VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ

FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY

INSTITUTE OF INFORMATICS

VYUŽITÍ AGILNÍCH METODIK PŘI ŘÍZENÍ VÝVOJE SOFTWARE

USAGE OF AGILE METHODOLOGIES IN SOFTWARE DEVELOPMENT MANAGEMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Simona Brejčáková

AUTHOR

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

SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Studentka:	**Simona Brejčáková**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
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 bakalářskou práci s názvem:

Využití agilních metodik při řízení vývoje software

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 bakalářské práce je využiti teoretických poznatků, nástrojů a metod agilního řízení při vývoji softwaru u vybrané firmy.

Základní literární prameny:

BECK, Kent. Test-driven development: by example. Boston: Addison-Wesley, 2003. ISBN 0321146530.

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

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

SMOLÍKOVÁ, Lenka. Projektové řízení: studijní text pro prezenční a kombinovanou formu studia. Brno: Akademické nakladatelství CERM, 2018. ISBN 978-80-214-5695-2.

ŠOCHOVÁ, Zuzana a Eduard KUNCE. Agilní metody řízení projektů. Brno: Computer Press, 2014. ISBN 978-80-251-4194-6.

Termín odevzdání bakalářské 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

Bakalárska práca sa zaoberá problematikou agilného vývoju softvéru. Teoretická časť je venovaná popisu tradičných a agilných metód, ich porovnaní. Následne sú podrobnejšie opísané agilné metodiky Scrum, Extrémne programovanie a agilné nástroje ako Kanban, Test Driven Development a Feature Driven Development. Ďalej je vykonaná analýza súčasného stavu spoločnosti, ktorá využíva niektoré prvky agilného riadenia. Na základe tejto analýzy sú navrhnuté riešenia nájdených problémov.

Abstract

The proposed bachelor thesis deals with agile software development. The theoretical part is devoted to the description of traditional and agile methodologies and their comparison. Agile methodologies such as Scrum, Extreme programming and agile tools such as Kanban, Test Driven Development and Feature Driven Development are described in more detail below. Furthermore, was carried out the current state analysis of the company, which used some tools of agile management. Based on analysis, are proposed the solutions to problems that were discovered.

Kľúčové slová

projektové riadenie, vývoj softvéru, agilné metodiky, Scrum

Key words

project management, software development, agile methodologies, Scrum

Bibliografická citace

BREJČÁKOVÁ, Simona. Využití agilních metodik při řízení vývoje software [online]. Brno, 2020 [cit. 2020-05-05]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/127006. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Lenka Smolíková.

Čestné prehlásenie

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

V Brně dne 10. května 2020

.....

podpis študenta

Poďakovanie

Rada by som na tomto mieste poďakovala Ing. Lenke Smolíkovej, Ph.D. za vedenie mojej bakalárskej práce. Veľmi si vážim jej ústretový prístup a cenné rady, ktoré mi pomohli pri písaní práce. Poďakovanie patrí aj spoločnosti MICo Vision s.r.o.

OBSAH

ÚVOD **10**

CIELE PRÁCE, METÓDY A POSTUP SPRACOVANIA **11**

1 TEORETICKÉ VÝCHODISKA PRÁCE **12**

1.1. Projekt 12
1.2 Vývoj softvéru 13
1.3 Tradičné metodiky 13
1.3.1 Vodopádový model 14
1.4 Tradičné metodiky vs. Agilné metodiky 14
1.5 Agilný vývoj 15
1.5.1 Manifest agilného vývoja 16
1.5.2 Robiť agilne vs. Byť agilný 17
1.6 Príklady agilných metodík 17
1.6.1 Extrémne programovanie (XP) 18
1.6.2 Scrum 21
1.7 Agilné nástroje 26
1.7.1 Kanban 27
1.7.2 Test Driven Development 29
1.7.3 Feature Driven Development 30

2 ANALÝZA SÚČASNÉHO STAVU **33**

2.1 Charakteristika podnikateľského subjektu 33
2.2 Organizačná štruktúra 33
2.3 Predmet podnikania 34
2.4 Nástroje 35
2.5 Technológie 36
2.5 Softvérový tím 36
2.5.1 Veľkosť tímu 36
2.5.2. Spolupráca 37
2.5.3 Vývojový proces 37
2.5.4 Prioritizovanie požiadavok 38

2.5.5 Dodávanie zákazky 38
2.5.6 Stand Up 39
2.5.7 Spísanie požiadavok 39
2.5.8 Prehľad o vývoji 39
2.6 Zhrnutie 40
3 NÁVRH RIEŠENIA A PRÍNOS NÁVRHU RIEŠENIA 42
3.1 Prioritizovanie požiadaviek 42
3.1.1 Zavedenie rolí 42
3.1.2 Spolupráca zavedených rolí 44
3.2 Dodávanie zákazky 45
3.3 Pravidlá pre Standup meeting 47
3.4 Zavedenie vzorca na písanie user stories 49
3.5 Kanban tabuľa 51
3.6 Ekonomické zhodnotenie 54
3.7 Prínosy riešenia 55
3.8 Zhrnutie 56
ZÁVER 57
ZOZNAM POUŽITÝCH ZDROJOV 58
ZOZNAM POUŽITÝCH SKRATIEK A SYMBOLOV 61
ZOZNAM POUŽITÝCH OBRÁZKOV 62
ZOZNAM POUŽIÝCH TABULIEK 63

ÚVOD

Projektový manažment je neoddeliteľnou súčasťou každého projektu. Za posledných niekoľko rokov sa rozšírilo používanie agilných metodík projektového riadenia. Agilné projektové riadenie je interaktívny spôsob riadenia projektov. Tento prístup riadenia je protikladom tradičného riadenia projektov, tzv. Vodopádového prístupu. Koncept tradičného riadenia projektov vyžaduje poznať kompletný zoznam požiadaviek na softvér, ktorý musí byť hotový už v prípravnej fáze. Pri agilnom prístupe je možné reagovať na zmeny požiadaviek počas realizačnej fázy projektu.

Mnoho organizácií prijíma agilné metodiky, ktoré pomáhajú zvyšovať výkonnosť tímu, zvyšovať spokojnosť zákazníka a zvyšovať univerzálnosť projektu. Organizácie, ktoré prijali agilné metodiky, sú schopné reagovať na dynamiku trhu a úspešne dokončiť viac svojich projektov.

Počas projektu sa podporuje zapojenie koncových používateľov, čo zabezpečuje viditeľnosť a transparentnosť. Počas celého procesu prebieha nepretržité plánovanie a spätná väzba, ktorá prináša hodnotu pre podnikanie od začiatku projektu. Zákazníci môžu robiť malé objektívne zmeny bez veľkých úprav rozpočtu alebo harmonogramu. Táto metóda šetrí zákazníkovi peniaze a čas, pretože testuje a schvaľuje produkt v každom kroku vývoja.

Agilných medík je viac, ale nie každá je vhodná pre všetky typy projektov. Metodiky majú svoje špecifiká, ktorých nasadenie môže priniesť pozitíva aj negatíva. Preto je pri zavádzaní agilných metodík potrebné zvážiť mnoho aspektov a charakterových vlastností firmy.

CIELE PRÁCE, METÓDY A POSTUP SPRACOVANIA

Spoločnosť MICo Vision je mladá brnenská spoločnosť, ktorá sa zaoberá najmä vývojom, výrobou, predajom vlastných röntgenových a inšpekčných systémov nielen pre priemysel. Všetky ich zariadenia obsahujú nimi vytvorený softvér pre snímanie RTG snímok a následnú prácu s obrazom a samotným vyhodnotením. Taktiež sa venuje integrovaním röntgénov do plne automatizovaných systémov navrhnutých na mieru.

Hlavným cieľom bakalárskej práce je využitie teoretických poznatkov, nástrojov a metód agilného riadenia pri vývoji softvéru u vybranej firmy.

Ďalším krokom bude implementácia agilných prvkov do softvérového tímu už spomínanej spoločnosti MICo Vision.

Potrebné informácie pre spracovanie analýzy boli získané na osobných konzultáciách so zamestnancami spoločnosti. Tieto poznatky boli následne porovnané s princípom agilného projektového riadenia. Na základe týchto metód boli v tejto práci navrhnuté riešenia pre optimalizáciu súčasného stavu softvérového tímu spoločnosti.

Pre spracovanie tejto bakalárskej práce bolo zvolených niekoľko spôsobov, vďaka ktorým sa dosiahne stanovený cieľ. Práca sa skladá z troch častí. Analýze a návrhu riešenia pre implementáciu agilných prvkov predchádzalo dôkladné naštudovanie poznatkov a pojmov najmä z oblasti projektového manažmentu, agilných metodík a nástrojov. Na základe údajov poskytnutých spoločnosťou MICo Vision sa v analytickej časti popísala spoločnosť, ich štruktúra a fungovanie softvérového tímu. Následne sa z týchto dát navrhla implementácia agilných prvkov do tímu. V návrhu riešenia boli využité všetky naštudované prostriedky, aby sa dokázali čo najviac štandardizovať procesy, ktoré pomôžu pri zvyšovaní efektívnosti tímu.

1 TEORETICKÉ VÝCHODISKA PRÁCE

V rámci tejto kapitoly bude vytvorený teoretický základ pre celú prácu. Budú tu spomenuté tradičné metodiky a ich porovnanie s agilnými metodikami. Táto časť práce priblíži Agilný manifest, jeho hodnoty a metodiky, ktoré z neho vychádzajú. Podrobnejšie sa bude venovať metodike Extrémne programovanie a Scrum. Ďalej bude spomenutý Kanban, Test Driven Development a Feature Driven Development.

1.1. Projekt

Slovo projekt pochádza z latinského názvu proicere, čo znamená hodiť niečo dopredu. Pojem projekt by sa dalo definovať ako súbor konkrétnych činností, ktoré vedú k naplneniu jedinečného cieľa. Je vymedzený časom, financiami, ľudskými a materiálnymi zdrojmi. Realizácia projektu je realizácia zmeny [1;3].

Projekt môžeme vymedziť na základe týchto piatich atributov:

- jedinečnosť
  Vzťahuje sa predovšetkým k cieľu projektu, ktorý nám hovorí, aký originálny problém sa bude riešiť a aký jedinečný výstup bude dodaný na konci projektu.
- komplexnosť
  Reprezentuje rôznorodosť metód, ktoré sú využívané podľa potrieb úmerne k životnému cyklu projektu.
- vysoká miera neistoty
  Najmä pri zahájeni je vysoká miera neistoty, z ktorej plynú riziká alebo príležitosti.
- vymedzenosť
  Čas, financie, ľudské a materiálne zdroje vymedzujú projekt a na základe ich dostupnosti je stanovený rozsah projektu.
- tím
  Vzniká v dobe zahájenia projektu a je rozpustený v momente ukončenia [1;3].

Obrázok 1 Atribúty projektu [1]

Definície projektu sa môžu líšiť podľa potrieb jednotlivých inštitúcií a spoločností. Pochopenie významu pojmu projekt je dôležité pre pochopenie problematiky projektového riadenia [3].

1.2 Vývoj softvéru

Klasické metódy riadenia projektu pri vývoji softvéru zlyhávajú. Vývoj softvéru je komplexný a empirický problém. Empirická činnosť sa veľmi ťažko odhaduje a ešte ťažšie sa plánuje. 70% softvérových projektov končí neskôr ako je plánované, prevyšujú počiatočný rozpočet a veľmi často dodajú niečo iné ako zákazník potreboval. Deje sa to najmä preto, že zákazník nevie, čo chce. Alebo vie presne čo chce, ale nedokáže to popísať svojmu dodávateľovi [4]. Agilné metodiky tento problém riešia užším spojením zákazníka s vývojovým tímom [2].

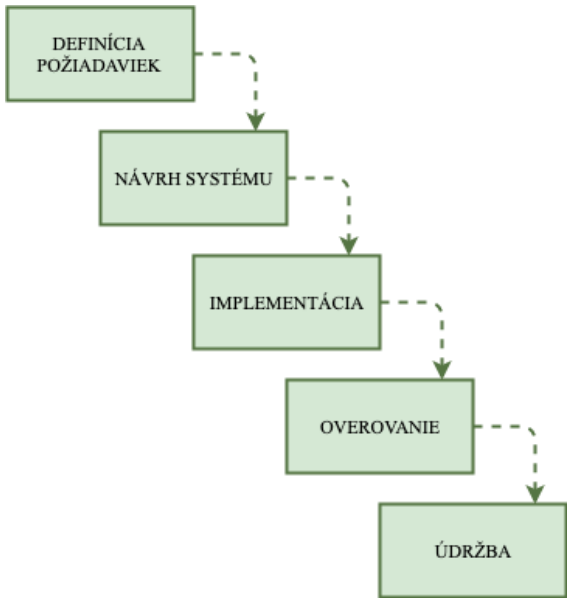
1.3 Tradičné metodiky

Toto označenie sa používa najmä preto, aby sme tradičné metodiky odlíšili od dnešných moderných agilných metodík. Väčšina z nich vznikla skôr, ale existujú aj moderné, novo vzniknuté tradičné metodiky, ktoré tiež uplatňujú tradičný prístup k vývoji softvéru. Pri týchto metodikách je snaha čo najlepšie určiť jednotlivé termíny a požiadavky.

Rolí je v tejto metodike pomerne dosť, každá má svoju špecializáciu a príliš sa nezapája do ostatných činností [5].

