Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra systémového inženýrství

Bakalářská práce

Agilní řízení projektu ve vybrané IT firmě

Jakub Tauš

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jakub Tauš

Informatika

Název práce

Agilní řízení projektu ve vybrané IT firmě

Název anglicky

Agile project managment in a selected company

Cíle práce

Hlavním cílem bakalářské práce je popis agilního řízení projektů ve vybrané firmě.

Další cíle jsou:

1) Popište historii agilního projektového řízení.
2) Charakterizujte agilní metodiky projektového řízení.
3) Porovnejte s klasickými metodami projektového řízení.
4) Popište fungování agilní metodiky ve vybrané firmě.
5) Definujte možná vylepšení řízení projektů ve vybrané firmě.
6) Navržené změny zhodnoťte a navrhněte další postup.

Metodika

Metodika vybrané problematiky bakalářské práce je založena na studiu odborné literatury a informačních zdrojů. Poté bude vypracován popis agilního projektového řízení. Dále bude zhodnoceno řízení projektů ve vybrané firmě a budou navržena možná vylepšení.

Doporučený rozsah práce

30-60 stránek

Klíčová slova

agilní projektové řízení, SCRUM, IT vývoj

Doporučené zdroje informací

AXELOS GLOBAL BEST PRACTICE. *PRINCE2 Agile®*. Norwich: Axelos, 2015. ISBN 978-0-11-331467-6.

DOLEŽAL, J. – KRÁTKÝ, J. *Projektový management v praxi : naučte se řídit projekty!*. Praha: Grada, 2017. ISBN 978-80-247-5693-6.

DOLEŽAL, J. – MÁCHAL, P. – LACKO, B. – SPOLEČNOST PRO PROJEKTOVÉ ŘÍZENÍ. *Projektový management podle IPMA*. Praha: Grada, 2012. ISBN 978-80-247-4275-5.

PRINCE2 Agile /. ISBN 978-0113314676.

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Petra Pavlíčková, Ph.D.

Garantující pracoviště

Katedra systémového inženýrství

Elektronicky schváleno dne 24. 11. 2021

doc. Ing. Tomáš Šubrt, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 29. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 08. 03. 2022

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Agilní řízení projektu ve vybrané IT firmě“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.03.2022 _____

Poděkování

Rád bych touto cestou poděkoval paní Ing. Petře Pavlíčkové za její dozor a vedení této práce. Zároveň bych také chtěl poděkovat svým kolegům, kteří mi ukázali jejich pohled na fungování týmu.

Agilní řízení projektu ve vybrané IT firmě

Abstrakt

Teoretická část má za úkol popsat historii agilní metodiky a jejích různých poddruhů, zároveň jsou zde nastíněny i jejich výhody a nevýhody. Mezi těmito druhy jsou popsány metodiky Scrum, Kanban, extrémní programování a metodika Crystal. I když všechny zmíněné metodiky spadají pod agilní, tak jsou různorodé a mají různé pohledy na vedení týmu. Díky tomu mají vedoucí skupin mnoho možností k vybrání metodiky, která bude pro jejich tým správná.

Praktická část popisuje vedení tří různých týmů ve vybrané IT firmě. Na základě nastudovaných teoretických materiálů následně podává jejich analýzu a možnosti, které by zlepšily fungování těchto týmů jako celku.

Klíčová slova: Agilní řízení, Scrum, Kanban, Crystal, Sprint, Stand-up, Planning,

Agile project management in a selected IT company

Abstract

The theoretical part is to describe the history of agile methodology, and the history of the different subtypes. At the same time, their advantages and disadvantages are outlined. Among its types, Scrum, Kanban, extreme programming, and Crystal methodologies are described. Even though all these methodologies fall under agile methodologies, they are diverse and have different perspectives on team leadership. As a result, group leaders have a great deal of choice to select the right methodology, right for their team.

The practical part describes the leadership of three different teams in a selected IT company. Based on the study of the theoretical material, it then provides their analysis and options that would improve their functioning as a whole.

Keywords: Agile Management, Scrum, Kanban, Crystal, Sprint, Stand-up, Planning,

Obsah

Seznam obrázků

[1] Project succes rates. Vitality Chicago [online]. [cit. 2022-03-13]. Dostupné z: https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/

[2] Skladovací plocha Kanban. Toyota Global [online]. [cit. 2022-03-13]. Dostupné z: https://www.toyota-

global.com/company/history_of_toyota/75years/text/entering_the_automotive_business/chapter1/section4/item4.html
[3] Kanban board. Digite [online]. [cit. 2022-03-13]. Dostupné z: https://www.digite.com/kanban/what-is-kanban/
[4] Jira. Atlassian [online]. [cit. 2022-03-13]. Dostupné z: https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=89541898222&campaign=9124878150&creative=542638211969&device=c&keyword=atlassian%20jira&matchtype=e&network=g&placement=&ds_kids=p51242161289&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=Cj0KCQiAybaRBhDtARIsAIEG3kld_q9Ki13lsZRJ5GC86eFjZS6K2SYK_0_Z1_Dk6jkjJZxOSPa2DTgaAnm_EALw_wcB&gclsrc=aw.ds
[5] Scrum sprint. Wrike [online]. [cit. 2022-03-13]. Dostupné z: https://www.wrike.com/scrum-guide/scrum-sprints/
[6] Metodika Crystal. Active Collab [online]. [cit. 2022-03-13]. Dostupné z: https://activecollab.com/blog/project-management/crystal-methods
[7] Tauš Jakub. Rozložení agilní části firmy. [cit. 2022-03-13].
[8] Tauš Jakub. Průběh vytvoření úkolu ve sprintu. [cit. 2022-03-13].
[9] Tauš Jakub. Tým 1. [cit. 2022-03-13].
[10] Tauš Jakub. Tým 2. [cit. 2022-03-13].
[11] Tauš Jakub. Tým 3. [cit. 2022-03-13].

Seznam použitých zkratek

Extrémní programování – XP

1 Úvod

V současné době jsou IT manažeři po celém světě pod obrovským tlakem, aby vyvíjeli a dosahovali co nejlepších výsledků v podobě aktualizovaných a nových aplikacích a softwarů, které zlepšují konečné výsledky a zároveň zvyšují profitabilitu a celkový tok peněz. Přes tyto náročné podmínky, které jsou v tomto odvětví již standardem, jsou rozpočty těchto firem a manažerů neustále snižovány. Mnoho informačně technologických společností bylo nuceno uzavřít provoz, jelikož čelily nereálným požadavkům ze strany zákazníků a nebyly schopny držet krok s rapidním tempem změn, které se v tomto odvětví udály. Vzhledem k tomu, že IT podniky, které používají tradiční metodiky, nedokáží zcela uspokojit potřeby podniků nové generace, roste tedy zájem o firmy, které používají agilní metodiky pro rychlé a flexibilní poskytování služeb při zachování kvality. Agilní metodika má oproti tradiční metodice mnoho výhod, které jsou vhodné pro všechny druhy společností, ale dají se především a nejsnadněji aplikovat pro IT podniky.

Cílem této práce je přiblížit agilní metodiku prostřednictvím osvědčených postupů od lidí, kteří se agilní metodikou zabývají každý den v praxi. Zároveň je v praktické části popsáno fungování tří různých týmů v jedné firmě, která byla vybrána z důvodu velikosti působení a zároveň díky rozmanitosti jejích IT systémů. Tyto týmy byly pozorovány a poté vyhodnoceny.

I když každý z těchto týmů působí v poměrně rozlišné oblasti IT, přesto jsou vidět mnohé jejich podobnosti díky faktu, že fungují v rámci jedné firmy a že používají agilní metodiku. A proto je na závěr navrhnuto řešení po každý z tří týmů, které by mohlo zlepšit jejich schopnost plánovat a vykonávat práci lépe než před tím.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem této práce je detailně popsat druhy agilního řízení projektu a vypracovat praktickou část na základě informací z teoretické části a praxe ve vybrané firmě. Praktická část bude popisovat tři týmy a jejich působení v agilním prostředí. Na závěr praktické části bude vypsáno možné řešení pro konkrétní týmy za účelem zlepšení jejich fungování.

2.2 Metodika

Teoretická část práce bude obsahovat analýzu a popis z odborných zdrojů v oboru agilního řízení. V praktické části bakalářské práce budou na základě poznatků z teoretické části popsány tři různé týmy a budou u nich aplikovány tyto poznatky pro vylepšení jejich funkčnosti.

3 Teoretická východiska

3.1 Historie projektového agilního řízení

V předešlých několika letech se vývojové týmy zaměřily na zrychlení doby dodání nových produktů a funkcí na trh, aby bylo možné řešit požadavky zákazníků v reálném čase. Většinou se vývojovým týmům podařilo tento cíl splnit velmi rychlým způsobem. Tato rychlost je z velké části zásluhou agilního přístupu.

Než vznikl agilní přístup, vývojové týmy (obzvláště v softwarovém, výrobním, leteckém a obranném průmyslu) identifikovaly problémy a plánovaly jejich řešení. Poté pracovaly na vývoji tohoto řešení a uváděly ho na trh v celé jeho šíři. Konkrétně většina týmů používala přístup Waterfall – metodiku vývoje, která se řídí určitou cestou, při níž týmy:[21]

1. Stanoví požadavky na projekt a rozsah práce.
2. Navrhnou produkt na základě těchto předem stanovených požadavků.
3. Vytvoří produkt.
4. Otestují produkt.
5. Opraví případné problémy zjištěné během testování.
6. Spustí hotový produkt.

Tento přístup může znít dobře, ale Waterfall vyžadoval, aby se týmy držely požadavků a rozsahu prací stanovených na samém začátku projektu a neprováděly v průběhu projektu žádné změny nebo doplňky. Dodržování tohoto pevného plánu se mohlo ukázat jako problematické, protože Waterfall upřednostňovala uvedení kompletního produktu na trh – to znamenalo, že mohlo trvat roky, než týmy dokončí daný projekt.

Během těchto let se často změnila charakteristika problému (ale požadavky projektu se nezměnily), takže plánované řešení bylo v době, kdy se konečně dostalo na trh, zastaralé. Na straně zákazníka toto zpoždění znamenalo, že kritické problémy zůstávaly nevyřešené po celé roky. I když tak bylo řešení k dispozici, problém, který mělo řešit, se pravděpodobně změnil.[15]

Na straně vývojářů tento boj znamenal, že uváděli na trh nové produkty, které ale již nebyly vhodné pro trh. V mnoha případech to také vedlo k vývojovému hřbitovu nedokončených produktů (cca 30 % skončilo a dalších 60 % bylo s problémy), protože týmy prostě raději opustily práci za pochodu, než aby dodaly zastaralý produkt. V důsledku těchto problémů můžeme Waterfall metodiku považovat za antagonistu v historii agilního vývoje.

Některé týmy zabývající se vývojem softwaru, jež byly frustrovány stávajícím stavem, začaly v 90. letech měnit svůj přístup k plánování a dodávání nových produktů.

