj ako v prípade vzdialenej šprint retrospektívy. Webová aplikácia TeamRetro ponúka možnosť Squad Health Check, ako je možné vidieť na obrázku č. 21. Umožňuje zaznamenávanie odpovedí jednotlivých členov tímu až na výsledný trend. Ten je potrebné dopísať ručne ako komentár pre jednotlivé oblasti.

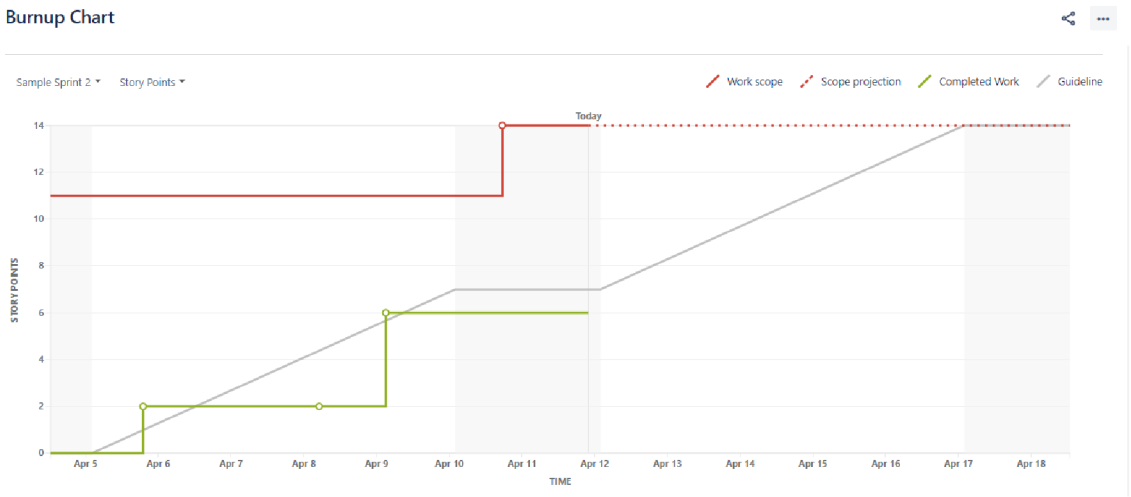
3.4.2 Burndown a burnup grafy

Po začatí šprintu je možné sledovať rýchlosť tímu voči očakávaniam na burndown grafe zobrazenom na obrázku č. 22 a burnup grafe zobrazenom na obrázku č. 23. Na oboch grafoch sú zaznamenané body, keď bol lístok (úloha, na Scrum tabuli, angl. *ticket*) dokončený. Zároveň sa znázorňuje aj pridanie ďalších lístkov do šprintu až po jeho začatí alebo odstránenie už existujúcich. Na grafe je možné zobrazovať víkendy alebo len pracovný týždeň.



Obrázok č. 20: Burndown graf v programe Jira (vlastné spracovanie podľa voľnej verzie [15])

Prvý očividný rozdiel je, že pri burndown grafe klesá červená čiara z hora dole s každým uzavretým ticketom, ktorý má priradené story pointy. Zatiaľ čo pri burnup grafe je to opačne. Druhý a zásadnejší rozdiel je, že burnup graf zobrazuje zvlášť rozsah práce a dokončenú prácu voči predpokladanému vývoju. Vo vzorovom šprinte bol pridaný ticket až po začatí šprintu, pričom na burndown grafe nie je na prvý pohľad jasné, čo sa udialo, avšak pri burnup grafe áno.



Obrázok č. 21: Burnup graf v programe Jira (vlastné spracovanie podľa voľnej verzie [15])

3.4.3 Sledovanie tímovej velocity

Tak ako kontrola stavu tímu je dôležité sledovať aj ich pravidelnú, či priemernú rýchlosť dodávania story pointov za šprint. To znamená niekoľko šprintov po zavedení sledovať ako sa zastabilizuje počet story pointov dodávaných za šprint a následne sledovať odchýlky. Ak by nastávali väčšie zmeny, vyšší počet smerom hore alebo dole, znamená to zvyčajne aj nejakú zmenu v rámci tímu. Zo začiatku sa očakáva, že sa budú tieto čísla meniť, pretože tím si bude zvykať spolupracovať a odhadovať jednotlivé úlohy. Niektoré úlohy budú podhodnotené, niektoré opačne nadhodnotené. Všetci si budú musieť zvyknúť na novú definíciu "dokončeného", čo znamená nielen implementované, ale aj úspešne otestované. To znamená, že sa vývojárovi môže niekoľkokrát za šprint otočiť ticket, kým sa úspešne dokončí. Ak by pracoval vo vodopádovej metóde, prišlo by sa na chybu až po niekoľkých týždňoch možno mesiacoch, keď by sa daná časť dostala na testovanie. Túto rýchlosť tímu je možné sledovať priamo v Jire ako je vidieť na obrázku č. 24, alebo si vytvoriť vlastný graf na základe zozbieraných hodnôt.