1.3.1 Vodopádový model

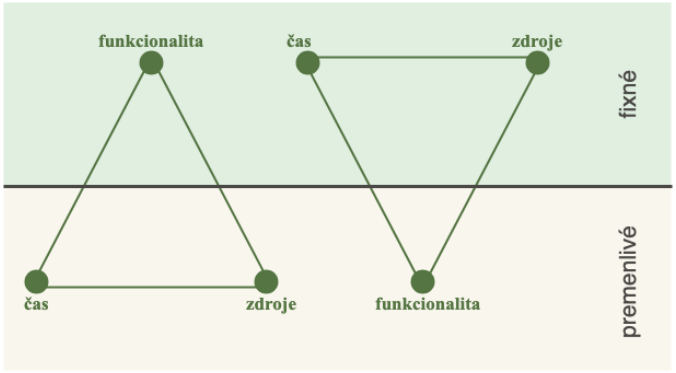
Vodopádový model nie je metodikou vývoja softvéru ale modelom životného cyklu vývoja softvérového produktu [2]. Patrí medzi najstaršie metodiky a v dnešnej dobe sa používa čoraz menej, pretože je pre väčšinu projektov nedostatočný. Znázorňuje určitý idealizovaný stav, postupnosť po sebe naväzujúcich fáz, bez cyklických návratov späť [7]. Všetky podstatné fázy, ako definícia problémov, špecifikácia požiadaviek, návrh, architektúra, implementácia a testovanie sa uskutočňuje vo vopred stanovenom poradí takmer so žiadnymi iteráciami [2;5]. Výhodou tejto metodiky je jej jednoduchosť. Metodika sa jednoducho riadi, plánuje a je jednoznačná [2].



Obrázok 2 Schéma vodopádového modelu životného cyklu [6]

1.4 Tradičné metodiky vs. Agilné metodiky

Rozdiel medzi tradičnými prístupmi a agilnými metodikami, je ukázaný na nasledujúcom obrázku.



Obrázok 3 Rozdiel medzi tradičnými a agilnými prístupmi [2]

Tradičné metodiky (znázornené ľavým trojuholníkom) vychádzajú z nutnosti za každú cenu naplniť dokument Špecifikácia požiadaviek [2]. Požiadavky (teda funkcionalita) sú fixné, zatiaľ čo čas a potrebné zdroje sú v roli premenné. V týchto premenných dochádza k zmenám v prípade, ak v projekte dochádza k ohrozeniu fixnej funkcionality. Často je dodatočne potrebné navýšiť rozpočet a pridávať do vývoja ďalších ľudí [6].

Agilné prístupy (znázornené pravým trojuholníkom) považujú čas a zdroje za fixné, zatiaľ čo požiadavky sa môžu v priebehu projektu meniť a prispôsobovať [2]. Agilný prístup považuje za výhodné čo najrýchlejšie uvoľniť produkt na trh a následne na ňom pracovať na základe spätnej väzby [6].

1.5 Agilný vývoj

Agilný je synonymom pre dynamický, interaktívny, rýchly, prispôsobivý, zábavný, rýchlo reagujúci na zmenu a mnoho ďalších. Prináša inú firemnú kultúru a náladu. Na to ako byť agilný neexistuje presný návod. Byť agilný znamená spolupracovať, komunikovať a byť pripravený na zmenu. Nejedná sa o striktný proces, ale nie je to ani žiaden chaos. Definuje hranice a vytyčuje menšie ihrisko, v ktorom rámci si tými môžu stanoviť vlastné pravidlá, tak aby sa im dobre pracovalo, boli produktívni, efektívni a dodali produkt čo najskôr. Snažia sa o to, aby bol zákazník čo najviac spokojný a dostal to, čo potrebuje [4].

1.5.1 Manifest agilného vývoja

Základným pilierom agilných metodík je manifest z roku 2001, ktorý zhrňuje v štyroch bodoch, čo znamená byť agilný [2;4].

Autori manifestu dávajú prednosť:

- **individualitám a komunikácii pred procesmi a nástrojmi**
  Procesy a nástroje pomáhajú dosiahnuť výsledky, ale nie sú pre úspech nijak kľúčové. Dôležitejší sú jednotlivci, ich potreby, prínos do projektu a tiež aj interakcia. Na druhú stranu, manifest nehovorí nič o tom, že procesy a dohody by nemali existovať, ani že by tím mal pracovať bez nástrojov. Len by mali mať možnosť si nástroje vybrať a používať len tie, ktoré im pomáhajú v dosiahnutí kvalitního výsledku. To je prvý významný rozdiel oproti tradičnému prístupu, ktorý často trvá na procesoch bez ohľadu na to, či sú ľudia v projektoch spokojní [4;5].
- **prevádzkyschopnému softvéru pred obsiahlymi, objemnými dokumentáciami**
  V agilných metodikách sa preferuje fungujúci softvér, to ale nie je dôvod k tomu, aby sa nedokumentovalo. Agilné metodiky odporúčajú obmedziť dokumentáciu na minimum, aby pomer medzi námahou a časom, ktorý sa investujete do písania dokumentácie odpovedal hodnote, ktorú zákazníci z dokumentu načerpajú. Dokumentácia sa netvorí pre to aby bola, ale aby sa softvér lepšie vyvíjal [4;5].
  Interné dokumenty sa často píšu pre budúce tímy, aby sa znalosť architektúry nestratila, ale vyhľadávanie v obsiahlych dokumentoch je vyčerpavajúce a okrem toho sa často stane, že to čo niekto hľadá tam nie je, prípadne je to v poslednej verzii už úplne inak. Interná dokumentácia má byť stručná a má pokryť len základné informácie. Ostatné je obvykle efektívnejšie dokumentovať priamo v kóde [4;5].
- **spoluprácou so zákazníkmi pred uzavretím mnoho zmlúv**
  Zmluvy sú důležité, ale nemali by byť prostriedkom nahradzujúcim komunikáciu. Často sa stáva, že že firma má preciznú zmluvu ale nemá fungujúci softvér. Preto agilné metodiky preferujú rozumnú spoluprácu a komunikáciu, ktorá umožní veľmi rýchlo zahájiť prácu na projekte, pred dokonalými zmluvami [5].

- **reakcii na zmenu pred striktným plnením plánu**

  Svet sa neustále hýbe dopredu a s tým sa menia požadavky zákazníkov, ktorí sa musia prispôsobiť trendom a konkurencii. Dodávateľ ich nemôže v týchto zmenách brzdiť. Zmena je proste nutná, ak má výsledok zákazníkovi naozaj slúžiť. Tradičné metodiky obvykle pripravia plán a požadujú jeho striktné dodržiavanie, čím sa zmeny často blokujú. Agilné metodiky sa tomu chcú vyhnúť. Snažia sa o ľudské jednanie, ktorého výsledkom je rozumná dohoda na zmenách tak, aby bol zákazník spokojný [4;5] .

1.5.2 Robiť agilne vs. Byť agilný

Existuje viacero názorov, že robiť agilne nie je to isté ako byť agilný. Ako robiť agilne sa je možné naučiť za niekoľko dní, napríklad na kurze, ktorý naučí základy Scrumu. Byť agilný si vyžaduje oveľa viac: kultúrny posun k agilnému spôsobu myslenia, ako aj zmeny spôsobu riadenia, motivácie, školení a prijímania zamestnancov [9]. Podľa certifikovaného Scrum kouča, Boba Hartmana, byť agilný znamená zmýšľať agilne a zároveň robiť agilne [10].

Robiť agilne prináša so sebou výhody ako zlepšenie komunikácie, väčší prehľad o tom, ktorý člen tímu na čom pracuje, zvýšenie produktivity a pravdepodobne aj schopnosť stanoviť priority [9].

Významné prínosy však prichádzajú s tým, že firma je agilná. To znamená, že dôsledne a rýchlo reaguje na zmeny, vyhovie zákazníkom a dosahuje excelentné výsledky prostredníctvom angažovaných zamestnancov, ktorí spoločne pracujú na spoločných cieľoch [9].

1.6 Príklady agilných metodík

Počet metodík, ktoré vychádzajú z agilného manifestu sa neustále zvyšuje. Medzi najvýznamnejšie patrí:

- Extrémne programovanie

- Scrum [2]

1.6.1 Extrémne programovanie (XP)

Poďla Kenta Becka je XP metodológia pre malé a stredné tímy vyvíjajúce softvér, ktoré čelia nejasným alebo rýchlo sa meniacim požiadavkám [17].

Názov tejto metodiky je odvodený z toho, že privádza do extrému princípy známe z iných metodík. Ak je metodika nasadená správne v správnom projekte, môže výrazne pomôcť k dosiahnutiu cieľa, ktorým je fungujúci softvér [5]. Namiesto plánovania, analyzovania a navrhovania pre ďalekú budúcnosť sa všetky tieto aktivity vykonávajú počas vývoja softvéru [20].

Medzi prvým extrém sa radí dĺžka iterácie, ktorá je podstate nižšia než v iných metodikách, jej dĺžka sa pohybuje v rozmedzí niekoľkých dní až po jeden jediný deň. Testovanie pri extrémnom programovaní prebieha neustále. Návrh, implementácia, otestovanie a nasadenie. A takto to prebieha stále dokola [5].

Do extrému je dovedená aj spolupráca so zákazníkom, a to tak, že zákazník sa stáva súčasťou vývojového tímu [5].

Čo sa týka zmien, tie sú úplne bežné. Aj niekoľkokrát denne sa menia požiadavky, s tým, že sa zahodí časť vytvoreného kódu [5].

1.6.1.1 Štyri hodnoty extrémneho programovania

Extrémne programovanie sa riadi a stavia na týchto hodnotách:

- **Komunikácia** - Celý vývojový cyklus je jedno veľké stretnutie. Tím často sedí v jednej miestnosti, kde neustále prebieha komunikácia medzi členmi tímu, zákazníkmi, programátormi a manažérmi [2; 21].
- **Jednoduchosť** - Čo nie je nevyhnutné sa nerobí. Vytvorí sa čo najjednoduchšia vec, ktorá môže fungovať [2].

- **Spätná väzba** - V projekte sa neustále testuje a ak niečo nefunguje, okamžite sa pracuje na oprave. Zákazník tiež testuje, či mu nová funkcionalita vyhovuje [2].
- **Odvaha** - Bez dostatočnej odvahy je použitie XP vylúčené. Tím musí byť pripravený, že tu ľahko môže dojsť k situácií, že bude potrebné vyhodiť napríklad týždennú prácu [2; 5].

1.6.1.2 Dvanásť základných postupov XP

1. Plánovacia hra - Zúčastniť sa jej musia obchodníci aj technici. V úvode projektu by sa mal vytvoriť veľmi stručný plán, podľa ktorého by vývoj prebiehal a priebežne sa aktualizoval. Následne by mala prebehnúť analýza zákazníka a jeho potrieb. Tieto požiadavky a predstavy sa následne premietnu do **plánu verzie**. Tento plán zahrňuje šírku zadania pre aktuálnu verziu, umožňuje zostaviť tím a odhadnúť náklady. V plánovacej hre sa tvoria **karty zadania**, ktoré sa stávajú nosičmi o požadovaných funkcionalitách [2].

2. Malé verzie - Uvoľňujú sa často a v čo najmenšej konfigurácií, predstavujú však funkčný a kompletný celok, ktorý prináša zákazníkovi úžitok [2;21].

3. Metafora – Vývoj je vedený pomocou jednoduchého príbehu o tom, ako má systém fungovať. Tento príbeh poznajú všetci účastníci [21].

4. Jednoduchý návrh – Systém je navrhovaný najjednoduchšie, ako v daný okamih môže byť. Komplikovanosť je odstránená hneď ako je ktorýmkoľvek členom tímu nájdená [2].

5. Testovanie – Najskôr sa rozhodne ako sa overí výsledok, keď bude hotový. Potom sa výsledok vytvorí a následne sa otestuje [5].

6. Refaktorizácia – Je tým myslená reštrukturalizácia systému bez zmien jeho chovania [21].

7. Párové programovanie – Všetok zdrojový kód píše dvojica programátorov, ktorí zdieľajú jeden monitor, klávesnicu a počítač. V každom páre sú dve role. Jeden z nich je tzv. šofér a druhý navigátor. Prvý píše kód, druhý všetko sleduje, pripomienkuje,

kontroluje a premýšľa o možných súvislostiach. Navzájom si role menia a menia sa aj programátorské dvojice [2;5].

8. Spoločné vlastníctvo – Ktokoľvek môže meniť akýkoľvek zdrojový text kdekoľvek v systéme v akúkoľvek dobu. V XP všetci prijímajú zodpovednosť za celý systém [2].

9. Nepretržitá integrácia – Systém je integrovaný, zostavovaný a testovaný niekoľkokrát denne [21].

10. Štyridsať hodinový pracovný týždeň - Nikde nie je zakázané mať nadčasy ale nesmie sa to stať pravidlom, keďže preťažovanie ľudských zdrojov, je známkou nesprávneho riadenia projektu [2;5].

11. Zákazník na pracovisku – Zákazník sa stáva súčasťou tímu a mal by byť k dispozícii na plný úväzok aby mohol určovať priority, odpovedať na otázky a spolupracovať na testoch [2].

12. Štandardy pre písanie zdrojového kódu – Je nutné aby programátori pri písaní zdrojového textu dodržiavali pravidlá za účelom umožniť komunikáciu prostredníctvom zdrojového textu. Zdrojový kód je základným nositeľom informácií, preto je potrebné aby bol zrozumiteľný všetkým [2].

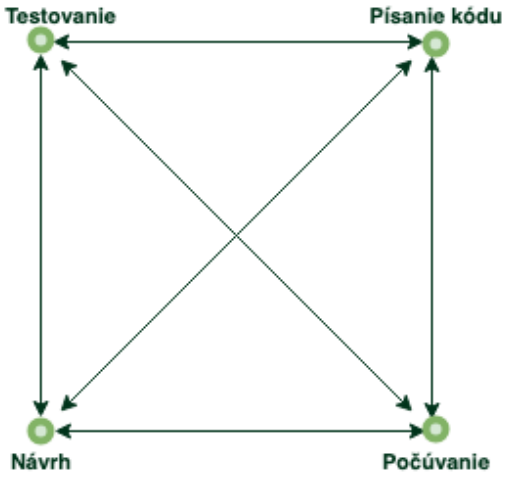
1.6.1.3 Štyri základné činnosti XP

1. Testovanie - prebieha priebežne, žiadny modul nejde integrovať bez toho, aby uspel v automatizovanom teste. Testy sú izolované (neovplyvňujú iné testy) a automatické (poskytujú nezávislý a neovplyvnený výsledok) [2].