V této době došlo k zavedení vývojových metod, jako jsou Scrum, Rapid Application Development, Extreme Programming, DSDM, Feature-Driven Development a Pragmatic Programming. I když se tyto metody liší, společným znakem všech je odlehčený model umožňující větší flexibilitu a menší režii plánování. Tyto přístupy k vývoji softwaru jsou nejstaršími metodami v historii agilního vývoje, jež nakonec vedly k tomu, co známe dnes.

Vše začalo na jaře roku 2000, kdy se v Oregonu sešla skupina 17 softwarových vývojářů, mezi nimiž byli Jim Highsmith, Bob Martin, Jon Kern, Martin Fowler, Jeff Sutherland a Ken Schwaber. Ti diskutovali o tom, jak by mohli zrychlit dobu vývoje, aby se nový software dostal na trh rychleji. Uvědomili si dvě klíčové možnosti, které by dosažení tohoto cíle umožnilo.

1. Zkrácení zpoždění přínosů pro uživatele, aby se vyřešily problémy s přizpůsobením produktu trhu a hřbitovem vývoje.
2. Rychlé získání zpětné vazby od uživatelů, která by potvrdila užitečnost nového softwaru a umožnila jej dále odpovídajícím způsobem vylepšovat.

Ačkoli toto setkání nevyústilo v agilní metodiku, jak ji známe dnes, jednalo se o zásadní milník v historii agilních metodik, neboť rychlost uvedení na trh, rychlá zpětná vazba a neustálé zlepšování jsou charakteristickými znaky agilní metodiky.

Necelý rok po tomto setkání v Oregonu došlo v historii agilního přístupu k zásadnímu posunu, když se stejná skupina 17 vývojářů sešla znovu, tentokrát v lyžařském středisku Snowbird v Utahu. Během tohoto setkání doufali, že se jim podaří dále rozvinout svůj pokrok a přistoupit ke konkrétnějšímu řešení hlavních problémů vývoje té doby.

Během tří dnů skupina vytvořila „Manifest agilního vývoje softwaru“ (také známý jako Agilní manifest). Tento manifest, který se stal skutečným zlomem v historii agilního vývoje, stanovil čtyři klíčové hodnoty:

- jednotlivci a interakce mají přednost před procesy a nástroji,
- funkční software před rozsáhlou dokumentací,
- spolupráce se zákazníkem místo vyjednávání o smlouvě,
- reakce na změny místo dodržování plánu.[15]

Tyto hodnoty představují významný průlom v historii agilní metodiky, ale skupina se u nich nezastavila. Aby svou vizi ještě více podbarvili, stanovili také 12 principů, které za těmito hodnotami stojí. Mezi tyto principy patří:

1. uspokojte zákazníky prostřednictvím včasného a průběžného dodávání softwaru,
2. přijímejte měnící se požadavky i v pozdější fázi projektu,
3. často dodávejte projekt,
4. rozbijte projektová sila,
5. budujte projekty kolem motivovaných jednotlivců,
6. nejefektivnější způsob komunikace je komunikace tváří v tvář,
7. fungující software je hlavním měřítkem pokroku,
8. udržujte udržitelné pracovní tempo,
9. neustálá dokonalost zvyšuje agilitu,
10. jednoduchost je zásadní,
11. samoorganizující se týmy vytvářejí největší hodnotu,
12. pravidelně reflektujte a upravujte svůj způsob práce, abyste zvýšili efektivitu.

Tyto čtyři hodnoty a dvanáct zásad jsou i dnes vodítkem agilní metodiky, kterou používají týmy.

Během setkání v únoru 2001 ve Snowbirdu tak urazila historie agilní metodiky dlouhou cestu, ale její vývoj byl teprve na začátku. Po tomto třídenním setkání byla skupina 17 vedoucích pracovníků připravena na další kapitolu v historii agilní metodiky: přesvědčit svět o hodnotě všeho, co stanovili v Agilním manifestu.

Aby mohli lépe šířit informace o Agilním manifestu, rozhodli se vytvořit trvalejší organizaci, díky tomu se zrodila Agile Alliance. Agile Alliance je nezisková organizace, která existuje dodnes. Cílem organizace je sdílet informace o agilní metodice, poskytovat zdroje týmům, které chtějí metodiku přijmout, a pokračovat ve vývoji tohoto přístupu s ohledem na měnící se potřeby.

Po vzniku aliance se metodika rozjela ve velkém stylu a získala si oblibu u týmů zabývajících se vývojem softwaru. Tyto týmy během vývoje často přispívaly k historii agilní metodiky, jak ji známe dnes, a zaváděly postupy, jako jsou rychlá rozhodnutí, formát „role-feature-reason" pro uživatelské příběhy, retrospektivy a každodenní schůzky (často označované jako stand-upy nebo daily-scrum).

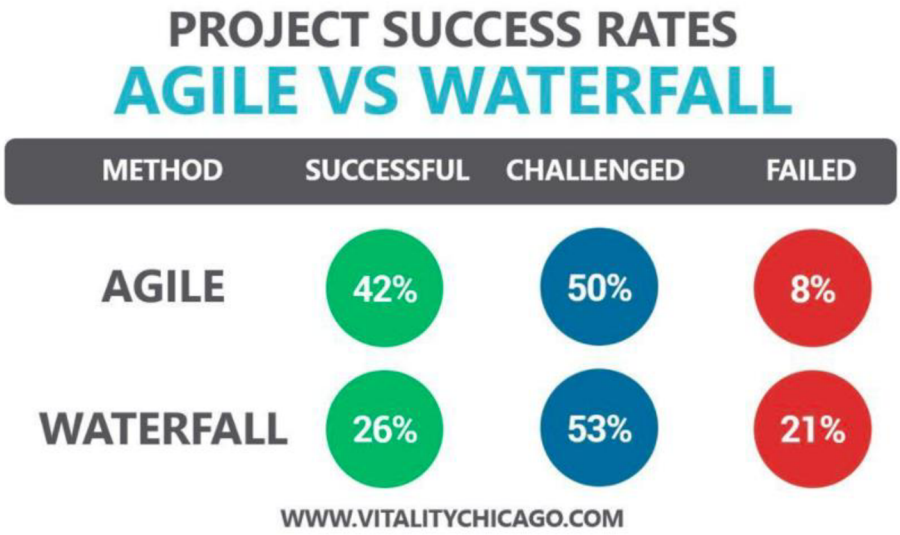
S rozmachem agilního přístupu se rozšířila i úloha aliance. V roce 2003 se nyní již formální Agile Alliance vrátila do Utahu na první výroční konferenci o agilním přístupu, což je důležitý milník v jeho historii. Akce byla podle účastníků věnována rozvoji agilních principů a poskytovala prostor pro rozkvět lidí a myšlenek. Tato výroční konference nazvaná Agile 20XX pokračuje dodnes.

V průběhu let se Agile Alliance rozšířila i geograficky, dnes podporuje přidružené skupiny po celém světě, které propagují agilní metodiku na místních trzích a pomáhají okolním organizacím přijmout manifest.

Ačkoli tato metodika zažila svůj rozkvět na počátku druhého tisíciletí, po roce 2010 jsme byli svědky toho, jak Agilní manifest nabral nový dech. Do té doby byla historie agilní metodiky běžně vyprávěným příběhem mezi vývojovými týmy, ale v letech 2012 až 2015 začaly tento příběh doprovázet reálné metriky úspěchu. V důsledku schopnosti prokázat úspěch se v té době staly výhody přijetí odlehčené metodiky nepopiratelnými. Proto není překvapením, že během tohoto tříletého období jsme také mohli zaznamenat to, jak agilní metodika překonala hranici 50 % v přijetí a skutečně vzala svět vývoje útokem.

Krátce poté začala agilní metodika expandovat, a to tentokrát tím, že se přesunula za hranice vývoje. V roce 2017 jsme se dočkali první stručné definice agilního testování. Tato definice nastínila společné testovací aktivity zaměřené na časté dodávání kvalitních produktů, které upřednostňují prevenci chyb před jejich odhalováním.

Obrázek 1 - porovnání agilní a vodopádové metodiky[1]



3.1.1 Historie Kanban

Průkopníkem metodiky Kanban byla společnost Toyota pod vedením Taiichiho Ohna, který je známý jako otec výrobního systému Toyota. Ohno si uvědomoval neefektivitu jejich výrobních linek a hledal způsoby, jak zlepšit jejich procesy. Výroba automobilu je rozhodně složitý proces; na montážní lince se pohybuje přibližně 30 000 dílů a součástek, což může být neefektivní. Ohno viděl, jak se v jejich provozu projevují zbytečné zásoby a nízká úroveň produktivity, a proto se rozhodl jednat.[13]

Ohno vzal základní myšlenku z nakupování v supermarketu a přenesl ji do výrobního procesu Toyoty. Přemýšlel o tom, jak by nakupoval potraviny. Nejčastěji by si cestu do supermarketu naplánoval až ve chvíli, kdy by mu docházelo zboží. Poznal by to po kontrole zásob doma a po zjištění, že má prázdnou spíž.

Poté by se vydal do obchodu a požadované zboží si vzal z regálů s potravinami. Když si personál supermarketu všimne, že v regálech dochází určitý výrobek, právě tehdy ho doplní. Tím je pak zajištěn dostatek zásob pro nadcházející zákazníky.

Na začátku 21. století si klíčoví hráči v softwarovém průmyslu rychle uvědomili, jak by Kanban mohl pozitivně změnit způsob, jakým se dodávají produkty a služby.

Se zvýšeným důrazem na efektivitu a s využitím pokroku ve výpočetní technice opustil Kanban oblast automobilového průmyslu a byl úspěšně aplikován v dalších komplexních komerčních odvětvích, jako je IT, vývoj softwaru, výzkum a vývoj a další.

To, co se dnes nazývá metodou Kanban, vzniklo na začátku roku 2007 a je výsledkem dlouholetého testování, zkušeností a společného úsilí předních osobností komunity Lean and Agile, jako jsou Dan Vacanti, Corey Ladas, David Aderson, Dominica DeGrandis, Rick Garber, Darren Davis a další.[12]

Obrázek 2 – Skladovací plocha Kanban[2]

3.1.2 Historie Scrum

K první úplné implementaci Scrumu došlo v roce 1993, kdy Jeff Sutherland, John Scumniotales a Jeff McKenna implementovali Scrum ve společnosti Easel Corporation.

Inspirovali se klasickým článkem HBR z roku 1986 „The New Product Development Game“, kde Takeuchi a Nonaka přirovnávali nový holistický přístup v inovacích k ragby, kde se celý tým „snaží dojít do cíle jako celek a předává si míč sem a tam“.

V roce 1995 Jeff Sutherland a Ken Schwaber společně přednesli příspěvek „The SCRUM Development Process“ na konferenci Object-Oriented Programming, Systems, Languages & Applications (OOPSLA) '95 v Austinu v Texasu, což bylo jejich první veřejné vystoupení.

V roce 2001 se Sutherland, Schwaber a patnáct dalších kolegů sešli ve Snowbirdu v Utahu a sepsali Agilní manifest, který se stal výzvou k akci pro vývojáře softwaru na celém světě, aby se snažili o radikálně odlišný způsob tvorby softwaru.

Od té doby Sutherland, Schwaber a stále rostoucí komunita lidí, kteří se Scrumu věnují, vytvořili desítky tisíc vysoce výkonných týmů ve firmách po celém světě.