Obrázok č. 22: Kontrola rýchlosti dodávania tímu za jednotlivé šprinty vo webovej aplikácií Jira (vlastné spracovanie podľa voľnej verzie [15])

Keď sa priemerná rýchlosť stabilizuje, je možné plánovať efektívnejšie budúce šprinty. Za predpokladu, že tímový odhad jednotlivých úloh sa spresní a bude sa odhadovať dopredu značná časť projektu, je možné naplánovať niekoľko budúcich šprintov. Ak Scrum tím začal pracovať aj na ďalších projektoch a nemal by len jeden,

je hodnotenie story pointami stále vhodné. Treba si však dať pozor, aby sa nemenil tímu príliš často kontext v rámci jedného šprintu. Ak budú musieť developeri pracovať na troch projektoch naraz a kontrolovať po sebe navzájom kód v rôznych projektoch, na ktorých nepracujú naraz, môže sa ich produktivita výrazne znížiť.

3.5 Ekonomické zhodnotenie návrhu

Zmena vedenia projektov vytvorením Scrumového tímu so sebou prinesie aj nové náklady. Tie budú vo forme mzdy pre dvoch až troch nových zamestnancov, ktorí budú členmi Scrum tímu. Predpokladaná mzda na pracovníkov podľa ich rolí a požadovaných skúseností je zobrazená v tabuľke č.1.

Názov pozície	Hrubá mzda mesačne [Kč]	Super hrubá mzda (náklady divízie) mesačne [Kč]	Ročné náklady [Kč]
Scrum master	50 000	66 900	802 800
Product owner	50 000	66 900	802 800
QA inžinier	50 000	66 900	802 800

Tabuľka č. 1: Minimálne náklady na mzdy za mesiac a rok (vlastné spracovanie)

Každý nový zamestnanec bude firmu z hľadiska mzdy stáť minimálne 66 900 Kč. Pre získanie skúsenejšieho kandidáta sa môžu tieto náklady výrazne zvýšiť. Náklady na zaobstaranie nových nástrojov nebudú príliš vysoké, keďže firma už nevyhnutné nástroje využíva. Prijatím nových zamestnancov sa rozšíri počet užívateľov o troch pre nástroj Jira, čo vychádza 7$ mesačne na jedného. Nástroj Aha! sa rozšíri o jedného užívateľa, pri ktorom sa líšia náklady podľa spôsobu platenia.

Názov produktu	Jeden užívateľ mesačne [$]	Náklady mesačne celkovo [$]	Ročné náklady [$]
Jira	7	21	252

Tabuľka č. 2: Náklady na troch nových užívateľov v programe Jira (vlastné spracovanie)

Názov produktu	Mesačné na jedného užívateľa [$]	Mesačné náklady na 1 užívateľa pri ročnom predplatnom [$]	Ročné náklady celkom pri mesačnej platbe [$]	Ročné náklady celkom pri ročnom predplatnom [$]
Aha!	124	99	1 488	1 188

Tabuľka č. 3: Náklady na jedného nového užívateľa v programe Aha! (vlastné spracovanie)

Jediný nový potrebný nástroj je TeamRetro, ktorý sa bude využívať v prípadoch vzdialeného pracovania viacerých členov tímu. Jeho podmienky sú 25$ pre jeden tým na mesiac, alebo 250$ pri ročnom predplatnom.

Názov produktu	Mesačne za 1 tím [$]	Mesačne za 1 tím pri ročnom predplatnom [$]	Ročné náklady celkom pri mesačnej platbe [$]	Ročné náklady celkom pri ročnom predplatnom [$]
TeamRetro	25	20,83	300	250

Tabuľka č. 4: Náklady na používanie aplikácie TeamRetro pre jeden tím (vlastné spracovanie)

Celkové náklady pri využití výhodnejších ročných predplatných alebo len mesačných platieb sú rozpísané v tabuľke č. 5.