2. Písanie zdrojového kódu - Písanie zdrojových kódov začne po pripravení príslušných testov [2].

3. Počúvanie - Vývojári musia zákazníka aj svojich kolegov vypočuť, aby vedeli čo majú implementovať [20].

4. Navrhovanie (dizajn) - Dobrý návrh usporiada logiku tak, že zmena v jednej časti systému nevyžaduje zmenu aj v ostatných častiach. Návrh je práca programátorov [2].



Obrázok 4 Vzájomný vzťah činností v Extrémnom programovaní [2]

1.6.2 Scrum

Scrum nie je proces, technika alebo definitívna metóda. Je to skôr rámec, v ktorom sa využívajú rôzne procesy a techniky [11].

Scrum najčastejšie používa tím, zaoberajúci sa vývojom softvéru, ale jeho princípy a lekcie sa dajú uplatniť na všetky druhy tímovej práce [12]. Scrum povzbudzuje tímy, aby sa učili prostredníctvom skúseností, aby sa samy organizovali pri riešení problému, aby premýšľali o svojich úspechoch a neúspechoch a aby sa neustále zlepšovali [12].

1.6.2.1 Artefakty

User story

Pri tvorbe akéhokoľvek softvéru je potrebné poznať požiadavky zákazníka. Pri správe požiadaviek sa stretávajú dve protichodné potreby:

- Pochopiteľnosť a jednoznačnosť pre vývojára
- Pochopiteľnosť a jednoduchosť pre zákazníka

Protichodné je to z dôvodu, že zákazník bude rozprávať o faktúrach, objednávkach a vývojár zas o objektoch, databázach a podobne. Preto zapísať požiadavky tak, aby vyhovovali obidvom stranám bude obtiažné [5].

User story je užívateľský príbeh toho, čo by mal systém robiť. Príbehy by mali mať určitú jednotnú podobu, ktorá má tri základné časti:

- Definícia role
- Definícia cieľu
- Definícia úžitku [5]

Príklad užívateľského príbehu:

Ako užívateľ chcem, aby som mohol vyhľadávať ostatných užívateľov podľa mena a priezviska [5].

Šprint

Šprint je iterácia, teda jeden z mnoho opakujúcich sa cyklov pri vývoji softvéru. Cieľom šprintu je vytvoriť spustiteľnú aplikáciu, ktorá bude validovateľná a testovateľná. Každý šprint musí byť naplánovaný tak, aby bol na konci spustiteľný softvér. Optimálna dĺžka šprintu je medzi dvoma až štyrmi týždňami [5].

Backlog

Backlog je zoznam user stories, ktoré je nutné implementovať do systému. Existujú dva druhy backlogov:

- Product Backlog: kompletný zoznam user stories, ktoré je nutné implementovať do systému v rámci vývoja softvéru
- Sprint Backlog: zoznam úloh, ktoré je nutné implementovať v rámci aktuálneho šprintu [5]

Velocity

Velocity je extrémne jednoduchá a výkonná metóda na presné meranie množstva práce, ktorú tím môže zvládnuť počas jediného šprintu a je kľúčovou metrikou Scrumu. Je to indikácia priemerného množstva Product Backlogu premeneného na Prírastok produktu počas Sprintu. Tím priradí body jednotlivým user stories z Backlogu na základe obtiažnosti [15]. Velocity sa vypočíta na konci Sprintu súčtom bodov za všetky úplne dokončené user stories [14;15].

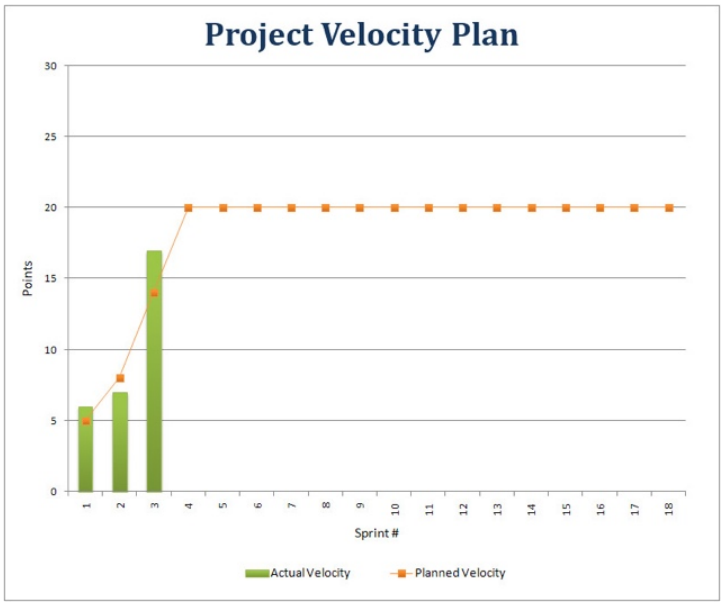
Podporí to tím v:

- Predpovedaní toho, čo je možné dodať k požadovanému termínu
- Pochopení limitov pri definovaní rozsahu, ktorý je možné splniť počas šprintu [15]

Burndown graf

Grafy slúžia ako pomôcka pri vizualizácií výsledkov.

Prvý graf je tzv. Velocity plan, teda plán, ako rýchlo bude tím postupovať. Obvykle sa vezme do úvahy, že na začiatku sa tím učí a potom už pracuje konštantnou rýchlosťou. Do plánu sa odporúča zahrnúť aj dlhšie dovolenky (napr. Vianoce) [13].



Obrázok 5 Velocity plán [12]

Project Burndown graf, kombinuje aktuálny stav s odhadmi a vizualizuje predpokladaný koniec projektu [13].

Obrázok 6 Project Burndown graf [12]

1.6.2.2 Role

Srum Master

Pracuje ako medzičlánok medzi tímom a akýmkoľvek rušivým elementom zvonku. Odstraňuje problémy, chráni pred vonkajšími vplyvmi, ktoré by mohli tím odvádzať od sústredenej práce a motivuje k lepším výsledkom. Jeho cieľom je vytvoriť samostatný, efektívny a spokojný samoorganizovaný tím. Mal by uprednostňovať koučovacie princípy, byť komunikatívny, vnímavý a utlmovať prípadné konflikty v rámci tímu. Scrum Master je členom tímu [4].

Product owner

Product owner je zástupcom zákazníka, v projektovom tíme zastupuje jeho záujmy a predovšetkým určuje, čo sa bude robiť. Ovláda vývoj softvéru a je zároveň je schopný komunikovať s ľuďmi z príslušného biznisu. Tvorí product backlog a stanovuje priority jednotlivých funkcionalít výsledného softvéru. Jeho primárnym cieľom je porozumenie produktu. Tento cieľ následne komunikuje s tímom, manažmentom a zákazníkom.

Rola Product ownera by nemala byť kombinovaná s rolou Scrum Master [2;4].

Vývojársky tím

Vývojársky tím je samoorganizovaný, vzájomne zastupiteľný a funkčný tím ľudí, ktorí sú kolektívne zodpovední za dodanie produktu, ktorý spĺňa zadané požiadavky. Ideálna veľkosť je 3-9 vývojárov [16]. Tím by mal sedieť v jednej miestnosti [4].

1.6.2.3 Stretnutia

Standup Meeting

Jedna z najznámejších praktik, ktoré sa v agilných metodikách využívajú. Tím sa stretáva pravidelne každý deň a každý z tímu zdieľa informácie o tom, na čom pracoval včera, na čom bude pracovať dnes a či má so svojou úlohou nejaký problém. Počas stretnutia by mali členovia tímu stáť (odtiaľ sa berie názov Standup). Stretnutie by sa malo konať pri fyzickej tabuli, aby bolo na prvý pohľad vidno, ktoré user stories sú ešte v sprint backlogu, na ktorých sa už začalo pracovať a čo je už hotové. Stretnutie by nemalo trvať viac ako 15 minút [4].

Sprint Planning Meeting

Cieľom tohto stretnutia je vytvoriť plán šprintu. Product Owner predstaví jednotlivé user stories a navrhuje ciele šprintu. Tím následne vyberá user stories, ktoré sú zvládnuteľné a vedú k splneniu cieľa. Scrum Master stretnutie moderuje a technicky schvaľuje konečné návrhy [4;5].

Sprint Review meeting

Prezentuje sa tu výsledok šprintu. Kedže výsledkom šprintu je spustiteľný a funkčný program, zákazníci si ho môžu prehliadnuť a vyskúšať. Počas stretnutia je snaha získať spätnú väzbu na dokončené user stories [4;5].

Sprint Retrospective meeting

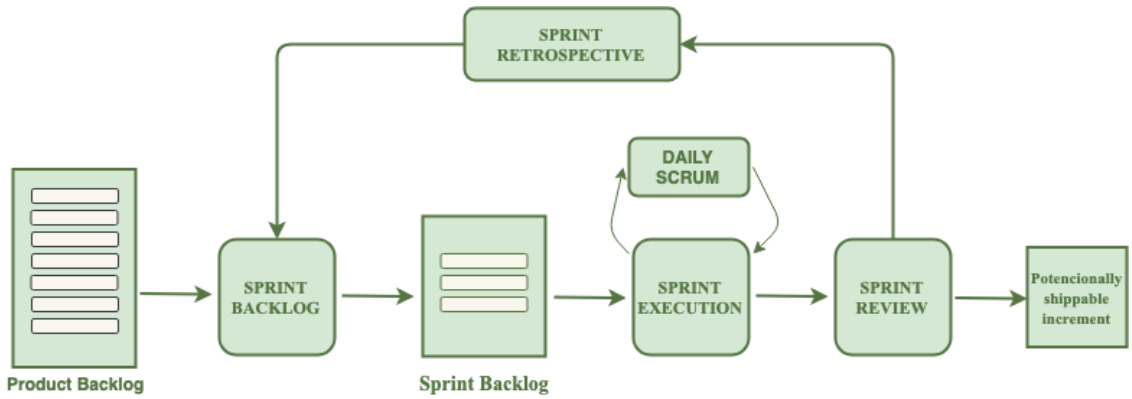
Počas stretnutia sa na uplynulý šprint tím pozerá kritickými očami a definuje, ako by sa dali procesy zlepšiť. Hodnotí sa tu všetko, čo sa počas šprintu udialo [5].

Konečný meeting

Toto stretnutie završuje celý projekt vývoja softvéru. Predáva sa tu všetka dokumentácia, odsúhlasujú sa akceptačné protokoly a dohaduje sa tu o spravovaní systému, tkz. Maintenance fáza. Výsledkom stretnutia je oficiálne a formálne ukončenie projektu [5].

1.6.2.4 Scrum proces

Proces začína tým, že Product Owner vytvorí product backlog. Pokračuje sa stretnutím tímu na Sprint Planningu, kde sa vyberú user stories, ktoré sa budú vyvíjať počas šprintu, čím vzniká Sprint Backlog. Následne začne tím pracovať. Každý deň sa koná Standup meeting. Po ukončení tím prezentuje svoj výsledok počas Sprint Review. Zhodnotenie šprintu sa koná na stretnutí Sprint Retrospective. Na konci šprintu je potenciálne doručiteľný inkrement [17].



Obrázok 7 Priebeh Scrumu [28]

1.7 Agilné nástroje

Kľúčom k úspechu v agilnom vývoji je umožniť flexibilitu pri zachovaní organizácie. Najlepším spôsobom, ako to dosiahnuť, je nasadiť súbor dobrých nástrojov [27].

1.7.1 Kanban

Kanban ma pôvod v Toyota Pruduction System. Je to nástroj na zefektívnenie procesu. Kanban nehovorí nič o tom, ako vyvíjať softvér a je možné ho aplikovať na ľubovoľný proces – je jedno či ide o Scrum, alebo vodopád [19].

1.7.1.1 5 základných princípov

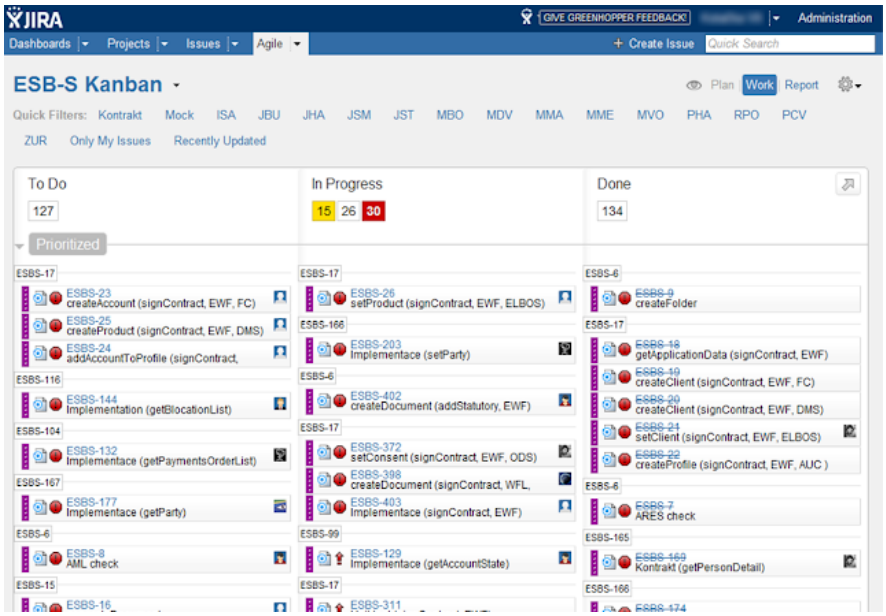
Kanban je definovaný pomocou piatich základných princípov:

1. Vizualizácia workflow

   Vizualizácia workflow sa realizuje pomocou **Kanban tabule.** Tabuľa je rozdelená na vertikále stĺpce (swim-line). Práca, ktorá sa má vykonať sa rozdelí na časti, ktoré nazývame tasky. Tasky sú prezentované kartičkami. Každý task sa umiestní do odpovedajúceho stĺpca [19;22].