Další významné příspěvky k praxi Scrumu přinesl Mike Cohn s vývojem uživatelských příběhů jako nástroje pro popis cílů práce orientovaných na klienta spolu s vývojem příběhových bodů jako způsobu měření množství práce a rychlosti týmů.

První příručka Scrum Guide byla vydána v roce 2010. Šlo o snahu tvůrců Scrumu objasnit, co Scrum je. Od té doby byl Scrum Guide několikrát znovu vydán a vylepšen. TheScrumMaster.co.uk spravuje archiv průvodce Scrumem a dalších důležitých dokumentů o Scrumu. Tato příručka je základem Scrum.org.

Scrum má dnes více než 28 let, ale jeho používání stále roste, nyní je dokonce široce používán i mimo oblast vývoje softwaru. Mění svět práce, jak ho známe, k lepšímu a jeho zastánci jsou hrdí na to, že mohu být součástí tohoto hnutí, které přináší agilitu lidem, týmům a organizacím po celém světě.[14][23]

3.1.3 Historie extrémního programování

Počátky extrémního programování sahají do 90. let, kdy ho vytvořil Kent Beck, který se později stal jedním z autorů Agilního manifestu. Vytvořil ho jako agilní přístup k vývoji softwaru popsaný určitými hodnotami, principy a vývojovými postupy.[24]

3.1.4 Historie metodiky Crystal

V roce 1991 požádala společnost IBM Alistaira Cockburna o vytvoření metodiky pro objektově orientované projekty. Po rozsáhlém výzkumu dospěl k závěru, že všechny úspěšné týmy sdílejí stejný vzor a techniky, aniž by používaly nějakou konkrétní projektovou metodiku. Následně využil svých poznatků a sestavil skupinu metodik a pojmenoval ji Crystal. Díky tomu vznikla nová metodika, která je cílená pro velikostně různé skupiny. Později tuto metodiku zdokumentoval ve své knize „Crystal Clear: A Human-Powered Methodology for Small Teams“. [17][26]

3.2 Druhy agilních metodik

Na trhu jsou k dispozici různé typy agilních metodik, které vyhovují požadavkům každého projektu. Ačkoli však existují různé agilní metodiky, vše vychází z hlavních principů Agilního manifestu.

Proto se každý rámec nebo chování, jež tyto principy adaptuje, nazývá agilní. Bez ohledu na různé typy agilních metodik, které tým implementuje, lze výhody agilní metodiky hojně využít pouze při spolupráci všech zúčastněných stran.

3.2.1 Kanban

Tento typ metodiky splňuje všech 12 různých principů agilního modelu. Jedná se o inkrementální proces, v němž je hlavním aspektem transparentnost vývoje softwaru.

Pro sledovatelnost projektu používá vývojář softwaru Kanban board, která se řídí třístupňovým procesem – To Do, In progress a Done. Tyto procesy představují stav každé práce. Používají se ke sledování veškeré práce prováděné v projektu, což poskytuje jasný obraz o postupu týmu a celkovém pracovním postupu.

David J. Anderson (průkopník v oblasti Lean/Kanban pro znalostní práci a jeden ze zakladatelů této metody) formuloval metodu Kanban jako přístup k postupné evoluční změně procesů a systémů pro organizace pracující se znalostmi. Je zaměřena na to, aby se věci dělaly, a její základy lze rozdělit do dvou typů principů a šesti postupů. [12][13]

3.2.1.1 Rozdělení Kanban

Zásady řízení změn

K hlavním zásadám řízení změn patří prolínání s již zavedenými procesy nenarušujícím způsobem nebo snaha o evoluční změny a neustálé zlepšování. Jednotlivé principy jsou blíže uvedeny níže.

Princip 1: Začněte s tím, co děláte nyní.

Kanban nabízí flexibilitu, díky níž lze metodu používat nad stávajícími pracovními postupy, systémy a procesy, aniž by se narušilo to, co je již zavedeno. Metoda uznává, že stávající procesy, role, odpovědnosti a tituly mají svou hodnotu a obecně stojí za to je zachovat. Samozřejmě upozorní na problémy, které je třeba řešit, a pomůže posoudit a naplánovat změny tak, aby jejich zavedení bylo co nejméně rušivé.

Princip 2: Dohodněte se, že budete provádět postupné evoluční změny.

Metoda Kanban je navržena tak, aby se setkala s minimálním odporem. Podporuje průběžné malé přírůstkové a evoluční změny stávajícího procesu zavedením forem spolupráce a zpětné vazby. Obecně se nedoporučují rozsáhlé změny, protože obvykle narážejí na odpor způsobený strachem nebo nejistotou.

Princip 3: Podporujte vůdčí činy na všech úrovních.

Vedení na všech úrovních vychází z každodenních poznatků a činů lidí, což má zlepšit jejich způsob práce. Ačkoli se to může zdát bezvýznamné, každý sdílený postřeh podporuje myšlení neustálého zlepšování s cílem dosáhnout optimálního výkonu na úrovni týmu, oddělení či firmy. To nemůže být aktivita na úrovni managementu.

Principy poskytování služeb

Kanban se zaměřuje na rozvoj přístupu orientovaného na služby. Vyžaduje hluboké porozumění potřebám zákazníka, vytvoření sítě služeb, kde se lidé sami organizují kolem práce, a zajištění toho, aby se systém neustále vyvíjel.

Princip 4: Zaměřte se na potřeby a očekávání zákazníka.

Poskytování hodnoty zákazníkovi by mělo být středobodem každé organizace. Pochopení potřeb a očekávání zákazníků přináší pozornost ke kvalitě poskytovaných služeb a hodnotě, kterou vytváří.

Princip 5: Řízení práce.

Řízení práce v síti služeb zajišťuje posílení schopnosti lidí organizovat se kolem práce. To umožní soustředit se na požadované výsledky bez „šumu“, který vzniká mikromanagementem lidí poskytujících služby.

Princip 6: Pravidelně revidujte síť služeb.

Jednou vyvinutý přístup orientovaný na služby vyžaduje neustálé vyhodnocování, aby se podpořila kultura služeb zákazníkům. Prostřednictvím pravidelného přezkoumávání sítě služeb a hodnocení uplatňovaných pracovních zásad podporuje Kanban zlepšování poskytovaných výsledků.[19]

3.2.1.2 Výhody Kanban

Zvýšená viditelnost toku

Základní myšlenkou Kanban je vizualizace každé práce. Tímto způsobem se Kanban board mění v centrální informační centrum a všichni jsou na stejné straně. Všechny úkoly jsou viditelné a nikdy se neztratí, což přináší transparentnost celého pracovního procesu. Každý člen týmu může mít rychlý přehled o stavu každého projektu nebo úkolu.

Zlepšení rychlosti dodávek

Kanban nabízí projektovým manažerům několik způsobů, jak pečlivě sledovat a provádět informované analýzy rozložení práce. Díky jasnému přehledu nad pracovními položkami dokončenými za určité období, a fázemi, kde úkoly tráví nejdelší dobu, lze snadno identifikovat kritická místa. Týmy mají možnost tyto problémy řešit, zlepšit svůj pracovní postup a v konečném důsledku i rychlost dodávek.

Sladění mezi obchodními cíli a jejich realizací

Díky podpoře transparentnosti a zpětné vazby i pravidelným kontrolním schůzkám umožňují postupy Kanban sladit strategické cíle společnosti s každodenní prací týmů. Toto sladění mezi obchodním směrem a realizací zvyšuje agilitu organizace a umožňuje týmům přizpůsobit se měnícím se prioritám a reorganizacím v důsledku změn na trhu nebo požadavků zákazníků.

Lepší předvídatelnost

Jakmile si vytvoříte tabuli Kanban a začnete na ní hromadit pracovní položky, budete schopni porozumět svému procesu do hloubky pomocí metrik toku. Analýza času, který úkoly zaberou ve vašem pracovním postupu (doba cyklu), vám umožní zlepšit předpovědi toho, kolik práce můžete v budoucnu dodat. Pochopení konzistence rychlosti dodávek (průchodnosti) zpřesní vaše předpovědi a vaše rozhodnutí budou vycházet z historických dat.

Lepší schopnost řídit rozsah a závislosti

Vizualizaci vlastní praxe Kanban se uplatňuje také při mapování a řízení závislostí. Začít s tím, co děláte nyní, znamená vizualizovat současné závislosti a řídit toky mezi nimi. Správa závislostí poskytuje jak náhled na současný stav pracovního postupu, tak nápady na zlepšení. Na druhou stranu také umožňuje plnou transparentnost pro strategické řízení nad pracovním postupem a existujícími vazbami mezi týmy.

Větší spokojenost zákazníků

Původ metody Kanban – pull systém, na kterém je založena, předpokládá, že práce se provádí, když existuje poptávka. Jinými slovy nás Kanban naviguje ke snížení plýtvání tím, že se pracuje výhradně na úkolech, které jsou aktuálně potřeba. Použitím vizualizačních technik a zavedením limitů rozpracovanosti do procesu navíc zjistíme, že konečný výsledek bude vyladěn podle očekávání zákazníka.

3.2.1.3 Kanban board

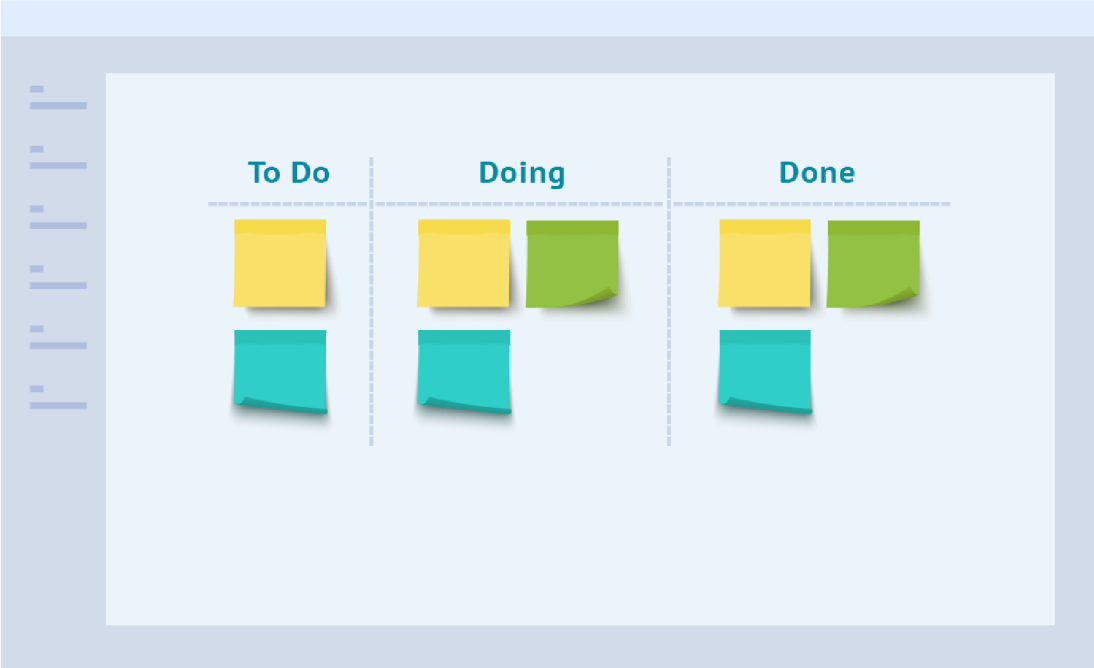
Klíčovým vizualizačním nástrojem pro zavedení systému Kanban ve firmě je Kanban board. Tabuli Kanban používá každý tým, který používá Kanban pro vizuální řízení své práce a zlepšování dodávek produktů a služeb z hlediska předvídatelnosti, kvality a časové náročnosti.