	Náklady za mesiac pri mesačnom platení [Kč]	Náklady za mesiac pri výhodnejšom ročnom predplatení [Kč]	Ročné náklady celkom pri mesačnej platbe [Kč]	Ročné náklady celkom pri ročnom predplatnom [Kč]
Scrum master	66 900	66 900	802 800	802 800
Product owner	66 900	66 900	802 800	802 800
QA inžinier	66 900	66 900	802 800	802 800
Jira	535,92	535,92	6 431,04	6 431,04
Aha!	3 164,48	2 526,48	37 973,76	30 317,76
Team Retro	638	531,58	7656	6 378,96
Spolu	205 038,4	204 293,98	2 460 460,8	2 451 527,76

Tabuľka č. 5: Celkové náklady na zavedenie predstaveného návrhu (vlastné spracovanie)

Keďže by spoločnosť ušetrila využitím ročných predplatných len okolo 9 tis. Kč, navrhujem platiť za nástroje mesačne. Týmto spôsobom môže kedykoľvek dané nástroje prestať využívať bez toho, aby za ne už zaplatila.

3.6 Prínosy návrhu

Návrh zavedenia agilnej metódy Scrum v rámci divízie prináša efektívnejšie a optimalizovanejšie riadenie a plánovanie projektov. Zachová sa aj klasický spôsob riadenia projektov a prechod na novú metodiku sa otestuje na jednom zostavenom Scrum tíme.

Zlepšenie zavedenia sa prejaví pri komunikácii a práci so zákazníkom. Odstráni sa problém s nepresnou špecifikáciou, ktorá by spôsobila značné problémy pri odovzdávaní produktu zákazníkovi. Zavedením šprintov sa bude každé dva týždne prezentovať pokrok a pridaná funkcionalita všetkým stakeholderom. Budú si vedieť pridané funkcie otestovať, vzniesť pripomienky, či zmeniť poradie plánovanej implementácie. Týmto spôsobom sa divízia vyhne situáciám, kde by bol zákazník nespokojný z hľadiska funkcionality výsledného produktu. Optimalizácia rýchlosti dodania výsledného produktu ušetrí divízii čas a financie, ktoré by boli vynaložené na prerábanie už dokončeného produktu.

Komunikácia medzi všetkými členmi tímu bude podporovaná Scrumovými meetingami. Každý člen bude vedieť, aké je jeho rola v rámci tímu, v akom štádiu sa projekt nachádza a aká funkcionalita sa pridá v ďalšej iterácii - šprinte. Napriek rozdielnym rolám je dodanie naplánovanej hodnoty stakeholderom zodpovednosťou celého tímu.

ZÁVER

Prvá časť diplomovej práce je dedikovaná teoretickému úvodu. V tejto časti boli vysvetlené základné pojmy a definície. Bola objasnená teória, ktorá bola potrebná pre porozumenie zvyšnej časti záverečnej práce. Boli popísané klasické a agilné metodiky vývoja softvérových projektov, role jednotlivých zamestnancov v Scrum tíme a priebeh jednotlivých meetingov.

Nasledujúca, analytická časť záverečnej práce pozostáva z predstavenia spoločnosti, hlavne jej softvérovej divízie, o ktorej táto práca pojednáva. Štruktúra divízie bola detailne popísaná. Proces riadenia vývoja projektov vodopádovým modelom bol analyzovaný a detailne popísaný, vrátane jeho nevýhod. Boli zachytené úlohy a kompetencie jednotlivých zamestnancov podieľajúcich sa na vývoji projektu. Taktiež boli predstavené využívané nástroje, ako napríklad Jira, Aha! či GitHub. Koniec kapitoly je venovaný celkovému zhrnutiu výsledkov analýzy divízie vybranej spoločnosti.

Návrhová časť vychádza z výstupu analytickej časti diplomovej práce. Na základe výsledku analýzy bol navrhnutý nový spôsob vedenia projektov pomocou agilnej metodiky Scrum. Metodika Scrum adresuje väčšinu nevýhod stroj metodiky vodopádového modelu využívanej divíziou ABC vybranej spoločnosti. Použitím agilnej metódy boli definované aj nové role a ich kompetencie. Bolo opísané vytvorenie Scrumového tímu a činnosti každého člena počas šprintu a meetingov. Účel nových meetingov bol vysvetlený aj s vykonávanými aktivitami počas nich. Následne boli opísané softvérové nástroje a metriky, ako napríklad TeamRetro, ktoré slúžia na monitorovanie zavedeného prístupu a stavu tímu.