   Kanban môže mať fyzickú (klasická biela tabuľa) alebo elektronickú podobu (napr. nástroj JIRA Agile). Výhodou fyzickej tabule je, že je vidieť neustále a dobre sa pred ňou robia Standup meetingy [19].

Obrázok 8 Fyzická Kanban tabuľa [19]



Obrázok 9 Elektronická Kanban tabuľa [19]

2. Limitovanie rozrobenej práce

   Kanban odporúča nastaviť explicitný limit, koľko taskov môže byť v danom stave workflow. Tomuto obmedzeniu sa hovorí Work-in-Progress (WIP). Pri používani fyzickej Kanban tabule sa limit taskov napíše nad každý stĺpec. Okrem maximálního počtu taskov je možné nastaviť aj minimálny počet taskov [22;26].Výhodou elektronickej Kanban tabule je, že nastavené WIP si systém sleduje a nedovolí limit prekročiť [19].

3. Meranie a spravovanie flow

   Aby sa vedelo, že sa proces zlepšuje je potrebné ho merať. Merá sa čas za ako dlho sa task dostane z jedného stavu do iného, typicky zo stavu *To do* do stavu *Done* [22;26]. Tento čas sa nazýva lead time. Kanban pracuje s kumulovaným lead time, keďže lead time jedného tasku nemá dostatočnú výpovednú hodnotu [19].

4. Urobenie procesných politík explicitných

   Procesné politiky slúžia na usmernenie, ako by mal tím vykonávať svoju prácu. Tieto zásady riadia postup tímu a slúžia ako hranice pre vykonávanie ich práce. Vyhlásenie týchto zásad a ich zviditeľnenie pripomína tímu pravidlá, ktorých sa majú držať pri pracovnom toku. Produktivita sa zvýši znížením času potrebného na prijímanie rozhodnutí [22].

5. Používanie modelov pre rozpoznanie príležitosti k zlepšeniu

   Tento princíp je opäť založený na vizualizácii – používa modely procesov a workflow, na základe ktorých môžeme zlepšiť celkové fungovanie procesov [26].

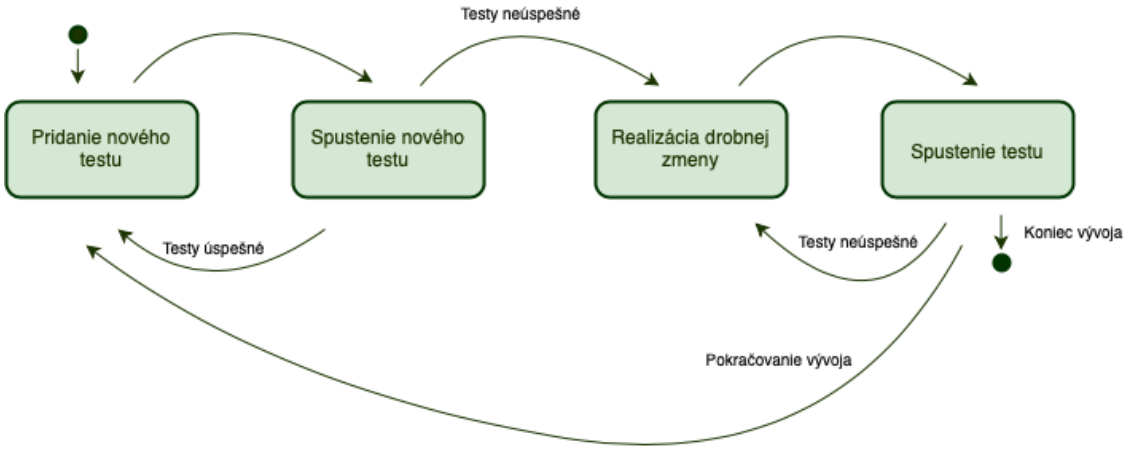
1.7.2 Test Driven Development

TDD je známe aj pod názvom „test-first programming“. Jedná sa o revolučný prístup k vývoji softvéru [2].

1.7.2.1. Fáze vývoja

Ako prvý krok sa ešte pred samotným napísaním kódu vytvorí test. Dôležité v tejto fáze je, aby vytvorený test neprebehol úspešne. Až po napísaní zdrojového kódu musia prebehnúť všetky testy úspešne. Následne sa vykonajú potrebné úpravy a odstránia sa duplicity. Test sa presunie do testovacej sady a overí sa, či nebola porušená integrita testov a ich logická previazanosť. Cieľom TDD je mať čistý kód, ktorý funguje [2;23].

Tieto kroky sa opakujú počas celého vývoja. Ak sa nepodarí aby test prebehol úspešne je najvhodnejšie zahodiť test aj zdrojový kód a implementovať nový, jednoduchší test (a teda aj nový a lepší kód) [2].



Obrázok 10 Grafické znázornenie metodiky Test Driven Development [2]

1.7.2.2. Výhody

1. včasné oznámenie o chybe

   Vývojári testujú svoj kód, ale vo svete databáz to často pozostáva z manuálnych testov, alebo jednorazových skriptov. Pomocou TDD si časom vývojári vybudujú balík automatizovaných testov [23;24].
2. lepšie navrhnutý, čistejší a rozšíriteľnejší kód

   Pomáha pochopiť, ako sa bude kód používať a ako interaguje s inými modulmi. Výsledkom je lepšie rozhodnutie o návrhu a lepšia údržba kódu. TDD umožňuje písať kód, ktorý má jednu zodpovednosť, čo uľahčuje pochopenie kódu. TDD tiež núti písať iba kód, ktorý ma prejsť testami na základe požiadaviek užívateľa [23;24].
3. dobré pre tímovú prácu

   Ak je neprítomný nejaký člen tímu, ostatní členovia tímu môžu jednoducho začať pracovať na kóde. Pomáha to zdieľať vedomosti, čím sa tím celkovo zefektívňuje [24].
4. dobré pre vývojárov

   Aj keď vývojári musia venovať viac času písaniu testov, na ladenie a vývoj nových funkcií je potrebné oveľa menšie množstvo času. Podporuje to v písaní čistejšieho a menej komplikovaného kódu [24].

1.7.3 Feature Driven Development

FDD je založené na vývoji, ktorý je riadený vlastnosťami produktu. Vzhľadom k tomu akým spôsobom sú v metodike definované stavebné kamene - vlastnosti, je relatívne jednoduché vykonávať priebežné kontroly a získavať presvedčenie, že vývoj ide správnym smerom [2].

1.7.3.1. Fáze vývoja

1. Vytvorenie celkového modelu

V tejto fáze sa vytvorí základný (skeletový) model. Až po vytvorení tejto kostry sa preberajú ďalšie vlastnosti a podrobnosti o očakávanom riešení [2].

2. Vytvorenie zoznamu vlastností

Táto fáza je zameraná na vytvorenie podrobného zoznamu jednotlivých vlastností. Je dôležité stále pamätať na to, že vlastnosť je to, čo zákazník ocení. Následne sa vykoná identifikácia konečného výsledku, aby sa ujasnilo aká množina vlastností bude považovaná za dokončený produkt [2;25].

3. Plánovanie

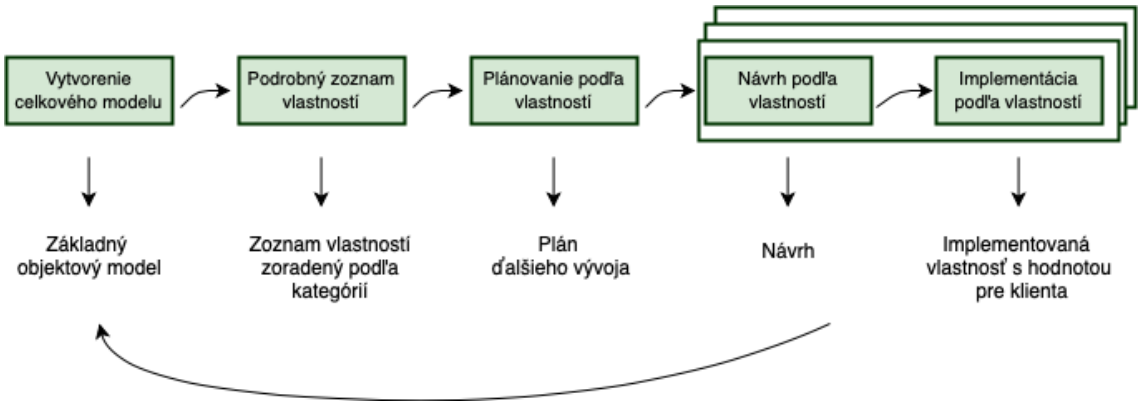
Analyzuje sa zložitosť každej vlastnosti a plánujú sa úlohy, ktoré majú členovia tímu vykonať. Určí sa poradie, v ktorom sa každá vlastnosť implementuje a priradí sa jej „vlastník“ a dátum ukončenia. Táto fáza by mala tiež identifikovať vlastníkov triedy a jednotlivých vývojárov, ktorí sú zaradení do tried [25].

4.Návrh

Posledné dve fázy sú iteratívne opakované. V tejto fáze sa pracuje už s konkrétnymi vybranými vlastnosťami. Hlavný programátor vyberie jedno alebo viacero vlastností a dbá na to, aby vybraná množina vlastností bola realizovateľná v jednom až dvoch týždňoch. Hlavný programátor následne spustí proces DBF („design by feature“). DBF spočíva v určení tried, ktoré danú vlastnosť pokrývajú. Následne vlastníci tried vytvoria tím, ktorého úlohou je vypracovať detailný návrh pre implementáciu a stanoviť plán pre realizáciu [2].

5. Implementácia

Táto fáza zahrňuje činnosť označovanú ako BBF („build by feature“). Vlastník triedy píše metódy, vytvára testovacie prípady a vykonáva testy. V okamihu keď je hlavný programátor s dosiahnutým výsledkom spokojný je vlastnosť integrovaná do hlavnej aplikácie (jej predošlej verzie) [2].



Obrázok 11 Postupnosť vývojových fáz v metodike Feature Driven Development [2]

2 ANALÝZA SÚČASNÉHO STAVU

Časť bakalárskej práce venovaná analýze súčasného stavu, predstaví spoločnosť MICo Vision s.r.o. Popíše aktuálny stav fungovania softvérového tímu umožní tak identifikáciu problému a spracovanie návrhov.

2.1 Charakteristika podnikateľského subjektu

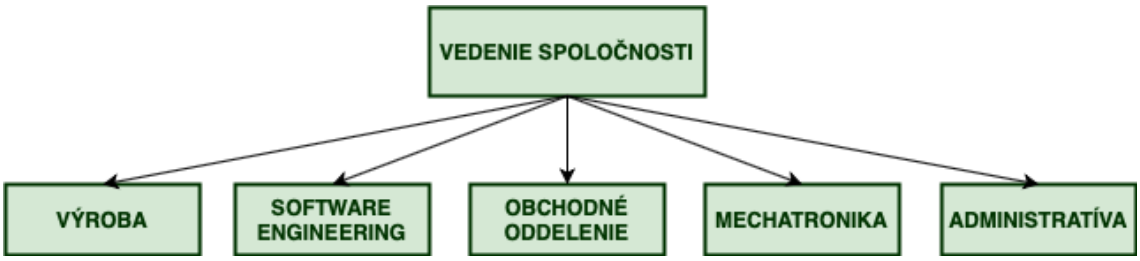
Spoločnosť Mico Vision s.r.o. vznikla v roku 2014 a sídli v Technologickom parku v Brne. Víziou spoločnosti je zabezpečovať špičkovú kvalitu výroby pomocou implementácie röntgenovej kontroly do výrobných podnikov. Spoločnosť dodáva softvér do röntgenových prístrojov vyrobený na mieru. Všetky prístroje sú po mechanickej, elektrickej aj softvérovej stránke navrhnuté vývojarmi spoločnosti. Vďaka tomu flexibilne reagujú na potreby podniku a prístroje vedia individuálne prispôsobiť.

Tabuľka 1 Základné informácie o spoločnosti (vlastné spracovanie)

Názov spoločnosti	MICo Vision s.r.o.
Právna forma	spoločnosť s.r.o.
IČ	02581035
Sídlo spoločnosti	Budischowského 1073, Borovina, 674 01 Třebíč
Dátum vzniku	29.1.2014
Predmet podnikania	výroba, obchod a služby neuvedené v přílohách 1 až 3 živnostenského zákona
Webové stránky	www.vision.mico.cz

2.2 Organizačná štruktúra

Vedenie spoločnosti tvoria v súčasnej dobe dvaja jednatelia. Tí majú pod sebou celkovo päť oddelení: výroba, software engineering, obchod, mechatronika a administratíva.



Obrázok 12 Schéma vedenia spoločnosti (vlastné spracovanie)

2.3 Predmet podnikania

Spoločnosť ponúka štandardizované riešenia röntgénov, ktoré sa uplatňujú nielen v priemysle, ale aj v bezpečnosti. Ďalej ponúkaju riešenia na mieru, podľa požiadaviek zákazníka na kľúč. Tieto zariadenia sa vyznačujú automatickou prevádzkou 24/7 bez nutnosti obsluhy stroja a je možne ich integrovať priamo do výrobnej linky.