Kanban board může být fyzická nebo digitální. Typicky se skládá z jedné nebo více drah a několika sloupců, které znázorňují pracovní proces (označovaný také jako Value Stream), k jehož řízení se používá. Kanban board představuje „virtuální systém Kanban“, který se používá k modelování procesu a sledování znalostní práce prováděné týmem.

Základní Kanban board se dělí do tří sloupců (To Do, In progress a Done). Tyto sloupce jsou naplněné kartami, které představují pracovní úlohy různých typů. Podle zásady Kanban, jež zní „začni s tím, co máš“, však většina týmů začíná modelováním svého současného procesu, který bývá propracovanější. Vývojářská Kanban board se například dělí na Backlog, To Do, Development, Testing a Done.

Sloupec Backlog slouží pro úlohy, které ještě nebyly započaty a ani nebyly zařazeny do sprintu, To Do slouží pro úlohy, které jsou zařazené do sprintu, ale pracovníci na nich ještě nezačali pracovat, Development je pro úlohy, na kterých se aktuálně pracuje, Testing je na již zhotovené úlohy, jež potřebují otestovat, pokud testem úloha projde, posune se dál do Done.[8]

Obrázek 3 – Kanban board[3]



[3]

3.2.1.4 Kanban software

Pro výběr softwaru pro Kanban je důležité se ptát na několik otázek. Tyto otázky mohou být například:

- Umožňuje vám design softwaru přehledně vizualizovat pracovní postup tak, aby vám co nejlépe vyhovoval?
- Je dostatečně flexibilní, abyste si mohli pracovní postupy přizpůsobit tak, jak to vyhovuje vašim potřebám?
- Jak software podporuje implementaci zpětnovazebních smyček (komentáře, integrace e-mailů, @označování)?

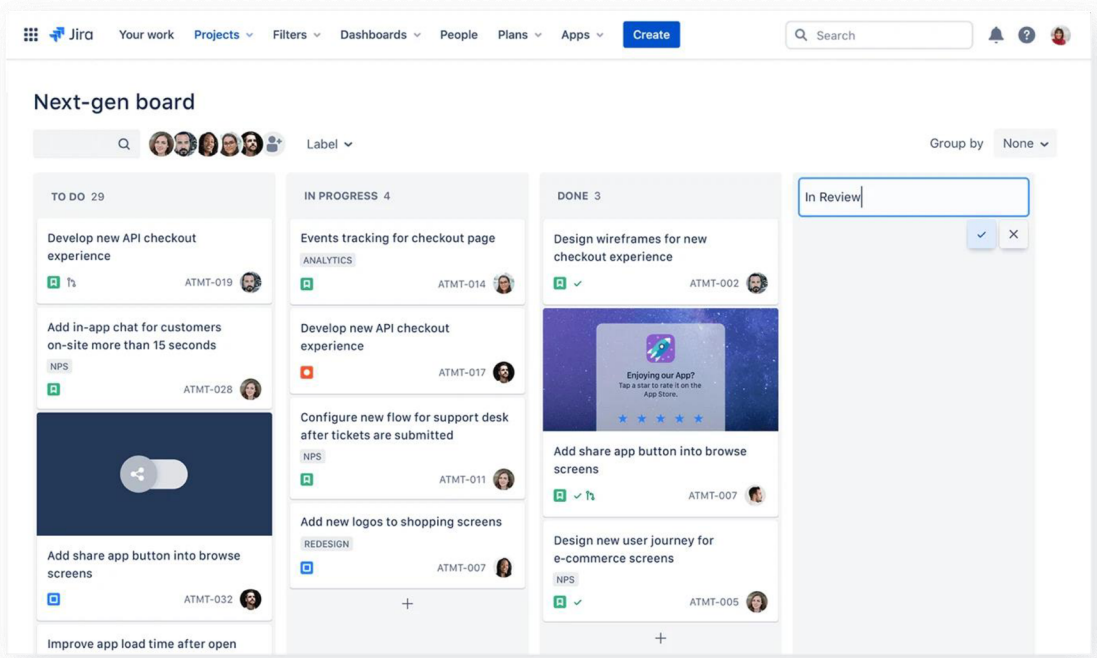
1. **Kanbanize**

Kanbanize je výkonný software pro správu projektů a portfolia Kanban. Platforma je vybavena mnoha funkcemi, které umožňují plánovat, organizovat a sledovat práci na úrovni týmu a portfolia. Lze využívat výkonný analytický modul a automatizovat procesy díky mechanismu obchodních pravidel. Software je mimořádně flexibilní a podporuje téměř všechny případy použití, navíc se integruje s některými z nejužitečnějších digitálních nástrojů, jako jsou Power BI, Microsoft Teams, Slack, Google Drive, GitHub, Gitlab a další.

2. Jira

S aplikací Jira je možné snadno organizovat a sledovat úkoly. Pomocí softwaru lze zaznamenávat problémy, přidělovat práci a sledovat činnost týmu. Pomocí aplikace Jira se také dá vytvářet různé přehledy, většina z nich je však zaměřena na Scrum (například Burndown grafy, Burn Up grafy, sprintové reporty atd.). I když je jejich počet omezený, lze také použít několik pravidel pro zefektivnění pracovního postupu týmu a automatizaci některých činností. [4]

Obrázek 4 – Jira[4]



1. Asana

Jedná se o flexibilní softwarové řešení pro správu pracovních postupů. Pracovní postupy na nástěnce Kanban je možné přizpůsobit podle konkrétních potřeb týmu a sledovat, co je nejdůležitější. Tým může také vidět aktualizace v reálném čase, a dokonce lze vytvářet závislosti úkolů. Asana se navíc integruje s různými užitečnými digitálními nástroji.[3]

3.2.2 Scrum

Scrum je framework, která pomáhá týmům spolupracovat. Podobně jako ragbyový tým trénuje na velký zápas, Scrum podporuje týmy, aby se učily prostřednictvím zkušeností, samoorganizovaly se při práci na problému a reflektovaly své výhry a prohry, což vede k neustálému zlepšování se.

I když se Scrum nejčastěji používá v týmech zabývajících se vývojem softwaru, jeho principy a poznatky lze aplikovat na všechny druhy týmové práce. To je jeden z důvodů, proč je Scrum tak populární. Scrum je často považován za rámec agilního řízení projektů a popisuje obecně soubor schůzek, nástrojů a rolí, které ve vzájemné souhře pomáhají týmům strukturovat a řídit jejich práci.[6][23]

3.2.2.1 Scrum hodnoty

Pěti hodnotami Scrumu jsou odhodlání, soustředění, otevřenost, respekt a odvaha. Scrum byl navržen speciálně pro řízení komplexních projektů, které se musí umět rychle přizpůsobit změnám v rozsahu nebo požadavcích. Proto je každá z těchto pěti hodnot Scrumu tak důležitá pro úspěch projektu Scrum.

3.2.2.1.1 Odhodlání

Aby každý sprint přinesl co nejvíce, musí se každý člen týmu soustředit na svůj úkol a na to, jak ovlivňuje cíl sprintu.

Aby členové týmu zůstali soustředění, může Scrum master omezit počet úkolů nebo priorit, které jsou na každého člověka během sprintu kladeny. Kromě toho může podpora plné účasti týmu na stand-upu pomoci jednotlivcům soustředit se na zadané úkoly.

3.2.2.1.2 Otevřenost

Aby Scrum tým dosáhl co největšího pokroku v co nejkratším čase, musí být každý člen týmu brutálně upřímný a otevřený ohledně svého vlastního pokroku. Účelem každodenních schůzek Scrumu je identifikovat a řešit problémy.

3.2.2.1.3 Respekt

Respekt v týmu Scrum znamená uznat, že žádný jednotlivec nebo jeho příspěvek není cennější než jiný. Respekt také znamená důvěřovat svým kolegům v týmu, že splní své úkoly, naslouchat jejich nápadům a uvažovat o nich a oceňovat jejich úspěchy.

3.2.2.1.4 Odvaha

Týmy Scrumu musí mít odvahu být upřímné, otevřené a transparentní jak k sobě, tak k zainteresovaným stranám, pokud jde o pokrok v projektu a případné překážky. Členové týmu musí mít také odvahu požádat o pomoc, když je potřeba.

3.2.2.2 Definiton of Done

Ačkoli je Definition of Done velmi jednoduchým artefaktem Scrum, mnoho týmů s ním bojuje. Je to v podstatě definice toho, jaké úkoly je třeba udělat, aby byly považovány za dokončené. Je to v podstatě seznam, který definuje kvalitu, kterou musí splňovat všechny úkoly na konci sprintu, abychom je mohli považovat za dokončené a zobrazit je v přehledu sprintu. Pokud některé položky Backlogu nesplňují všechny části Definition of Done, jsou neúplné a měly by být vráceny do Product Backlogu.[27]

3.2.2.3 Scrum Artefatky

Artefakty Scrumu představují práci nebo hodnotu. Jsou navrženy tak, aby maximalizovaly transparentnost klíčových informací. Každý, kdo je kontroluje, tak má stejný základ pro přizpůsobení. Každý artefakt obsahuje závazek zajistit, aby poskytoval informace, které zvyšují transparentnost a zaměření, podle kterého lze měřit pokrok:

- Pro produktový backlog je to produktový cíl.
- Pro Sprint Backlog je to Sprint Cíl.
- Pro přírůstek je to Definice hotového.

Tyto závazky existují proto, aby posílily hodnoty Scrumu pro Scrum tým a jeho členy.

3.2.2.4 Transparentnost

Vznikající proces a práce musí být viditelné pro ty, kteří práci vykonávají, i pro ty, kteří ji přijímají. práce. Ve Scrumu jsou důležitá rozhodnutí založena na vnímání stavu jeho tří formálních artefaktů. Artefakty s nízkou transparentností mohou vést k rozhodnutím, která snižují hodnotu a zvyšují riziko. Transparentnost umožňuje kontrolu. Inspekce bez transparentnosti je zavádějící a zbytečná.

3.2.2.5 Inspekce

Artefakty Scrumu a pokrok při dosahování dohodnutých cílů je třeba často a pečlivě kontrolovat, aby bylo možné odhalit potenciálně nežádoucí odchylky nebo problémy. Inspekce umožňuje přizpůsobení. Inspekce bez adaptace je považována za zbytečnou. Události Scrumu jsou navrženy tak, aby vyvolaly změnu.