V záverečnej časti práce bolo vypracované ekonomické zhodnotenie návrhu zmeny ako aj jeho prínosy pre divíziu ABC a tým pádom pre celú spoločnosť. Výhodou Scrum metódy je zvýšená flexibilita ohľadom zavedenia zmien v prebiehajúcich projektoch, ako aj zakomponovanie klienta do procesu vývoja prostredníctvom účasti na prezentačnom meetingu. Na druhú stranu zavedenie môže byť pre spoločnosť finančne náročné, keďže zavedenie novej Scrum agilnej metodiky na rok by spoločnosť stálo okolo 2,5 mil. Kč.

Z môjho uhľa pohľadu, by zavedenie navrhnutých zmien, bolo pre spoločnosť z dlhodobého hľadiska prínosné. Navrhnutá agilná metodika by znížila počet pripomienok po odovzdaní projektu zadávateľovi ako aj časové, ekonomické a personálne náklady divízie s nimi spojené.

ZOZNAM POUŽITEJ LITERATÚRY

[1] J. Carroll, *Effective project management*. Leamington Spa, Warwickshire, UK: In Easy Steps Ltd., 2012, ISBN 978-1840784466.

[2] J. Doležal, P. Máchal, B. Lacko, a Společnost pro projektové řízení, *Projektový management podle IPMA*. Praha: Grada, 2012, ISBN 978-80-247-4275-5.

[3] Z. Šochová a E. Kunce, *Agilní metody řízení projektů*. Brno: Computer Press, 2014, ISBN 978-80-251-4194-6.

[4] "What is a Milestone in Project Management? | TeamGantt". https://www.teamgantt.com/blog/the-how-and-why-of-using-milestones-in-your-project-plan (cit apr. 23, 2020).

[5] P. McGhee a P. McAliney, *Painless project management: a step-by-step guide for planning, executing, and managing projects*. Hoboken, N.J: John Wiley & Sons, 2007, ISBN 978-0-470-11721-7.

[6] L. LaPrad, "The Triple Constraints of Project Management | TeamGantt". https://www.teamgantt.com/blog/triple-constraint-project-management (cit apr. 23, 2020).

[7] I. Sommerville, *Softwarové inženýrství*. Brno: Computer Press, 2013, ISBN 978-80-251-3826-7.

[8] A. Buchalcevová, *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky*. Praha: Grada, 2005, ISBN 80-247-1075-7.

[9] "Manifesto for Agile Software Development". http://agilemanifesto.org/ (cit apr. 23, 2020).

[10] Paradigm, "What are the 8 Wastes in Lean?" https://www.visual-paradigm.com/tw/scrum/what-are-the-eight-lean-wastes/ (cit apr. 23, 2020).

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

[12] "What Is A Kanban Board? - The Fundamentals", *Digite*, aug. 22, 2016. https://www.digite.com/kanban/kanban-board/ (cit apr. 23, 2020).

[13] "Scrum Sprint | Product Management Framework | Infinity". https://startinfinity.com/product-management-framework/scrum-sprint (cit apr. 23, 2020).

[14] Paradigm, "What are Scrum Artifacts?" https://www.visual-paradigm.com/scrum/what-are-scrum-artifacts/ (cit apr. 23, 2020).

[15] Atlassian, "Jira | Issue & Project Tracking Software", *Atlassian*. https://www.atlassian.com/software/jira (cit apr. 24, 2020).

[16] D. Kelly, "Advantages and Disadvantages of the Scrum Project Management Methodology | Chron.com". https://smallbusiness.chron.com/advantages-

disadvantages-scrum-project-management-methodology-36099.html (cit apr. 23, 2020).

[17] Chandana, “Scrum Project Management Pros and Cons”. https://www.simplilearn.com/scrum-project-management-article (cit apr. 23, 2020).

[18] “Aha! — The World’s #1 Roadmap Software”. https://www.aha.io/ (cit apr. 23, 2020).

[19] “The Agile Testing Pyramid – Agile Coach Journal”. https://www.agilecoachjournal.com/2014-01-28/the-agile-testing-pyramid (cit apr. 23, 2020).

[20] “The One Page Test Plan | MoT”. https://www.ministryoftesting.com/dojo/lessons/the-one-page-test-plan (cit apr. 23, 2020).

[21] “TeamRetro - Online Retrospective and Team Health Check Tool”, *TeamRetro*. https://www.teamretro.com/ (cit apr. 23, 2020).

ZOZNAM POUŽITÝCH SKRATIEK A SYMBOLOV

iOS – je mobilný operačný systém od firmy Apple Inc.

QA – zabezpečenie kvality (z angl. *quality assurance*)

SaaS – Softvér ako služba (z angl. *Software as a Service*)

SWOT – Strengths, Weaknesses, Opportunities, Threats

ZOZNAM OBRAZKOV

ZOZNAM TABULIEK