Zariadenia obsahujú vlastný softvér spoločnosti a po pripojení na internet ich dokáže spravovať na diaľku. Vďaka využitiu čítačky kódu produktu je možné archivovať všetky snímky konkrétnych produktov (vďaka čomu je možná spätná identifikácia a preukázanie kvality). Röntgénové systémy umožňujú napojenie na podnikové informačné systémy, alebo cloudové riešenia, vďaka čomu je možné získané dáta archivovať v rámci celého podniku a využiť ich při zlepšovaní výrobných procesov.

Návrh robotického kontrolnho systému je vždy realizovaný v úzkej spolupráci s výrobným podnikom.

Obrázok 13 Sciox custom [31]

2.4 Nástroje

- **RingCentral**

  Na komunikáciu využíva celá spoločnosť chatovací kooperačný nástroj RingCentral.

  RingCentral ponúka tímové chaty, videohovory, zdieľanie dokumentov, zdieľanie kalendára, vytváranie a sledovanie úloh.

- **Gitlab**

  Softvérový tím využíva k rozdeľovaniu úloh, spoluprácu na kóde a bug trackingu nástroj Gitlab.

  GitLab je kompletná platforma DevOps dodávaná ako jedna aplikácia. Poskytuje jedinú aplikáciu pre celý vývoj softvéru a životný cyklus operácií.

- **Xwiki**

  Na písanie dokumentácie a návodu je využívaný nástroj xwiki.

  XWiki je bezplatná softvérová platforma napísaná v jazyku Java.

- **Google G suite**

  Na zdieľanie súborov a kancelárských dokumentov je využívaný nástroj Gsuite.

  Google G Suite je predplatné cloudových služieb a kancelárskych aplikácií od spoločnosti Google.

  Výhodou je, že G Suite je prevádzkovaný vo veľkých dátových centrách, takže spoločnosť nepotrebuje investovať do serverov. Taktiež majú zamestnanci neustály prístup k firemnému e-mailu a dátam, nech sa nachádzajú kdekoľvek.

  Obsah služby **Google G suite**, ktoré spoločnosť využíva:

  Videohovory - Hangouts Meet

  Kalendáre - Google Kalendár

  Sociálna sieť pre firmu - Google+

  Dátové úložisko - Google Disk

  Archivácia e-mailov a chatov - Google Sejf

2.5 Technológie

Spoločnosť využíva nasledujúce technológie:

- C#, C+ +, Python, Javascript
- OpenCV, Jupyter Notebook
- Linux, Windows
- Docker, Git, GiLab

Pri vývoji je používané:

- Code reviews
- Automatické testovanie kódu
- Continuous integration
- Verzovanie
- Pokročilé komunikačné nástroje
- Zostavovanie knowledge-databázy (wiki)

2.5 Softvérový tím

SW tím spoločnosti zaisťuje tri hlavné odvetvia:

- Vývoj softvéru pre univerzálne a jednoúčelové priemyslové röntgénové prístroje
- Nasadzovanie a správa vlastního softvéru a softvéru tretích strán pre dodávateľov prístrojov
- Správa IT infraštruktúry a IT vybavenie spoločnosti

2.5.1 Veľkosť tímu

Optimálna veľkosť sa pohybuje medzi 3-9 členmi (viď. 1.6.2), softvérový tím vo firme má deväť členov, ktorých jednotlivé role sú:

1 x Head of Development

1 x Lead Developer

5 x Developer

1 x Junior Developer

1 x IT Support & Customer Care

2.5.2. Spolupráca

Tím spolupracuje na väčšine dodávaných prístrojov s oddelením mechatroniky. Vo väčšine prípradov spočíva spolupráca v predávaní elektronických schémat a mechatronických výkresov.

Ďalej prebieha spolupráca s oddelením výroby. Oddelenie výroby pripravuje stroje do fázy, kedy je potrebné ho softvérovo oživiť. V prípade závad na mechanike vykonáva výroba úpravy na mechanike/elektronike prístroja podľa spätnej väzby softvérového tímu.

Head of Development reportuje výsledky softvérového tímu vedeniu spoločnosti.

Head of Development a Lead Developer spolupracujú s obchodným oddelením, oddelením mechatroniky, a vedením spoločnosti pri odhade náročností pre zostavenie cenových ponúk budúcich projektov.

2.5.3 Vývojový proces

Vývoj sa dá rozdeliť do dvoch základných skupín:

1. Dlhodobý vývoj – knižnice a hlavné produkty (softvér pre univerzálny prístroj SCIOX)
2. Krátkodobý vývoj – jednoúčelové stroje a zákazky

Dlhodobý vývoj prebieha kontinuálne a nie je časovo rozdelený. Krátkodobý vývoj je väčšinou daný termínom projektu. Cieľom je maximalizovať dlhodobý vývoj a vylepšovať základňu knižnice, ktorá zjednoduší a zlacnie krátkodobý vývoj.

Pre krátkodobý vývoj je dôležitá správna špecifikácia zadania. Tá je riešená priamou komunikáciou softvérového tímu so zákazníkom.

2.5.4 Prioritizovanie požiadavok

Pri prioritizovaní požiadaviek sa stretáva pohľad zákazníka a pohľad vývojára, ktoré sú z pochopiteľných dôvodov občas rozličné. Pre spoločnosť je dôležité nájsť kompromis. Tímu by pomohlo stanovenie rolí, kde by jedna predstavovala hlas zákazníka a druhá hlas tímu, ktorý by definoval čo tím dokáže stihnúť v určitom časovom období. Okrem ich stanovenia je dôležité ako bude prebiehať ich spolupráca.

2.5.5 Dodávanie zákazky

Zákazníci oceňujú transparentnosť, obchodnú hodnotu, angažovanosť a zmeny. Zároveň chcú vedieť, koľko ich daná sada funkcií bude stáť, čo bude implementované a kedy to bude hotové.
Spolupráca so zákazníkom prebieha v nasledujúcich krokoch:

- Dopyt
- Schôdzka
- 3D model
- Realizácia
- Dodanie
- Inštalácia, zaškolenie a predanie
- Podpora v produkcii

V prvej fáze sa zákazník a firma dohodnú na dodaní požadovaného výrobku. Druhá fáza má dve iterácie. V prvej sa ukazuje zákazníkovi 3D model, v druhej iterácii si prezerá fungovanie stroja vo firme. Tretia fáza je dodanie zákazky. Vzhľadom k tomu, že s podobným typom prístroja zákazník skúsenosť nemá, problémy a pripomienky prichádzajú až po nasadení stroja do prevádzky. Ideálne by bolo dodávať stroj ako službu, to ale v niektorých prípadoch nie je možné.

2.5.6 Stand Up

V súčasnej dobe tím používa len pár prvkov z agilného riadenia. Jedným z nich je Standup meeting, ktorý sa koná dva krát v týždni (každý utorok a štvrtok od 11:00) a vedie ho Head of Development. Zúčastňuje sa ho celý tím. Počas celého stretnutia každý člen stojí a odpovedá na tri otázky: na čom pracoval od posledného stretnutia, na čom bude pracovať dnes a či má so svojou úlohou nejaký problém. Člen, ktorý rozpráva má v ruke predmet, ktorý následne podáva ďalšiemu členovi. Ak niekto začne rozprávať mimo témy, iný člen tímu ho zastaví.

Tím sa snaží aby bolo stretnutie čo najrýchlejšie a naefektívnejšie a jeho trvanie sa pohybuje okolo 15 min. Koná sa v kancelárií kde tím sedí.

Standup meeting by sa mal konať každý deň a mal by spĺnať pravidlá, ktoré si tím nastaví čím sa zvýši jeho efektivita (viď. 1.6.2.3).

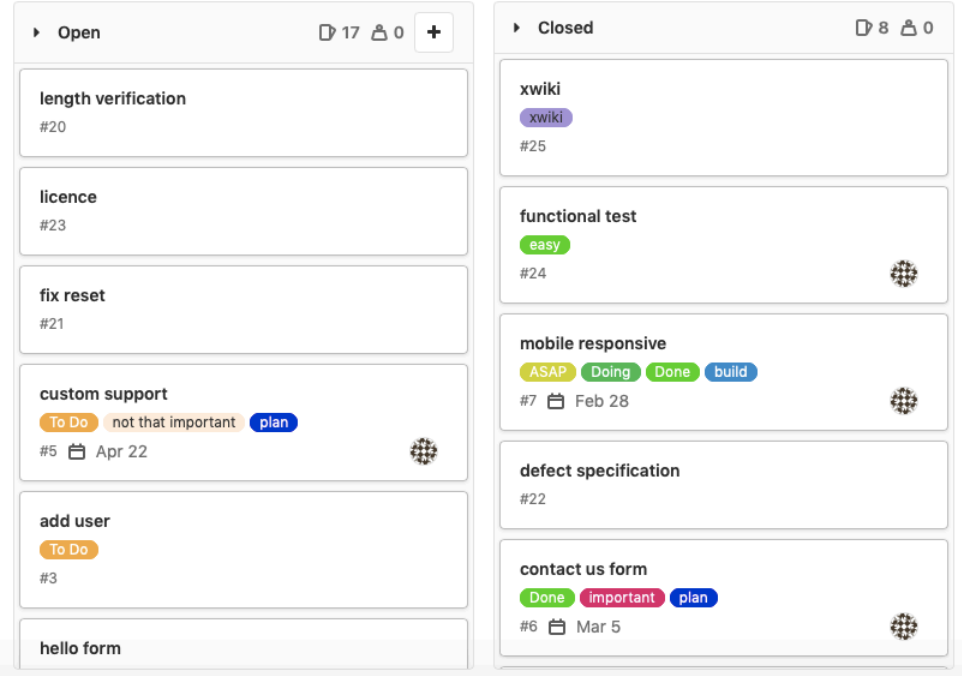
2.5.7 Spísanie požiadavok

V súčasnej dobe komunikuje zo zákazníkom Head of Development a člen tímu, ktorý sa venuje IT Support & Customer Care. Požiadavky nemajú presne stanovenú formu a väčšina požiadavok je zhrnutá hneď na začiatku ich spolupráce so zákazníkom. Požiadavky je možné spísať podľa presne stanovej formy (viď. 1.6.2.1).

2.5.8 Prehľad o vývoji

Tím využíva nástroj Gitlab, ktorý primárne používajú pre správu a sledovanie zmien v zdrojovom kóde behom celého vývoja softvéru. Slúži im aj k rozdeľovaniu úloh, zdieľaniu kódu a bug trackingu. Hlavnou výhodou používania tohto nástroja je minimalizovanie rizika straty dosiahnuté práce a možnosť dohľadania všetkých zmien, ktoré boli vykonané v rámci vývoja aplikácie v tzv. verziách, ktoré sa vytvárajú pomocou pull request. Tieto pull request sa následne synchronizujú so špecifickou vetvou, v ktorej je obsiahnutý zdrojový kód aplikácie, ktorý slúži ako východiskový bod pre všetky nasledujúce vývojové úlohy

Členovia tímu si označujú tasky týmito jednotlivými stavmi a to im umožňuje jednoduchšiu orientáciu medzi ich veľkým množstvom. Momentálne má tím jednu veľkú tabuľu, kde sa nachádzajú tasky všetkých užívateľov.



Obrázok 14 Tabuľa v GitLabe [29]

Tabuľa by sa mohla rozdeliť na niekoľko stĺpcov (To do/ In progress / Done) po vzore Kanban tabule (viď. 1.7.1.1). Ďalej by si každý člen tímu vytvoril svoju vlastnú Kanban tabuľu, a vedúci tímu by si mohol jednoducho filtrovať členov tímu so zobrazením aktuálnych stavov taskov, na ktorých pracujú. Zvýšil by si tým prehľad vedúci tímu aj jednotlivci, ktorí nie sú súčasťou tímu.

2.6 Zhrnutie

V rámci firmy sa agilné prvky využívajú najmä v softvérovom tíme. Tím ma v zavádzaní agility plnú podporu riaditeľa aj ostatných vedúcich pracovníkov. Prvky, ktoré majú v tíme nasadené akceptuje a podporuje každý člen. Tím prvky do procesov pridáva

postupne, ako to situácia vyžaduje a nič nenasadzuje nútene. Zmyslom je štandardizovať procesy, ktoré pomôžu pri zvyšovaní efektívnosti tímu.

Spoločnosť využíva metodológie, postupy a principy, ktoré reflektujú dobu a reagujú na aktuálne požiadavky zákazníka. Ďalšia časť práce navrhne riešenia, ktoré by mali pomôcť tímu zlepšiť niektoré procesy.

Na základe analýzy boli nájdené oblasti, kde je možný posun k lepšiemu:

- V tíme by mohlo byť nastavené efektívnejšie prioritizovanie požiadaviek.
- V ideálnom prípade by bolo dodávať stroj ako službu. To ale v niektorých prípadoch kvôli požiadavkam zákazníkov nie je možné.
- Je možné zefektívniť stretnutie Standupu.
- Požiadavky je možné spísať podľa presne určenej formy.
- Vedúci tímu aj ostatní členovia si môžu zvýšiť prehľad o postupe plnenia jednotlivých úloh.

Riešeniu spomínaných oblastí sa bude venovať nasledujúca kapitola.

3 NÁVRH RIEŠENIA A PRÍNOS NÁVRHU RIEŠENIA

V tejto kapitole budú navrhnuté riešenia problémov, ktoré sú popísané v analýze súčasného stavu.

3.1 Prioritizovanie požiadaviek

Požiadaviek je stále viac, ako sa za daný čas dokáže stihnúť. Požiadavky je potrebné vyberať podľa potrieb zákazníka, ale aj podľa toho, čo tím za daný čas stihne dodať. V tíme je potrebné stanoviť role, kde by jedna predstavovala hlas zákazníka a druhá hlas tímu, teda čo tím dokáže stihnúť v určitom časovom období. Z tohto vyplýva, že predtým než sa stanoví ako sa budú požiadavky prioritizovať je potrebné stanoviť v tíme role, ktoré to budú vykonávať.