3.2.2.6 Adaptace

Pokud se některý z aspektů procesu odchyluje mimo přijatelné meze nebo pokud je výsledný produkt nepřijatelný, je třeba upravit používaný proces. Úprava se musí provést co

nejdříve, aby se minimalizovaly další odchylky. Od Scrum týmu se očekává, že se přizpůsobí v okamžiku, kdy se prostřednictvím kontroly dozví něco nového.

3.2.2.7 Sprinty

Sprint je krátké, časově ohraničené období (1-4 týdny), při kterém tým Scrum pracuje na dokončení určitého množství práce. Sprinty jsou jádrem metodiky Scrum a agilních metodik, jejich správné zvládnutí pomůže agilnímu týmu dodávat lepší software s menšími problémy.

Pro mnoho lidí jsou Scrum sprinty spojeny s agilním vývojem softwaru natolik, že se Scrum a agilní metodika často považují za jedno a to samé. Avšak není tomu tak. Agilní metodika je soubor principů a Scrum je rámec pro to, jak to udělat.

Velké množství podobností mezi agilními hodnotami a procesy Scrum vede ke spravedlivé asociaci. Sprinty pomáhají týmům dodržovat agilní zásadu „častého dodávání funkčního softwaru“ a také žít agilní hodnotu „reagovat na změny namísto dodržování plánu“. Scrumovské hodnoty transparentnosti, kontroly a přizpůsobení doplňují agilní postupy a jsou ústředním prvkem konceptu sprintů. Příručka Scrum Guide vytváří solidní teoretický základ pro tuto diskusi o sprintech.

K naplánování nadcházejícího sprintu slouží schůzka pro plánování sprintu. Plánování sprintu je společná akce, při níž tým odpovídá na dvě základní otázky (Jaká práce může být v tomto sprintu provedena a jak bude vybraná práce provedena?).

Výběr správných pracovních položek pro sprint je společným úsilím product ownera, Scrum Mastera a vývojového týmu. Product owner diskutuje o cíli, kterého by mělo být ve sprintu dosaženo, a o položkách produktového backlogu, jejichž dokončením by bylo cíle sprintu dosaženo.

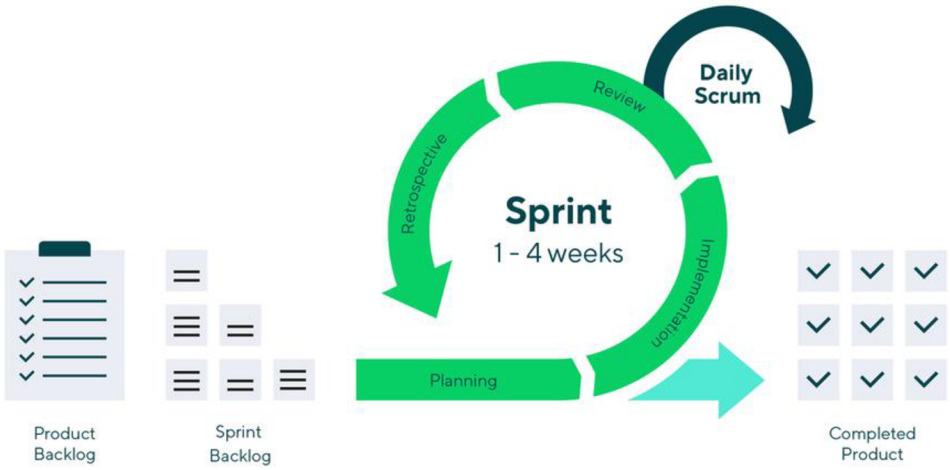
Tým pak vytvoří plán, jakým způsobem bude položky backlogu vytvářet a jak je dostane do „Done“ před koncem sprintu. Vybrané položky práce a plán, jak je dokončit, se nazývají sprintový backlog. Na konci plánování sprintu je tým připraven začít pracovat na backlogu sprintu a přebírat položky z backlogu.

Při sprintu tým během každodenního Scrumu neboli stand-upu kontroluje, jak práce pokračuje. Cílem této schůzky je odhalit všechny překážky a problémy, které by mohly ovlivnit schopnost týmů dosáhnout cíle sprintu. Dá se také označit jako mini plánovací meeting.

Po skončení sprintu tým předvede, co dokončil během sprint review. Je to příležitost pro tým předvést svou práci zúčastněným stranám a hlavně zákazníkovi, aby mohl získat zpětnou vazbu a zapracovat ji do projektu ještě dříve, než se dostane do výroby.

Sprintový cyklus je zakončen autorovým oblíbeným setkáním – sprintovou retrospektivou. Je to příležitost pro týmy identifikovat oblasti, které je třeba zlepšit pro příští sprint.[26]

Obrázek 5 – Scrum sprint[5]



3.2.2.8 Role ve Scrum

Scrum tým se skládá ze tří různých rolí: Scrum Mastera, Product Ownera a členů Scrum týmu. Zatímco Scrum Master i Product Owner je vždy jen jeden, členů Scrum týmu je obvykle více. Týmy jsou ale relativně malé (doporučuje se maximálně deset členů), aby byla zajištěna optimální komunikace a produktivita.[6]

3.2.2.8.1 Scrum Master

Scrum Master je zodpovědný za efektivitu týmu. Toho dosahuje tím, že umožňuje týmu zlepšovat postupy v rámci Scrumu. Scrum Masteři jsou skuteční lídři, kteří slouží svému týmu i celé organizaci.

Ačkoli je Scrum Master důležitým členem týmu řízení projektu Scrum, nepodílí se na agilním plánování. To provádí Product Owner a Scrum tým. Scrum Master nepůsobí jako projektový manažer (tým se organizuje sám), ve skutečnosti ani není zodpovědný za úspěch výsledku projektu.

Přesto by se ale bez Scrum Mastera celý Scrum framework rozpadl. Scrum Master je tmel, který drží projekt pohromadě tím, že usnadňuje každodenní stand-upy (i když se jich většinou neúčastní, tak kontroluje, že tým měl stand-up). Dále pomáhá týmu udržovat burndown chart a organizovat retrospektivy, sprint review a agilní sprint planning.

Scrum Master má v projektu několik rolí a odpovědností. Jedním ze způsobů, jak se na něj může dívat, je jako na servant leadera. Není součástí hierarchie, místo toho zaujímá komplexnější přístup k práci na vývoji produktu nebo softwaru, nabízí ostatním své znalosti o řízení projektů Scrum a zároveň podporuje smysl pro komunitu a sdílenou rozhodovací pravomoc uvnitř týmu.[25]

Níže jsou zmíněné některé z hlavních rolí Scrum Mastera.

- Scrum Master slouží Product Ownerovi tím, že zajišťuje, aby byly cíle, rozsah a doména produktu jasné všem členům Scrum týmu.
- Další velkou rolí, kterou hraje, je neustálé předávání informací zúčastněným stranám projektu o tom, jak si stojí aktuální agilní sprint, a úsilí o vývoj produktu nebo softwaru. To lze provádět prostřednictvím různých scrumových artefaktů (tj. od produktových backlogů a scrumových schůzek až po burndown grafy) a prostřednictvím běžných komunikačních snah v oblasti řízení projektu.
- Scrum Masteři také znají plánování projektů v empirickém prostředí, jsou skvělí v plánování agilních sprintů a dokážou vést agilní týmy. Jsou zodpovědní za pořádání Scrum schůzek podle potřeby, aby řídili nebo předávali informace o procesu.

Pokud jde o povinnosti Scrum Mastera, jeho hlavním cílem je působit jako agilní kouč, který pomáhá Scrum týmům se samoorganizovat a pracovat napříč funkcemi, aby lépe spravovaly svůj produktový backlog a maximalizovaly svou efektivitu.

Hlavní povinnosti Scrum Mastera jsou následující.

- Scrum Master pomáhá týmu vytvářet vysoce hodnotný produkt tím, že odstraňuje překážky v jeho procesu a koučuje ho na každodenních schůzkách nebo na jiných místech, kde je potřeba pomoci.
- Scrum tým by měl mít možnost spolehnout se na to, že Scrum Master před ním uvolní cestu tím, že odstraní překážky nebo určí priority úkolů To jim umožní soustředit se na úkoly, které mají aktuálně na starosti, aby je zvládli co nejefektivněji.
- V neposlední řadě pomáhá organizaci také tím, že vede a koučuje přechod na scrumový framework. V této funkci bude Scrum Master vést změny, které zvýší produktivitu týmu, zároveň bude spolupracovat s ostatními Scrum Mastery a Product Owners, aby pomohl podpořit používání metodiky Scrum v celé organizaci.

3.2.2.8.2 Product Owner

Product Owner je osoba v týmu, která je zodpovědná za správu backlogu práce a dohled nad sprinty. Product Owner je zodpovědný za zastupování hlasu zákazníka a zajišťuje, aby veškerá vývojová práce odrážela potřeby koncového uživatele.

Pravdou je, že společnosti využívají Product Owners různými způsoby a jejich role se může měnit v závislosti na tom, co daná společnost potřebuje. Některé společnosti pohlížejí na Product Owner jako na taktického a praktického manažera práce, jiné ho mohou využívat volněji, aby reprezentoval vizi produktu.

Ve finále jsou pak jeho povinnosti definovány společností a týmem, ve kterém pracuje.

Product Owner je také zodpovědný za efektivní správu Product Backlogu, která zahrnuje:

- vypracování a výslovné sdělování cíle produktu,
- vytváření a jasné sdělování položek produktového zásobníku,
- objednávání položek produktového zásobníku,
- zajištění transparentnosti, viditelnosti a srozumitelnosti seznamu produktů.

3.2.2.8.3 Členové Scrum týmů

I když se to také nazývá development tým, tak i přesto mohou být ve skutečnosti všechny role (například i test nebo analýza). Ideálně ve Scrumu, ale role nerozlišujeme a nejagilnější týmy mají členy, kteří dokáží zastat všechny role od začátku do konce.

Konkrétní dovednosti, které členové týmu potřebují, jsou často široké a liší se podle oblasti práce.

Členové týmu jsou však vždy zodpovědní za:

- vytvoření plánu pro sprint, sprint backlog,
- zavádění kvality dodržováním definice hotového,
- přizpůsobování svého plánu cíli sprintu každý den,
- vzájemné zodpovídaní se jako profesionálové.

3.2.2.8.4 Product Owner vs. Scrum Master

Ačkoli Scrum Master a Product Owner úzce spolupracují, jejich role jsou velmi odlišné. Scrum Master vede agilní vývojový tým a podporuje Product Ownera tím, že předává aktuality příslušným zaměstnancům. Product Owner spravuje produktový backlog a zajišťuje, aby společnost získala z produktu maximální hodnotu.

Produktový backlog je klíčovou součástí agilního vývoje produktů. Obsahuje všechny úkoly, které je třeba během projektu dokončit. Product Owner musí tento zdroj snadno zpřístupnit Scrum Masterovi i týmu a zajistit, aby přesně odrážel potřeby zákazníka, podniku a všech dalších relevantních zainteresovaných stran.

Za tímto účelem musí Product Owneři plnit pro vývojový tým více rolí. Působí jako návrháři produktu, styční pracovníci se zákazníky a obchodní stratégové, aby získali úplné pochopení účelu produktu.

Jakmile Product Owner zdokonalí plán produktu, je na Scrum Masterovi, aby tuto vizi uvedl k životu. Pokud se backlog v průběhu realizace změní, Scrum Master o aktualizaci informuje vývojový tým. V tom je rozdíl oproti Kanban, protože v Kanban to nástroje udělají místo Scrum Mastera.[7]

3.2.2.9 Scrum poker

Scrum poker je druh planningu, kde všichni členové se domlouvají na hodnotě úkolu do té doby, než se shodnou.

3.2.3 Extrémní programování

Extrémní programování, které se také označuje jako XP, je rámec agilního vývoje softwaru, jehož cílem je vyšší kvalita softwaru a vyšší kvalita života vývojového týmu. XP je nejkonkrétnější z agilních rámců, pokud jde o vhodné inženýrské postupy pro vývoj softwaru.

Na rozdíl od jiných agilních metodik je extrémní programování velmi názorově vyhraněné, co se týče inženýrských postupů. Kromě postupů je extrémní programování postaveno na hodnotách a principech.

Hodnoty dávají týmům smysl, jsou však abstraktní a příliš mlhavé pro konkrétní vedení. Když například řeknete, že si ceníte komunikace, může to vést k mnoha různým výsledkům.

Postupy jsou v některých ohledech opakem hodnot. Jsou konkrétní a přízemní, definují konkrétní kroky, které je třeba udělat. Postupy pomáhají týmům udržet si odpovědnost za dodržování hodnot. Například praxe „Informativní pracovní prostory“ upřednostňuje transparentní a jednoduchou komunikaci.

Principy jsou pokyny specifické pro danou oblast, které překlenují mezeru mezi postupy a hodnotami.[1] [2][16]

3.2.3.1 Hodnoty

Existuje pět základních hodnot pro extrémní programování, a to: komunikace, jednoduchost, zpětná vazba, odvaha a respekt.[24]

3.2.3.1.1 Komunikace

Komunikace je klíčovým faktorem pro úspěch každého projektu. Důležitá je komunikace mezi týmem XP a zákazníky, což umožní hladkou spolupráci mezi členy týmu. Osobní rozhovor mezi členy týmu je například mnohem efektivnější než sezení a posílání e-mailů sem a tam.

3.2.3.1.2 Jednoduchost

Klíčem k vytvoření a udržení dobrého softwaru je jednoduchost. Ani při programování malé webové aplikace není možné kdykoli prověřit všechny části softwaru. Místo toho začneme software kontrolovat a snažíme se zjistit chování aplikace krok za krokem. Bod v programu, kde se mísí nesouvisející zájmy, „doplňuje“ celý program, tím ho činí složitějším. Když se pokusíme tato místa v programu odstranit nebo minimalizovat, zvýšíme jednoduchost aplikace a usnadníme její budoucí údržbu.

3.2.3.1.3 Zpětná vazba

V systému extrémního programování máme kratší iteraci (obvykle 1–3 týdny), což dává příležitost přezkoumat produkt a získat zpětnou vazbu od uživatelů.

Rychlá zpětná vazba od obchodních partnerů a koncových uživatelů udržuje tým pro zlepšování zaměřený na určené cíle řešení a pomáhá zajistit, aby dodával vysoce hodnotné funkce. Probíhá kontinuální sestavování, integrace a spouštění automatizovaných testovacích případů. Řídí se také konceptem fail-fast, který funguje obzvlášť dobře, když se požadavky vyvíjejí.

3.2.3.1.3.1 Fail-fast

Systém fail-fast na svém rozhraní okamžitě hlásí jakýkoli stav, který by mohl znamenat poruchu. Fail-fast systémy jsou obvykle navrženy tak, aby zastavily normální provoz místo toho, aby se pokoušely pokračovat v případně chybném procesu. Takové konstrukce často kontrolují stav systému v několika bodech operace, takže případné selhání lze odhalit včas. Odpovědností systému fail-fast je detekovat chyby a poté je nechat vyřešit.[11][22]

3.2.3.1.4 Odvaha

Tým pracuje společně a je product ownerem, takže se cítí podporován, má odvahu být otevřený a je pro něj výzvou jít za hranice svých možností.

Tým má odvahu říct, na co se v kódu a designu zaměřit, co odpovídá dnešním obchodním potřebám, a nepřemýšlí ve funkcích daleko dopředu.

3.2.3.1.5 Respekt

Respekt znamená, že při společné práci, sdílení úspěchů i neúspěchů respektujeme jeden druhého, naše dohody a závazky.

Respekt je klíčovou hodnotou při spolupráci týmů, které respektují odlišnosti a různorodost. Týmy mají kolektivní vlastnictví kódu, návrhu a architektury systému, a proto každý nese odpovědnost za návrh, kód, nasazení a sestavení nebo za absolvování regrese.

3.2.3.2 Principy

Existuje také mnoho základních principů, které poskytují konkrétnější vodítko než hodnoty. Jsou to pokyny, jež hodnoty objasňují a činí je jednoznačnějšími a méně nejasnými.

Mezi principy patří: lidskost, ekonomika, vzájemná prospěšnost, sobě podobnost, rozmanitost, reflexe, průběh, příležitost, redundance a dětské krůčky.[18][20]

3.2.3.2.1 Lidskost

Lidé vytvářejí software pro lidi, což je často opomíjená skutečnost. Zohlednění základních lidských potřeb i silných a slabých stránek však vytváří produkty, které lidé chtějí používat. Pracovní prostředí, jež dává příležitost k úspěchu a růstu, pocit sounáležitosti a základní bezpečí, je místem, kde se snáze zohlední potřeby druhých.

3.2.3.2.2 Ekonomika

V XP týmy neustále dbají na ekonomickou realitu vývoje softwaru, neustále vyhodnocují ekonomická rizika a potřeby projektu.

3.2.3.2.3 Vzájemná prospěšnost

Podle XP se snažíme vyhnout řešením, která přinášejí prospěch jedné straně na úkor druhé. Například rozsáhlé specifikace mohou někomu jinému pomoci v pochopení, ale nás to připraví o implementaci a uživatele to zdrží.

Vzájemně prospěšným řešením je použití automatizovaných akceptačních testů, kdy jedna strana získá okamžitou zpětnou vazbu k implementaci, druhá pak přesné specifikace v kódu, díky čemuž uživatelé dostanou své funkce dříve. Navíc všichni získají záchrannou síť proti možným regresím.

3.2.3.2.4 Sobě podobnost

Pokud určité řešení funguje na určité úrovni, může fungovat i na úrovni vyšší nebo nižší. Získávání včasné a stálé zpětné vazby je například v XP důležité na různých úrovních.

3.2.3.2.5 Rozmanitost

V rámci tohoto principu je možné dosáhnout rozmanitosti, a to jak v oblasti vývoje, tak i v oblasti zlepšování. Všichni spolupracovníci těží z rozmanitosti pohledů, dovedností a postojů. Taková rozmanitost může často vést ke konfliktům, ale to je v pořádku. Konflikty a neshody jsou příležitostí pro vznik lepších nápadů, pokud všichni hrají podle hodnot odvahy a respektu. Odvaha k vyjádření opačných názorů i respekt k jejich zdvořilému a empatickému vyjádření je zároveň cvičením v efektivní komunikaci.

3.2.3.2.6 Reflexe

Skvělé týmy reflektují svou práci a analyzují, jak být lepší. XP k tomu nabízí spoustu příležitostí, a to nejen ve svých týdenních a čtvrtletních cyklech, ale v každé praxi, kterou podporuje. Vedle logické analýzy je důležité zohlednit i pocity. Instinkt může informovat dříve, než o něčem může člověk uvažovat. Stejně tak člověk může mluvit s netechnickými lidmi, ti mohou klást otázky, které otevírají zcela nové možnosti.

3.2.3.2.7 Průběh

Termín průběh ve vývoji softwaru znamená poskytování nepřetržitého toku hodnot. Je to protiklad k dodávkám s velkým třeskem (to znamená, že ničeho nějak vznikla aplikace). Kontinuální průběh může být dodáván každý týden, den, nebo dokonce hned v okamžiku, kdy je nový kus kódu úspěšně integrován do kódové základny. Když tým integruje kód častěji, možnost selhání je menší.

Některé týmy integrují nový kus kódu do kódové základny co nejdříve. Nasazení však odkládají o několik dní nebo týdnů, aby měly jistotu, že se v něm nevyskytují chyby. Tento proces neprobíhá plynule. Existují chyby, které se bez reálných (produkčních) dat těžko hledají, některé z nich jsou skryté v okrajových případech a testeři je téměř vůbec nenajdou.

3.2.3.2.8 Příležitost

Problémy jsou součástí vývoje softwaru. I když se jim tým snaží vyhnout sebevíc, dříve či později se obvykle objeví. Vypořádat se s nimi znamená vnímat problémy jako příležitost ke zlepšení a učení. Tým si může myslet, že řešit problémy více, než je nutné, aby to zvládl a mohl pokračovat ve vývoji, zpomalí tým. Avšak učení se z těchto příležitostí tým dlouhodobě zlepší.

3.2.3.2.9 Redundance

Redundance je klasickou vlastností vývoje softwaru. Je také spojená s refactoringem. Každý programátor se snaží odstranit nadbytečný kód. Redundance žije v kódu a také v procesech. Dva různé kódy mohou vypadat stejně, lze je považovat za redundantní, ale některé nuance je odlišují a tyto dva nesouvisející kusy kódu by měly zůstat nedotčeny.

Stejná pravidla by se dala aplikovat i na procesy. Testování softwaru po programátorech lze považovat za nadbytečnost. My však tento typ redundance chceme, jelikož slouží jako další síť pro zachycení chyb.

3.2.3.2.10 Dětské krůčky

To, že je vždy lepší provádět změny spíše po menších krocích než po velkých, je ve vývoji softwaru známé od samého počátku. Pokud by se prováděly zbytečně velké kroky, mohlo by se zvýšit riziko defektu, tím i doba produkce. Při malých krocích je i rychlejší zpětná vazba. Je to v podstatě práce v cyklech

3.2.3.3 Výhody extrémního programování

Snížení času a nákladů

Největší výhodou extrémního programování je, že tato metodika umožňuje společnostem vyvíjejícím software ušetřit náklady a čas potřebný k realizaci projektu. XP eliminuje neproduktivní činnosti, čímž snižuje náklady i frustraci všech zúčastněných a umožňuje vývojářům soustředit se na kódování.

Snížení rizika

Další z výhod extrémního programování je, že snižuje rizika spojená s programováním nebo s neúspěchem projektu. Na konci XP zajišťuje, že klient dostane přesně to, co chce.

Jednoduchost

Vývojáři dávající přednost této metodice vytvářejí extrémně jednoduchý kód, který lze kdykoli vylepšit.

Transparentnost

Základní výhodou XP je, že celý proces je transparentní a zodpovědný. Vývojáři se konkrétně zavazují k tomu, čeho dosáhnou, a vykazují konkrétní pokrok.

Důležitá je neustálá zpětná vazba, je dobré software předvádět včas a často, pozorně naslouchat a provádět potřebné změny. Sprinty pomáhají týmu postupovat správným směrem.