Tieto role bude predstavovať Scrum Master a Product Owner. Výsledkom ich vzájomnej spolupráce budú implementované vlastnosti podľa potrieb zákazníka.

3.1.1 Zavedenie rolí

Product Owner

Túto činnosť by mohol vykonávať člen, ktorý sa venuje IT podpore a komunikácii so zákazníkom. Vďaka definovaniu jeho role ako Product Owner by sa jasne stanovila jeho náplň a aj to, akým spôsobom bude získavať a následne doručovať získané informácie od zákazníka k tímu.

Náplň práce:

- Zúčastňuje sa denného standup meetingu
- Je k dispozícii vývojárom pre priebežné vyjasňovanie user stories, na ktorých sa práve pracuje
- Píše user stories
- Prioritizuje backlog spolu so Scrum Masterom určuje požiadavky, ktoré sa budú vyvíjať v nadchádzajúcom šprinte

- S tímom diskutuje nad navrhovanými riešeniami, akceptuje alebo odmieta navrhnuté alternatívy
- Účastní sa dema a akceptuje prezentovanú dodávku
- Testuje vyvinuté funkcionality
- Diskutuje so Scrum Masterom nad problémami v tíme a hľadá ich riešenia
- Informuje stakeholderov o aktuálnom statuse šprintu a nadchádzajúcom release

Scrum Master

V súčasnej dobe má najblžšie k roli Scrum Mastera Head of Development, ktorý vedie tím, standupy aj komunikáciu so zvyškom firmy.

Vzhľadom k tomu, že je plnohodnotný vývojár v tíme a firma sa na miesto Scrum Mastera rozhodla nikoho nezamestnať jeho náplň práce nebude obsahovať všetko to, čo by mal Scrum Master vykonávať. Cieľom je, aby aj naďalej tím viedol, motivoval a určoval požiadavky, ktoré sa budú implementovať.

Náplň práce:

- Udržuje správnu formu Standup meetingu – sleduje čas a upozorní člena tímu ak začne rozprávať mimo témy
- Spolu s Product Ownerom určuje požiadavky, ktoré sa budú vyvíjať v nadchádzajúcom šprinte
- Premýšľa akým spôsobom má facilitovať diskusiu o téme, ktorú posledné dni tím páli, aby z toho vznikla produktívna diskusia s kvalitními výstupmi
- V polovici Šprintu vykonáva review dodávky hlavných user stories s Product Ownerom
- Zdieľa s Product Ownerom informácie o nálade v tíme, upozorňuje ho na technický dlh, ktorého riešenie by bolo vhodné naplánovať do ďalšieho šprintu
- Rieši s tímom úpravu podoby Definitin of Done, následne o navrhovanej úprave diskutuje zo všetkými navrhovanými stranami

Po definovaní rolí Scrum Mastera a Product Ownera je potrebné definovať ich vzájomnú spoluprácu a taktiež zapojenie zvyšku tímu.

3.1.2 Spolupráca zavedených rolí

Požiadavky, ktoré má Product Owner zo stretnutia so zákazníkom sú spísané vo forme user stories a ako prvé je potrebné rozhodnúť sa, na ktorých sa začne pracovať najskôr.

Prioritizovanie user stories vykonáva Scrum Master spolu s Product Ownerom. Scrum master je skúsený a má predpoklad čo najlepšie odhadnúť zložitosť jednotlivých user stories, zatiaľ čo ho Product Owner usmerňuje, akú hodnotu má daná požiadavka pre zákazníka. Výsledkom je konsenzus na tom, čo je pre zákazníka najdôležitejšie a čo sa stihne za stanovenú dobu dodať.

Na prioritizovanie bola navrhnutá jednoduchá numerická prioritizačná metóda,

pričom každá user story má jednu z nasledujúcich priorít:

- User story klasifikované ako 1 sú kritické a musia sa dodať.
- User story klasifikované ako 2 majú hodnotu, ale ak je potrebné môžu sa obmedziť ak je to potrebné z dôvodu nedostatku času alebo rozpočtu.
- User story klasifikované ako 3 majú buď nižšiu prioritu alebo je potrebné nad nimi ďalej diskutovať.

Jednotlivá hodnota user story je flexibilná a časom sa môže zmeniť.

Po prioritizovaní prichádza na rad **spísanie akceptačných kritérií**. Akceptačné kritérium je definovanie stavu, kedy sa bude môcť považovať user story za dokončené. Za ich spísanie je zodpovedný Scrum Master.

Následne Scrum Master user story rozdelí do taskov, ktoré je potreba splniť pre jej dodanie. Do procesu prichádzajú dvaja členovia tímu, ktorý sú zodpovední za **ohodnotenie zložitosti a doby trvania taskov**. Ich úlohou je poskytnúť odhad na to, aká je potrebná skúsenosť vývojára na splnenie tasku a koľko dní to bude trvať.

Tabuľka 2 Vzor spísania user stories (vlastné spracovanie)

priorita	user story	akceptačné kritéria	majiteľ user story	tasky	doba trvania	potrebná skúsenosť
Klasifikácia user story (1/2/3)	Zapísanie user story v tvare: **ako (typ užívateľa) chcem (nejaká vlastnosť) takže (nejaký dôvod)**	Definovanie stavu, kedy sa user story bude môcť považovať za dokončené	Zodpovedná osoba za dodanie splnenej user story	task 1	x hod.	1/2/3
				task 2	x hod.	1/2/3
				task 3	x hod.	1/2/3

Potrebná skúsenosť je ohodnotená od 1 do 3 kde 1 značí menej skúseného vývojára a 3 skúseného vývojára. Doba trvania je udaná v hodinách.

Náklady na navrhnuté riešenie

Náklady sú nulové, jedná sa len o zmeny v aktuálnom fungovaní tímu.

Zhrnutie prínosu riešenia

- Dodanie vlastností, ktoré budú pre zákazníka najprínosnejšie
- Jasne stanovená kooperácia jednotlivých členov v rámci tímu
- Presnejšie odhadnutie trvania dodania jednotlivých požiadaviek
- Užšia spolupráca so zákazníkom

3.2 Dodávanie zákazky

V predošlej kapitole bol popísaný súčasný priebeh dodávania zákazky. Prebieha v troch fázach, a pre spoločnosť je dôležitá akceptácia zákazníka, že dodaním stroja ich spolupráca nekončí.

Pre lepšiu definíciu prostredia, kde sa má stroj nasadzovať by pomohlo, ak by po spísaní prvotných požiadavok Product Owner prevádzku u zákazníka navštívil a strávil by tam jeden týždeň. Mal by priestor na pozorovanie prostredia, získavanie informácií od zamestnancov, vďaka čomu by vedel vypracovať analýzu, ktorá by prevádzku priblížila aj zvyšku tímu. Pri tomto type prístrojov je dôležité aké prístroje sa nachádzajú v okolí, ako by zamestnanci dokázali čo najefektívnejšie obsluhovať stroj a rôzne iné poznatky, ktoré je možné získať len pri detailom analyzovaní okolia.

K súčasným dvom iteráciám (zobrazovanie 3D modelu a ukazovanie prístroja vo firme) by sa pridala tretia. Stroj by bol do prevádzky nasadený aspoň 10 dní pred plánovaným odovzdaním. Tím by tak mohol pracovať na pripomienkach a problémoch, ktoré vznikajú priamo na mieste. Tieto problémy a pripomienky zo strany zamestnancov zákazníka sú jedinečné a nedokážu sa odhadnúť bez nasadenia do prevádzky.

Po ukončení a uvedení stroja do prevádzky by sa spolupráca so zákazníkom neskončila. Služba by ďalej pokračovala, do stroja by sa implementovali časom nové funkcie a vlasnosti. Toto riešenie vyžaduje agilné zmýšlanie aj u zákazníka. Používanie agilného prístupu k vývoju softvéru potvrdzuje, že časom dôjde k zmene rozsahu.

Náklady na navrhnuté riešenie

Toto riešenie zo sebou prináša náklady v súvislosti s návštevou Product Ownera prevádzky, kde by strávil aspoň jeden týždeň. Ďalšie náklady vznikajú s nasadením stroja do prevádzky pred plánovaným odovzdaním, kde vzniká potreba aby boli zamestnanci prítomní pri testovaní fungovania stroja v prevádzky. Vzniknuté náklady pozostávajú z nasledujúcich náhrad: náhrada preukázaných cestovných výdavkov, náhrada preukázaných výdavkov za ubytovanie, stravné, náhrada preukázaných potrebných vedľajších výdavkov. Celková výška nákladov na jednotlivé návštevy prevádzky závisí od jej lokality.

Zhrnutie prínosu riešenia

- Lepšie prispôsobenie fungovania stroja pre potreby zákazníka
- Možnosť dodania ďalšej funkcie po určitej dobe

- Sledovanie prevádzky a obsluhovanie stroja jej zamestnancami prinesie firme poznatky, ktoré môžu využiť pri ďalších projektoch

3.3 Pravidlá pre Standup meeting

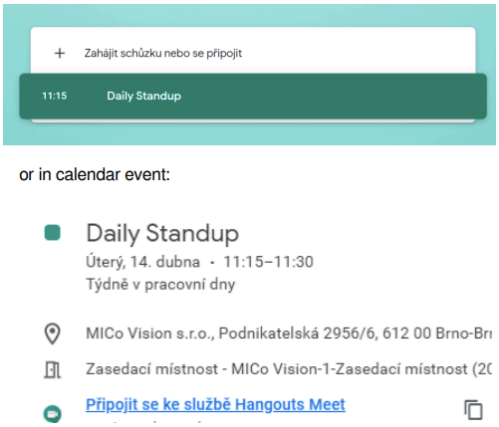
Pre toto stretnutie bolo navrhnutých niekoľko pravidiel, ktoré by ho mali zefektívniť. Tím doteraz praktizoval stretnutie dvakrát do týždňa o 11:00.

Po vykonaní zmien sa stretnutie bude konať každý deň. Začiatok sa stanovil na 11:15, keďže tím chodí na obed o 11:30. Posunutie o 15 minút je výhodné z dôvodu, že každé stretnutie a každá prestávka prináša so sebou stratené minúty na prípravu. V tomto prípade sa koná stretnutie a hneď po ňom je obedná prestávka po ktorej sa každý púšťa späť do práce. Taktiež je tu výhodou, že nikomu sa pred obedom nechce diskutovať nad nepodstatnými vecami a stretnutie by sa malo pomerne jednoducho stihnúť do 15 minút.

Keďže sa stretnutie vykonáva v miestnosti, kde členovia sedia celý deň, by sa stretnutie mohlo posunúť do zasadacej miestnosti hneď vedľa kancelárie. Presun tímu by netrval ani minútu ale zmena prostredia, a pár krokov na premiestnenie by mohli mať na členov pozitívny vplyv.

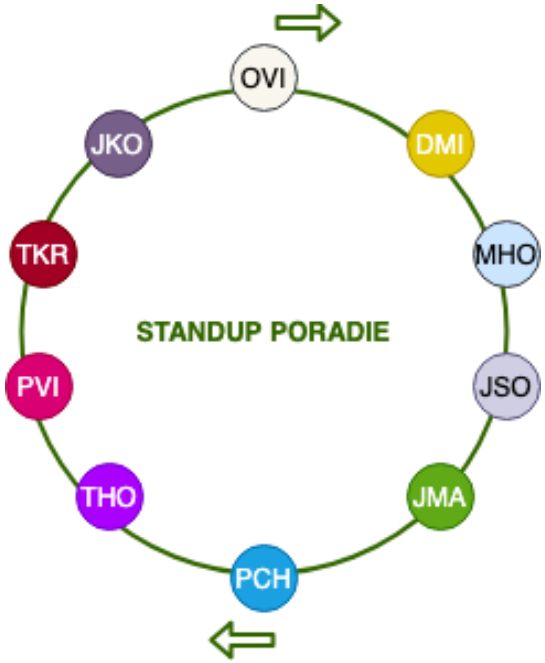
Zo začiatku by mohol vedúci stretnutia nastavovať časovač na 15 minút. Po ukončení časového limitu by stretnutie malo skončiť. Tím by si časom zvykol a stihol by prebrať potrebné veci do 15 minút.

Každý člen tímu by mal mať v kalendári nastavenú pripomienku o konaní Standupu.



Obrázok 15 Pripomienka StandUp v kalendári [30]

V súčasnej dobe sa čoraz častejšie využíva možnosť pracovať z domu, prípadne nie všetci členovia tímu vykonávajú prácu z rovnakého sídla a preto bolo navrhnuté ako by v tomto prípade mohol StandUp prebiehať. Platforma, ktorú by tím pre tieto stretnutia využíval je Hangouts Meeting, ktorá je súčasťou G Suite a je bezplatná. V kalendári, ktorý konanie StandUpu každý deň pripomína, sa nachádza odkaz, ktorý členov prepojí na videohovor. Videohovoru sa naraz môže zúčastniť všetkých deväť členov tímu. Poradie členov je nasledovné:



Obrázok 16 StandUp poradie (vlastné spracovanie)

Náklady na navrhnuté riešenie

Toto riešenie so sebou neprináša žiadne náklady. Členovia tímu si len potrebujú na toto stretnutie vyhradiť 15 minút denne. Videohovory prostredníctvom Hangouts Meeting sú bezplatné.