Tento přístup rychleji vytváří funkční software. Pravidelné testování ve fázi vývoje zajišťuje odhalení všech chyb a používání zákazníkem schválených validačních testů pro určení úspěšného dokončení kódovacího bloku zajišťuje implementaci pouze toho, co zákazník chce, ničeho navíc.[9][10]

Spokojenost

Extrémní programování pomáhá zvyšovat spokojenost a udržení zaměstnanců. Je to přístup zaměřený na hodnoty, stanovuje pevnou pracovní dobu s malým prostorem pro přesčasy. Rozdělení rozsahu projektu na dílčí komponenty a neustálá zpětná vazba od zákazníka zabraňují hromadění velkého množství práce, kterou je třeba dokončit před napjatým termínem.

Týmová práce

Každý je součástí týmu. Členové týmu spolupracují na všem – od požadavků až po kód. Vývojáři pracují ve dvojicích, programují ve dvojicích a nikdy se necítí osamoceni nebo zapomenuti. Také zde existuje párové programování, kde kód pro produkční prostředí vytvářejí vždy dva programátoři společně.

3.2.4 Metodika Crystal

Oproti ostatním agilním metodikám, které se nezaměřují na velikost skupiny, se metodika Crystal právě na toto zaměřuje. Metodika Crystal je ve skutečnosti seskupením různých metod, jež vždy zohledňuje velikost týmu. Z tohoto důvodu lze metodiku Crystal dále rozdělit do barevných skupin, které se odvíjejí od velikosti týmu a složitosti nebo velikosti projektu.

Malé týmy se mohou rozvíjet rychle a spolupracovat přímočařeji. S rostoucí velikostí týmu však často roste i složitost projektu. To představují různé metody Crystal.

Metodika Crystal má také tři různé priority:

- Bezpečnost projektu

Prioritou bezpečnosti je získat z projektu přiměřený obchodní výsledek, a to vzhledem k tomu, že je třeba priority projektu a omezení zdrojů.

- Efektivita vývoje

Efektivita vývoje je zásadní prioritou, protože mnoho projektů je ekonomicky nadměrně omezeno.

- Obyvatelnost výsledných konvencí

Obyvatelnost se stala nejvyšší prioritou, když si Alistair Cockburn při rozhovorech o projektech začal všímat, že většina programátorů má moc ignorovat jakoukoli metodiku, která je předepsána jejich organizací.[5][17]

3.2.4.1 Sedm vlastností

Existuje sedm vlastností, podle kterých se metodika Crystal řídí. Jsou to: časté doručování, reflexivní zlepšování, osmotická komunikace, osobní bezpečnost, soustředěnost, snadný přístup k odborným uživatelům a technické prostředí s automatizovanými testy, konfigurace, správa konfigurace a častá integrace.

3.2.4.1.1 Časté doručování

Umožňuje často doručovat funkční a otestovaný kód skutečným uživatelům. Nemusí se tak potýkat s tím, že se energie a čas investují do produktu, který nikdo nechce koupit.

3.2.4.1.2 Reflexivní zlepšování

Bez ohledu na to, jak špatný nebo dobrý produkt je, vždy existují oblasti, kde lze produkt vylepšit. Také vždy existují nové techniky a metody, jež může tým zavést, aby zlepšil své budoucí postupy.

3.2.4.1.3 Osmotická komunikace

S týmem, který pracuje společně, proudí informace kolem týmu. To jim umožňuje získávat cenné informace, aniž by se přímo účastnili diskuse o určité záležitosti. Toto postupné vstřebávání myšlenek se nazývá osmotická komunikace. Alistair Cockburn se domnívá, že tento druh pracovní atmosféry může fungovat i s velmi malou strukturou.

3.2.4.1.4 Osobní bezpečnost

Jediným způsobem, jak vybudovat zdravou pracovní atmosféru a skutečnou týmovou kulturu, je praktikovat otevřenou a upřímnou komunikaci. Členové týmu by měli mít možnost mluvit beze strachu, ať už prezentují nový nápad nebo hovoří o potenciálním problému.

3.2.4.1.5 Soustředěnost

Každý člen týmu přesně ví, na čem má pracovat, což mu umožňuje soustředit pozornost a vyhnout se přepínání z jednoho úkolu na druhý. Také to posiluje týmovou komunikaci a pomáhá týmu stanovit priority a pracovat na stejných cílech.

3.2.4.1.6 Snadný přístup k odborným uživatelům

Metodika Crystal umožňuje týmu udržovat komunikaci a získávat pravidelnou zpětnou vazbu od skutečných uživatelů.

3.2.4.1.7 Technické prostředí s automatizovanými testy, konfigurace, správa konfigurace a častá integrace

Obsahuje velmi specifické technické nástroje, které má tým pro vývoj softwaru používat při testování, správě a konfiguraci. Tyto nástroje umožňují týmu identifikovat jakoukoli chybu v kratším čase.

3.2.4.2 Pět barevných skupin

Existuje pět barev představujících pět skupin metodiky Crystal, které je třeba přijmout v závislosti na velikosti projektu.

3.2.4.2.1 Clear Crystal

Bílá nebo průhledná barva je určená pro malé týmy o velikosti 1–6 lidí. Podporuje pevnou cenu a smlouvu bez vyjednávání. Orientuje se na lidi a příliš se nezaměřuje na procesy a produkty. Vyžaduje dokumentaci a je zaměřená na bezpečnost projektu.

3.2.4.2.2 Crystal Yellow

Žlutá barva značí tým o velikosti 7–20 osob. Doporučuje se, aby každý vývojář vlastnil svoji oblast kódu. Vlastnictví kódových oblastí je definováno tak, že v případě potřeby změn se o ně stará pouze osoba, která daný kód vlastní nebo vytváří. Zároveň je důležitá zpětná vazba, jež se získává od skutečných uživatelů produktu. Tím se eliminují další nejasnosti, které mohou nastat v důsledku nepřímé komunikace. Přednost má přímá komunikace. Ta snižuje potřebu příliš rozsáhlé dokumentace. Proto se pro vývojáře stává snadnějším porozumět své práci. Programové prohlášení jsou cíle, které jsou definovány a ověřeny u zákazníka. K rychlejšímu řešení chyb se používá automatizované testování. Také se stanovují měsíční plány na zlepšování, což zahrnuje sestavení seznamu úkolů a jejich dosažení v daném čase.

3.2.4.2.3 Crystal Orange

Pro oranžovou barvu je specifická velikost týmu 21 až 40 osob. Projekt trvá jeden až dva roky, a to s tím, že rozdělení týmů je podle jejich funkčních dovedností, to se používá v projektech, které se neustále kódově vyvíjejí a které používá veřejnost

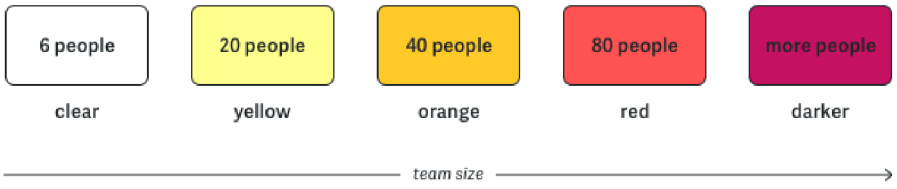
3.2.4.2.4 Crystal Red

Červená značí tradiční metodu vývoje softwaru a používá se při velikosti týmu 40–80 lidí. Kromě toho se týmy vytvářejí a rozdělují podle potřeby práce.

3.2.4.2.5 Crystal Maroon

Tmavě červená až kaštanová barva je pro tým o velikosti 80–200 členů. Je určena pro velké projekty. Kromě toho jsou metody definované různě podle potřeby softwaru.

Obrázek 6 – Crystal metodika[6]



4 Vlastní práce

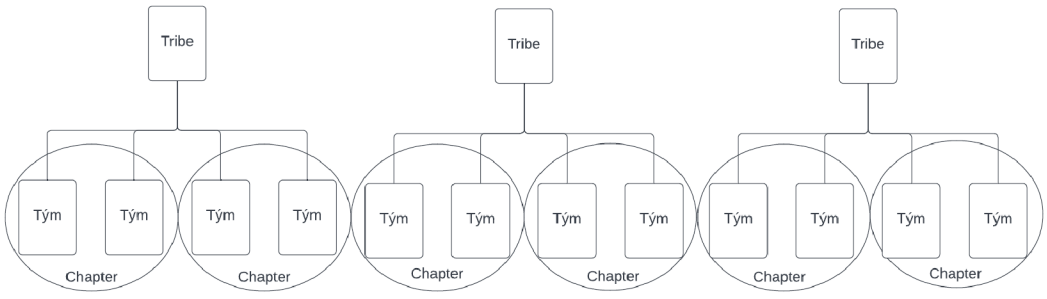
4.1 Popis podniku

Praktickou část práce autor vytvořil ve spolupráci s českou společností zabývající se bankovními službami, kde spolupracoval na vývoji podnikových informačních systémů pro různá průmyslová odvětví. Společnost patří k nadnárodním společnostem působícím především na českém a evropském trhu, ale je distribuována i do dalších zemí světa.

Po diskusi s Product Ownerem vývoje jednoho z týmů lze vymezit interní rozdělení jednotlivých vývojových týmů a jejich produktů podle velikosti cílových aplikací, na něž je software aplikován. Obecně je každý tým přiřazený do různých tribů, které jsou rozdělené podle koncového využití. Tribe je skupina oddílů pracujících v příbuzných oborech. Je navržen tak, aby neměl více než 100 členů. Některé jsou například zaměřené na koncového uživatele a jiné zase na vnitřní aplikace ve společnosti. Díky tomu je organizační struktura rozdělená stromově.

Celá informační struktura funguje na základě metodiky Scrum. Pro Scrum tabuli se využívá upravený software Jira.