Zhrnutie prínosu riešenia

- Zlepšenie produktivity, keďže každý člen tímu vie na čom má pracovať
- Vedúci tímu bude mať prehľad o postupovaní každého člena tímu

- Zvýšenie súdržnosti tímu, každý má prehľad o tom, čo sa deje a na vzniknuté problémy môže ponúknuť včasné riešenie
- Postavenie sa a zmena od celodenného sedenia prináša so sebou aj výhody pre zdravie ako je prekrvenie a natiahnutie si svalov

3.4 Zavedenie vzorca na písanie user stories

Bolo navrhnuté aby sa komunikácii so zákazníkom venoval najmä Product Owner. Jednou z úloh Product Ownera je získavanie informácií od zákazníka, ako si výsledný produkt predstavuje. Jeho úlohou je požiadavky zákazníka zapísať vo forme user stories. Následne získané informácie prezentuje vývojárom. Zoznam user stories sa zapíše do backlogu.

Keďže sa vo firme doteraz neprezentovali požiadavky zákazníka vo forme user stories, nasledujúci text by mohol slúžiť ako zjednodušený návod a pravidlá na ich zapisovanie, ktoré by kontroloval a reguloval Product Owner.

Forma na získavanie user stories:

ako (typ užívateľa) – Pre KOHO to tvoríme? Kto je užívateľom?

chcem (nejaká vlastnosť) – ČO tvoríme? Aký je zámer?

takže (nejaký dôvod) – PREČO to tvoríme? Aká je hodnota pre zákazníka?

Pravidlá pre user stories:

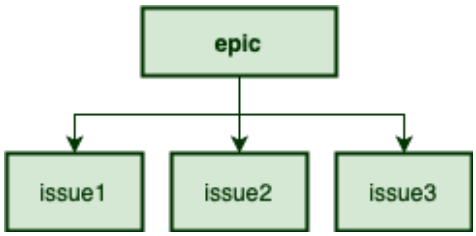
- Krátke
- Jednoduché
- Písané z perspektívy užívateľa
- Je viditeľný jej dôvod
- Opisuje jeden kus funkcionality
- Sú písané v tíme
- Nie sú to úlohy (tasky), jeden user story môže vyžadovať úspešné splnenie stoviek jednotlivých úloh
- Napísané user stories sa nikdy z backlogu nevymazávajú, len sa prioritizujú

Aplikácia Gitlab, ktorú tím používa ponúka vytvoriť epic, ktorý sa dá využiť ako user story.

Pri každom epic je možné nastaviť nasledujúce podrobnosti:

- Názov
- Popis
- Dátum začiatku
- Dátum ukončenia
- Tagy

Každému epic sa dá priradiť issue, čo znamená, že ku každej user story sa dajú priradiť tasky, ktoré sú potrebné pre jej splnenie. Týmto priradením bude mať tím prehľad o tom, aké tasky sú potrebné pre splnenie user story aj v Gitlabe, ktorý denne používa.



Obrázok 17 Schéma epicu v Gitlabe (vlastné spracovanie)

Náklady na navrhnuté riešenie

Náklady v súvislosti so zapisovaním požiadaviek vo forme user stories nevznikajú. Náklady ale vznikajú pri používaní epicu v Gilabe. Túto funkciu ponúka Gitlab s predplatným Silver a Gold. Mesačná suma za predplatné Gitlab Silver je 480 Kč na jedného užívateľa.

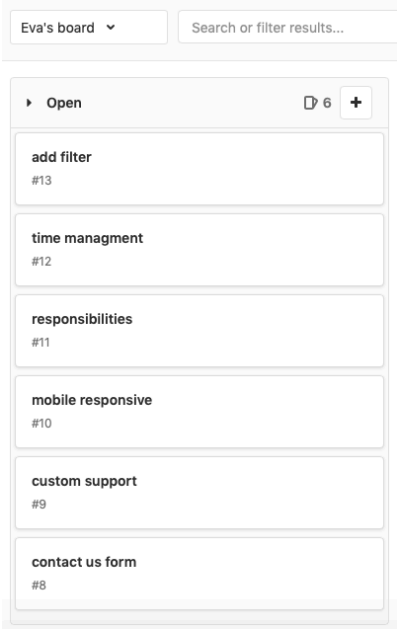
Zhrnutie prínosu riešenia

- Pochopiteľnosť pre vývojárov aj zákazníka
- Úzky kontakt zo zákazníkom
- Po ich spísaní je ich možné prioritizovať

- Transparentnosť - zákazník, Product Owner a zvyšok tímu vie, na čom sú dohodnutí a čo sa od nich požaduje
- Zvýšenie prehľadu v aplikácii GitLab

3.5 Kanban tabuľa

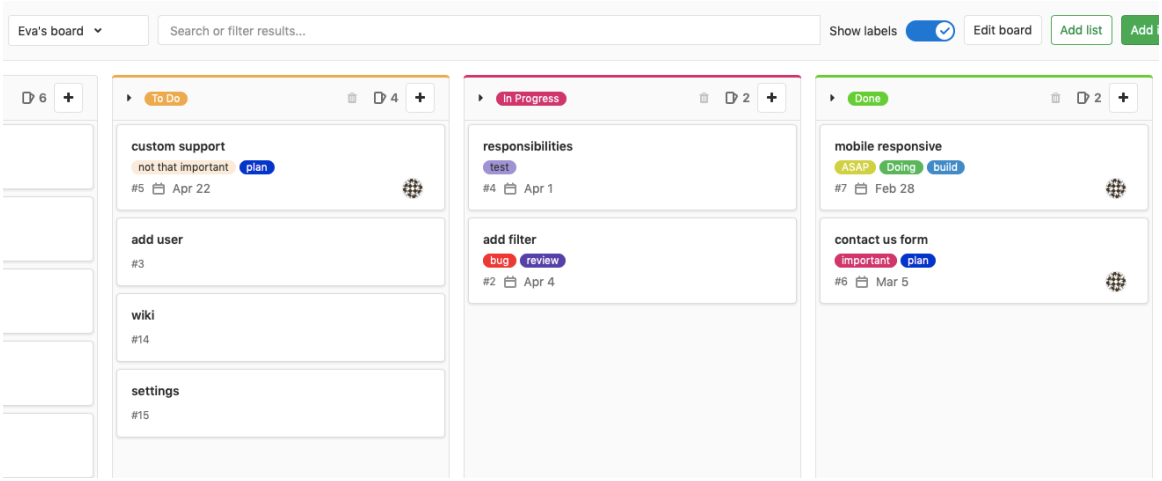
Pre zvýšenie prehľadu vedúceho tímu ale aj pre vlastný prehľad bolo navrhnuté aby každým členom tímu bola v Gitlabe vytvorená Kanban tabuľa. Na ľavej strane tabule sa by nachádzali tasky (v Gitlabe pod pojmom issues), ktoré majú byť splnené.



Obrázok 18 Produktový backlog [29]

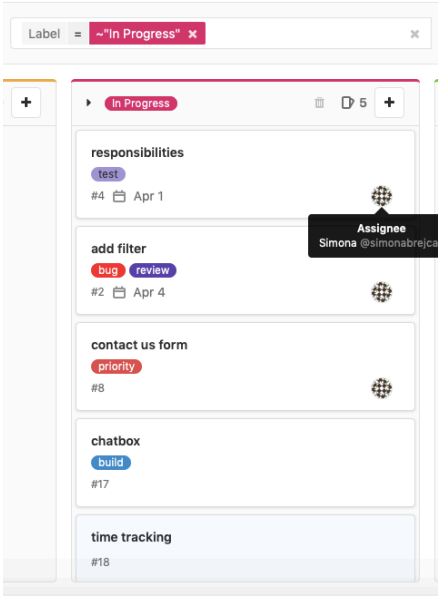
Tabuľa by bola rozdelená do troch stĺpcov – To do/ In Progress/ Done a jednotlivé tasky by sa medzi nimi mohli pohybovať. Ku každému tasku je možné vložiť požadovaný dátum dokončenia a vlastný label, ktorý odlišuje jednotlivý stav taskov. Label na dodatočné označenie tasku môže vytvoriť každý člen. Label je užívateľsky nastaviteľný, takže je možné si vytvoriť rôzne stavy ako napríklad: bug, test, čakanie na schválenie Product Ownerom.

Každý člen by mal mal tasky medzi jednotlivými stĺpcami posúvať primerane často (každý deň pred Standup) aby reflektovali skutočný stav.



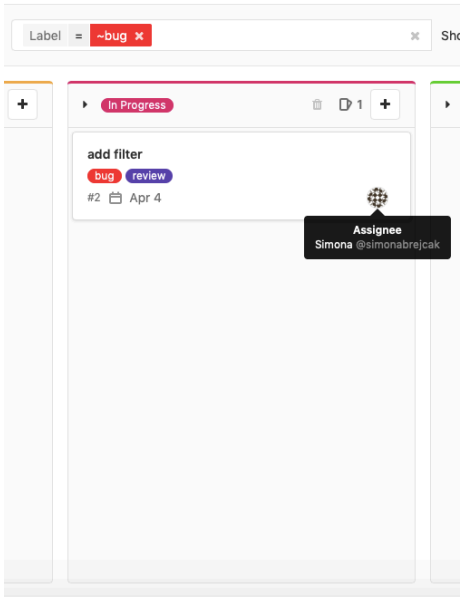
Obrázok 19 Kanban tabuľa v Gitlab [29]

Každému tasku je možné zadať stav a priradiť osobu, je teda hneď jasné čo sa s danou user story deje a kto za tento úkon práve zodpovedá Vedúci tímu si môže vyfiltrovať tabuľu každého člena, ale taktiež aj jednotlivé stĺpce Kanban tabule čím získa prehľad o tom na čom sa momentálne pracuje.



Obrázok 20 Zoznam taskov označených labelom „In Progress“ v Gitlab [29]

Taktiež je možné filtrovať aj podľa jednotlivých labelov, ktorými sú jednotlivé tasky označené.



Obrázok 21 Priradenie tasku v Gitlab [29]

Týmto jednoduchým usporiadaním sa zvýši prehľad o prebiehajúcom vývoji u vedúceho tímu, ale aj u jednotlivých členov tímu.

Náklady na navrhnuté riešenie

Náklady nevznikajú, keďže túto funkciu ponúka aj bezplatná verzia Gitlabu.

Zhrnutie prínosu riešenia

- Každý člen tímu bude mať väčší prehľad o tom, ako postupuje s dodaním priradených taskov
- Vedúci tímu zvýši svoj prehľad o tom, aká je produktivita jednotlivých členov
- Jednotlivci, ktorí nie sú súčasťou tímu sa dokážu informovať o priebehu plnenia jednotlivých taskov

3.6 Ekonomické zhodnotenie

V tejto časti sa zhrnú náklady, ktoré boli uvedené pod každým riešením. Značnú časť nákladov tvorí návšteva u zákazníka, ktorá so sebou prináša cestovné, stravné a ubytovacie náklady. Ich výška záleží od sídla zákazníka.

V nasledujúcej tabuľke sú zhrnuté náklady v prípade, že by zákazník sídlil v českom meste Ostrava a pracovná cesta by trvala 5 dní (pondelok-piatok), kedy zamestnanec prevádzku navštívi a vypracuje analýzu.

Tabuľka 3 Náhrady zamestnancovi za 5 dňovú pracovnú cestu (vlastné spracovanie)

stravné	1 030 Kč
náhrada za používanie motorového vozidla	1 436,4 Kč
cena za pohonné hmoty	652,5 Kč
ubytovanie	4 888 Kč
CELKOM	**8 006,9 Kč**

Za každý kalendárny deň pracovnej cesty, ktorá trvá dlhšie než 18 hodín zamestnávateľ zamestnancovi poskytuje stravné 206 Kč. Náhrada za používanie motorových vozidiel činí 4,2 Kč za 1 km. Vzdialenosť z Brna do Ostravy je 171 km. Priemerná cena za 1 liter motorovej nafty je 31,8 Kč. Na cestu Brno - Ostrava - Brno sa pri technickej spotrebe 6 litrov/100 km spotrebuje 20,52 litrov nafty. Cena priemerného hotela v Ostrave je 1 222 Kč na jednu noc.

V nasledujúcej tabuľke je 10 dňová cesta v Ostrave, ktorú by zamestnanec vykonal po nasadení stroja do prevádzky.

Tabuľka 4 Náhrady zamestnancovi za 10 dňovú pracovnú cestu (vlastné spracovanie)

stravné	2 060 Kč
náhrada za používanie motorového vozidla	2 872,8 Kč
cena za pohonné hmoty	1 305 Kč
ubytovanie	9 776 Kč
CELKOM	**16 013,8 Kč**

V tabuľke je rátané s tým, že zamestnanec vykoná pracovnú cestu od pondelka do piatka, na víkend sa vráti späť do Brna. Nasledujúci týždeň pondelok až piatok opäť strávi u zákazníka. Je počítané s rovnakými dennými nákladmi za stravu ako v predchádzajúcej tabuľke. Náhrada za používanie je tiež 4,2 Kč za 1 km a celkom bude vykonaných 684 km. Cena za 1 liter pohonnej hmoty sa nemení a celková spotreba bude 41,04 litrov.

Celková suma, ktorá vzniká v súvislosti s návštevou zákazníka pred a počas nasadenia stroja do prevádzky, ak sídli zákazník v Ostrave, je 24 021 Kč. Suma je orientačná a je potrebné počítať s tým, že sa môže zvýšiť (nie je počítané s nákladmi, ktoré vznikajú pohybovaním sa z hotela do prevádzky a podobne).

Používanie epic v Gitlabe, ktorý by mohol slúžiť na zapisovanie user stories, je možné len s predplatenou verziou GitLab Silver. Suma je 480 Kč za mesiac na jedného užívateľa. Suma na jeden rok pre celý tím je vyjadrený v nasledujúcej tabuľke.

Tabuľka 5 Náklady na predplatné GitLab Silver (vlastné spracovanie)

	mesiac	rok
1 užívateľ	480 Kč	5 760 Kč
9 užívateľov	4 320 Kč	51 840 Kč

Cena za predplatné pre celý tím na jeden rok činí 51 840 Kč.

V návrhoch riešení prioritizácia požiadaviek, Kanban tabuľa a pravidlá pre Standup meeting nevznikajú náklady.

3.7 Prínosy riešenia

Navrhnuté riešenia majú pozitívny dopad najmä na nasledujúce oblastí:

- **Prehľad** - Zvýšenie prehľadu o postupe plnenia úloh, získa vedúci tímu, jednotliví členovia tímu, ale aj zamestnanci firmy, ktorí do tímu nepatria.

Taktiež sa zvýši prehľad aj u zákazníka, ktorý bude informovaný o postupe tímu a vývoji jednotlivých funkcií.

- **Implementácia požadovaných vlastností** - Nastavením rutiny, v ktorej sa požiadavky prioritizujú na základe schopnosti tímu, čo dokáže za daný čas dodať, a požiadavok zákazníka sa zabezpečí, že bude dodané len to, čo zákazníkovi skutočne prinesie hodnotu.
- **Zvýšenie efektivity** - Včasné vyriešenie problému, ktorý bráni jednotlivcom v dokončení úlohy pomôže k vyššej efektivite.
- **Zlepšenie atmosféry v tíme** - Jasne stanovené postupy pre plnenie jednotlivých zákaziek a každodenné stretnutie, kde je možné vyjadriť problém, ktorý vznikol znížia stresový faktor a podporia súdržosť.
- **Užšia spolupráca so zákazníkom** - Product Owner zákazníka informuje o postupe jednotlivých požiadavok a jeho prevádzku vrámci dodania zákazky navšteví niekoľko krát, čo zvýši zákazníkovu dôveru voči spoločnosti.

3.8 Zhrnutie

Každé z navrhnutého riešenia bralo do úvahy viacero charakteristík spoločnosti. Ako prvá bola prehodnotená možnosť nasadenia metodiky Scrum, no to sa z viacerých dôvodov zamietlo. Hlavným dôvodom bol predmet podnikania a to, že spoločnosť dodáva softvér do priemyselných strojov, čo nedovoľuje dodať potencionálny produkt na konci každého šprintu.

Väčšina agilných prvkov bola vybraná z metodiky Scrum a Kanban, keďže nástroje, ktoré ponúkajú sa najviac hodia na riešenie vzniknutých problémov.

Všetky riešenia boli konzultované s vedúcim softvérového tímu, čo zabezpečilo, že návrhy je možné ihneď realizovať a pred nasadením do praxe ich nie je potrebné modifikovať.

ZÁVER

V úvode tejto bakalárskej práce bol stanovený cieľ, ktorým bolo využiť teoretické poznatky, nástroje a metódy agilného riadenia pri navrhnutí agilných nástrojov do softvérového tímu vybranej spoločnosti.

Cieľ bol následne rozdelený do troch dielčích cieľov a každý z nich predstavuje jednu z troch základných kapitol.

Prvá časť práce je venovaná teoretickým poznatkom, sú v nej popísané tradičné a agilné metodiky a ich vzájomné porovnanie. Pri popise agilných metodík je dôležité spomenúť *Manifest agilného vývoja*, ktorý v štyroch bodoch zhrňuje, čo znamená by agilný. Ďalej je táto časť práce venovaná popisu najznámejších agilných metodík a nástrojov. V práci sú spomenuté metodiky a nástroje do takej miery aby bolo pri zavádzaní riešení možno prehodnotiť viacero možností. Už len samotný prehľad dnes známych metodík a nástrojov bol pre firmu prínosom.

Analytická časť práce popisuje spoločnosť MICo Vision a jej softvérový tím na základe poskytnutých dát. Táto časť práce bola dôležitá na pochopenie aktuálneho stavu fungovania, z ktorého vyplývajú priestory na zlepšenie.

Praktická časť obsahuje vlastný návrh riešenia pre problémy, ktoré vznikajú v analytickej časti. Z analýzy súčasného stavu tímu bolo zistené, že zavádzanie celej metodiky by nebolo prínosné najmä vzhľadom k predmetu podnikania spoločnosti. Navrhnuté riešenia obsahujú implementáciu agilných prvkov, ktoré dokážu štandardizovať procesy a pomôžu pri zvyšovaní efektívnosti tímu, no zároveň to nenaruší jeho fungovanie. Cieľom bolo dodať súbor nástrojov, ktoré pomôžu byť agilnými.

Pri každom návrhu sú zhodnotené náklady (vzhľadom k charakteru riešenej problematiky sa nejedná o konkrétne sumy) a prínosy, ktoré sa jeho realizovaním dosiahnú.

Niektoré návrhy už dokázali byť realizované. Patrí medzi ne Kanban tabula a konanie StandUpu každý deň. Na zhodnotenie efektívnosti týchto riešení je však ešte príliš skoro.

V budúcnosti by bolo možné prácu rozšíriť o ďalšie návrhy, vzhľadom k tomu, že teoretická časť práce ponúka dostatok nástrojov.

ZOZNAM POUŽITÝCH ZDROJOV

1. ŠTEFÁNEK, Radoslav. *Projektové řízení pro začátečníky*. Brno: Computer Press, 2011. ISBN 978-80-251-2835-0.
2. KADLEC, Václav. Agilní programování: metodiky efektivního vývoje softwaru. Brno: Computer Press, 2004. ISBN 80-251-0342-0.
3. SMOLÍKOVÁ, Lenka. *Projektové řízení: studijní text pro prezenční a kombinovanou formu studia*. Brno: Akademické nakladatelství CERM, 2018. ISBN 978-80-214-5695-2.
4. ŠOCHOVÁ, Zuzana a Eduard KUNCE. Agilní metody řízení projektů. Brno: Computer Press, 2014. ISBN 978-80-251-4194-6.
5. MYSLÍN, Josef. Scrum: průvodce agilním vývojem softwaru. Brno: Computer Press, 2016. ISBN 978-80-251-4650-7.
6. KACVINSKÝ, Matin. Agilné metodológie riadenia vývoja softwaru [online]. Brno, 2011 [cit. 2020-01-09]. Dostupné z: http://docplayer.net/47416349-W-012345-ya.html. Diplomová práca. Masarykova univerzita.
7. MARTINŮ, Jiří a Petr ČERMÁK. METODIKY VÝVOJE SOFTWARE [online]. Olomouc: Moravská vysoká škola Olomouc, 2018 [cit.2020-01-09].
   Dostupné z: https://mvso.cz/wp-content/uploads/2018/02/Metodiky-vývoje-software-studijn%C3%AD-text.pdf
8. Diagnostika a testování elektronických systémů [online]. In: . 2012 [cit.2020-01-09].
   Dostupné z: http://www.umel.feec.vutbr.cz/bdts/index.php/embedded-systemy/vyvojove-modely
9. EWEL, Jim. Doing Agile vs Being Agile. In: AGILE MARKETING: Applying Agile to Marketing [online]. 2017 [cit. 2020-01-09]. Dostupné z: https://www.agilemarketing.net/agile-agile/
10. Doing Agile Isnt The Same As Being Agile. In: SlideShare [online]. 2010 [cit. 2020-01-09].

Dostupné z: https://www.slideshare.net/lazygolfer/doing-agile-isnt-the-same-as-being-agile

11. SCHWABER, Ken a Jeff SUTHERLAND. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game [online]. 2017 [cit. 2020-01-09]. Dostupné z: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf
12. DRUMOND, Claire. Scrum: Learn how to scrum with the best of 'em. In: ATLASSIAN: Agile Coach [online]. [cit. 2020-01-09]. Dostupné z: https://www.atlassian.com/agile/scrum
13. ŠOCHOVÁ, Zuzana. Burndown graf. In: Zuzi's blog: Agile and Lean, Scrum, Kanban, XP @ Business [online]. 2008 [cit. 2020-01-09]. Dostupné z: https://soch.cz/blog/management/agile/burndown-graf/
14. Velocity [online]. [cit. 2020-01-09]. Dostupné z: https://www.scruminc.com/velocity/
15. What is Velocity in Scrum? [online]. [cit. 2020-01-09]. Dostupné z: https://www.visual-paradigm.com/scrum/what-is-scrum-velocity/
16. About Scrum: Scrum Team [online]. [cit. 2020-01-09]. Dostupné z: https://www.scrumalliance.org/about-scrum/team
17. RIORDAN, Jeb. Scrum Process Flow. In: Youtube [online]. 2016. Dostupné z: https://www.youtube.com/watch?v=iiWMMz5nuo0
18. ANDRES, Cynthia a Kent BECK. What is XP? In: InformIT [online]. 221 River Street, Hoboken, NJ 07030: Pearson Education, 2005 [cit. 2020-01-09]. Dostupné z: http://www.informit.com/articles/article.aspx?p=367636
19. KOTAČKA, Vít. Kanban, lehký úvod [online]. 2019 [cit. 2020-01-09]. Dostupné z: https://sw-samuraj.cz/tags/kanban/
20. BECK, K. Embracing change with extreme programming. Computer [online]. (10) [cit. 2020-01-09]. DOI: 10.1109/2.796139. ISSN 00189162. Dostupné z: https://www.computer.org/csdl/magazine/co/1999/10/rx070/13rRUEgs2Pl
21. NEWKIRK, James. Introduction to agile processes and extreme programming. In: Proceedings of the 24th international conference on Software engineering - ICSE '02 [online]. New York, New York, USA: ACM Press, 2002, 2002

[cit. 2020-01-09]. DOI: 10.1145/581441.581450. ISBN 158113472X. Dostupné z: https://ieeexplore.ieee.org/abstract/document/1008034

22. Make Process Policies Explicit [online]. Kanban Zone, 2020 [cit. 2020-01-09]. Dostupné z: https://kanbanzone.com/kanban-resources/process-explicit-policies/
23. BECK, Kent. Test-driven development: by example. Boston: Addison-Wesley, c2003. ISBN 0321146530.
24. KULKARNI, Kanchan. What is Test Driven Development (TDD)? Tutorial with Example [online]. 4023 Kennett Pike #50286 Wilmington, Delaware 19807, US [cit. 2020-01-09]. Dostupné z: https://www.guru99.com/test-driven-development.html
25. WHAT IS FDD IN AGILE? [online]. 12301 Research Blvd Austin,TX 78759: Planview [cit. 2020-01-09]. Dostupné z: https://leankit.com/learn/agile/fdd-agile/
26. RADIGAN, Dan. Kanban: How the kanban methodology applies to software development. In: ATLASSIAN: Agile Coach [online]. [cit. 2020-01-09]. Dostupné z: https://www.atlassian.com/agile/kanban
27. WAYNER, Peter. Top agile tools that keep software engineers productive [online]. [cit. 2020-01-11]. Dostupné z: https://techbeacon.com/app-dev-testing/top-agile-tools-keep-software-engineers-productive
28. BOER, Gregg. What is Scrum [online]. 2017 [cit. 2020-01-11]. Dostupné z: https://docs.microsoft.com/en-us/azure/devops/learn/agile/what-is-scrum
29. *GitLab* [online]. [cit. 2020-04-12]. Dostupné z: https://about.gitlab.com
30. *Google Calendar* [online]. [cit. 2020-04-12]. Dostupné z: https://calendar.google.com
31. Rentgenová nedestruktivní kontrola - MICo Vision s.r.o.. Rentgenová nedestruktivní kontrola - MICo Vision s.r.o. [online]. Copyright © 2019 [cit. 15.04.2020]. Dostupné z: https://vision.mico.cz

ZOZNAM POUŽITÝCH SKRATIEK A SYMBOLOV

backlog	zoznam úloh
FDD	Feature Driven Development
IT	information technology
product owner	vlastník produktu
Standup	stretnutie, počas ktorého všetci účastníci stoja
tasky	úlohy
tzv.	takzvaný
TDD	Test Driven Development
user story	príbeh používateľa
viď.	viditeľne
XP	Extreme programming

ZOZNAM POUŽITÝCH OBRÁZKOV

Obrázok 1 Atributy projektu 13
Obrázok 2 Schéma vodopádového modelu životného cyklu 14
Obrázok 3 Rozdiel medzi tradičnými a agilnými prístupmi 15
Obrázok 4 Vzájomný vzťah činností v Extrémnom programovaní 21
Obrázok 5 Velocity plán 23
Obrázok 6 Project Burndown graf 24
Obrázok 7 Priebeh Scrumu 26
Obrázok 8 Fyzická Kanban tabuľa 27
Obrázok 9 Elektronická Kanban tabuľa 28
Obrázok 10 Grafické znázornenie metodiky Test Driven Development 29
Obrázok 11 Postupnosť vývojových fáz v metodike Feature Driven Development 32
Obrázok 12 Schéma vedenia spoločnosti 34
Obrázok 13 Sciox custom 34
Obrázok 14 Tabuľa v GitLabe 40
Obrázok 15 Pripomienka StandUp v kalendári 47
Obrázok 16 StandUp poradie 48
Obrázok 17 Schéma epicu v Gitlabe 50
Obrázok 18 Produktový backlog 51
Obrázok 19 Kanban tabuľa v Gitlab 52
Obrázok 20 Zoznam taskov označených labelom „In Progress“ v Gitlab 52
Obrázok 21 Priradenie tasku v Gitlab 53

ZOZNAM POUŽIÝCH TABULIEK

Tabuľka 1 Základné informácie o spoločnosti 33
Tabuľka 2 Vzor spísania user stories 45
Tabuľka 3 Náhrady zamestnancovi za 5 dňovú pracovnú cestu 54
Tabuľka 4 Náhrady zamestnancovi za 10 dňovú pracovnú cestu 54
Tabuľka 5 Náklady na predplatné GitLab Silver 55