Obrázek 7- Rozložení agilní části firmy[7]

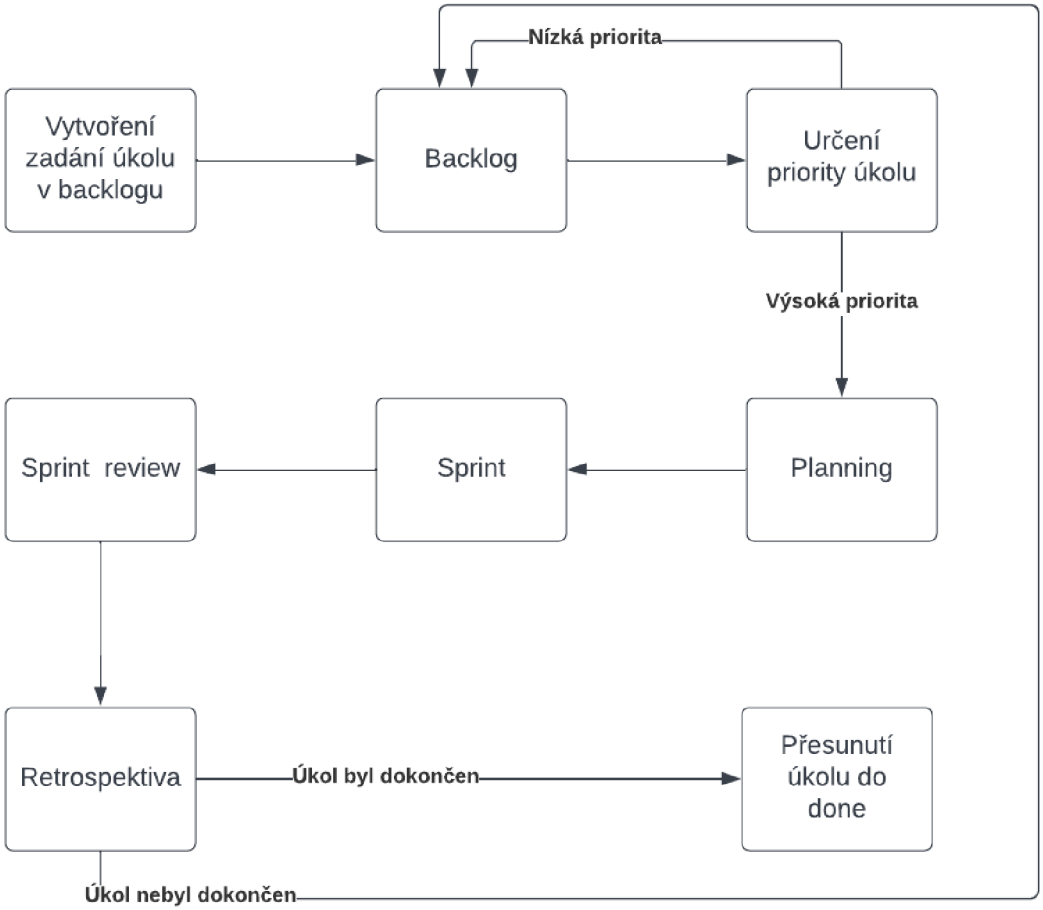


4.2 Součinnost s podnikem

Autor práce pravidelně osobně docházel na místo působení firmy a pozoroval, jakou agilní metodiku vývoje tým využívá. Nejprve je nastíněn obecný popis týmu a praktik, které realizuje. Na základě tohoto pozorování je provedena analýza využívané metodiky a je podán návrh řešení na zlepšení efektivity řízení vývoje. Sběr informací probíhal skrze práci ve firmě a kladení dotazů.

Vývojový tým společnosti vytváří podnikové informační systémy. Tým se skládá z analytiků, frontend a backend vývojáře, testerů, agilního kouče a Product Ownera. Tým využívá praktiky stand-up meetingů, groomingů, sprint planningů a retrospektiv. Grooming je proces zpracování úkolů v backlogu s cílem zajistit, aby byly prioritní. Autor bakalářské práce se zúčastnil celých sprintů u tří rozdílných projektů, a ačkoliv sprinty byly podobné, v mnoha detailech se přesto lišily.

Obrázek 8 - Průběh vytvoření úkolu ve sprintu[8]



4.3 Tým 1

Tento tým je zaměřený na vývoj aplikace, která úzce spolupracuje s databázemi a funguje s dalšími aplikacemi, kvůli tomu ji běžný pracovník ani uživatel nepoužívají. Jedna část vývojářů pracuje s SQL a druhá s Pythonem. Tým je rozdělen do více částí podle oboru.

Tento tým má flexibilní docházku, člověk by ale měl dorazit nejpozději do 9:00 a musí chodit do práce minimálně 3 dny v týdnu (z toho je povinná středa), zbylé dva dny může zaměstnanec pracovat distančně (home office).

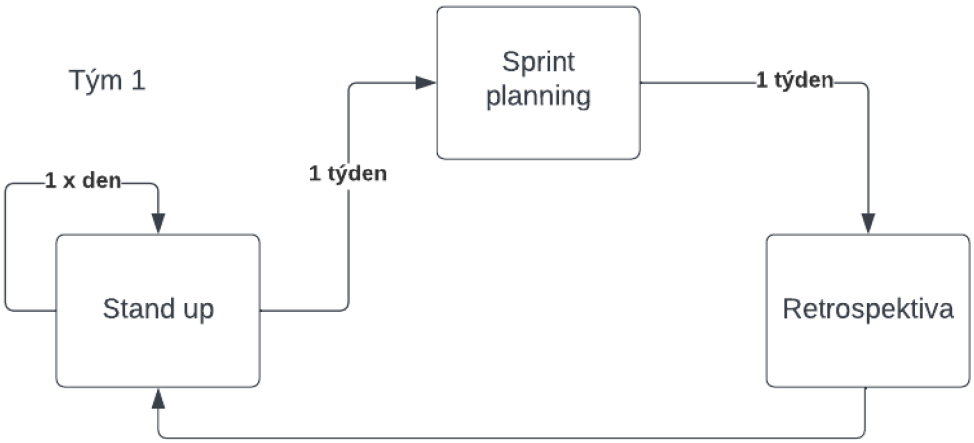
Tento tým má 14 členů, ve skupinách jsou podle oboru. To znamená Product Owner, databáze, vývojáři, testeři, analytici a systémoví administrátoři.

Každá z těchto skupin má dost odlišnou práci od ostatních, ale zároveň je potřebná komunikace mezi nimi, aby celý tým fungoval bezproblémově.

Díky tomu, že tento tým existuje už víc než dva roky, tak spolupráce a zvyky fungují.

Jako velkou chybu jsem našel, že při Retrospektivě si říkají pouze to špatné. Nevymýšlí k tomu ani řešení a jen si stěžují. Tím chybí Definition of Done.

Obrázek 9 - Tým1[9]



4.3.1 Sprinty

Každý sprint je opakující se časový úsek, který je v týmu působení autora plánovaný na čtrnáct dní. V tomto úseku tým dodá předem určené vylepšení nebo úpravu vyvíjené aplikace. Stand-up týmu probíhá denně, sprint planning a retrospektiva pak každých čtrnáct dní. Z důvodu, že sprinty jsou nastavené tak, že začínají a končí v půlce jednoho dne, tak téměř celý poslední/první den sprintu se zaplní schůzkami.

4.3.2 Stand-upy

Stand-up je pravidelný časový úsek, který je v týmu vždy plánovaný na každý den. Stand-up týmu probíhá denně, sprint planning a grooming pak probíhají každých čtrnáct dní. Tyto schůzky jsou povinné a vždy se do toho musí zapojit celá skupina. Stand-up vždy začíná v 9:00 a je na něj vyhrazených 30 minut. Každý člen musí říct to, co dokončil a co dokončí

dnes. Zároveň si ve svém časovém úseku může říct o pomoc, když ji potřebuje. Každý člověk má své místo a čas vyhrazený pro sebe, jsou to dvě minuty. Když zaměstnanec potřebuje více času, než je vymezeno, Product Ownerovi to nevadí, stand-up se pouze protáhne. Zároveň se promítá Scrum board, kde jsou vidět úkoly přiřazené každému členovi a lze sledovat, jak s nimi pokročil.

Zaměstnanci jsou seřazení podle pozic, díky tomu na sebe dobře navazují.

4.3.3 Sprint planning

Sprint planning probíhá v první den ve sprintu. Bývá ve 14:00 a je časově vymezený na 45 minut. Ve sprint planningu se plánuje, jaké úkoly tým udělá a jakou mají náročnost. Story point popisuje náročnost. Zde se používá škála od 0,5 do 10, a to s tím, že nejběžnější jsou v týmu hodnoty 3 a 4. Díky tomu mají členové možnost si vypočítat, kolik toho stihnou za čtrnáctidenní období sprintu. Na to, aby si členové týmu zadávali velikostně správné hodnoty, dohlíží Product Owner, který také obstarává všechny schůzky.

Během sprint planningu se také vybírají úkoly z backlogu a dávají se do Sprint backlogu ve Scrum boardu.

4.3.4 Retrospektiva

Tato schůzka je vždy poslední schůzkou sprintu a používá se jako konzultace o tom, co by mohlo jít lépe, ale i toho, co naopak šlo tak dobře, že to vyžaduje pozornost. Zároveň se zde podávají informace o novinkách anebo o věcech, co se začali dělat a osvědčily se

Retrospektiva je časově vymezená na 60 minut.

4.3.5 Review

Tento tým bohužel review nedělá, i když by měl.

4.4 Tým 2

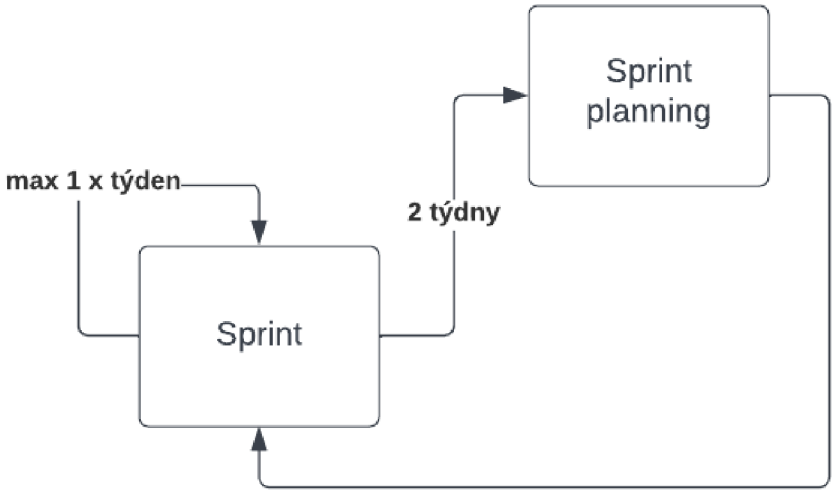
Tento tým je zaměřený na vývoj aplikací na webové stránky, které používají zákazníci každý den. Zároveň udržuje celé webové stránky společnosti. Vývojáři z tohoto týmu pracují s JavaScriptem, HTML a CSS. Zde je tým také rozdělen do více částí podle druhu práce. Tým má flexibilní docházku, většina členů však chodí jenom do kanceláře

Tento tým má pouze 8 členů, ti jsou ve skupinách podle oboru. To znamená Product Owner, vývojáři, testeři, analytici a systémoví administrátoři.

I když tento tým také existuje více než dva roky, tak jeho spolupráce funguje, ale zvyky ne. Další chyba, kterou jsem zjistil při spolupráci je ta, že se v Kanban boardu vůbec nepohybují úkoly. Díky tomu, že nemají stand-upy nikdo neví v jakém stavu je sprint a díky tomu chybí transparentnost. Také díky tomu, že nemají retrospektivu, tak porušují adaptaci.

Také nemají Grooming, u kterého by si tým měl říkat o co jde v úkolu a jak se daný úkol musí vyřešit. Díky tomu to dělá tým až za pochodu a tím pádem se často projevují nečekané problémy, se kterými nepočítali. Proto přesouvají úkoly do dalších sprintů.

Obrázek 10 - Tým 2[10]



4.4.1 Sprinty

Sprinty jsou naplánovány stejně jako u Týmu 1. Každý sprint se opakuje, co čtrnáct dní. Stand-up týmu je naplánován denně a sprint planning a retrospektiva probíhají každých čtrnáct dní. Tyto schůzky nejsou povinné, kvůli tomu se ne vždy zúčastní všichni. Sprinty jsou nastavené tak, že začínají ve čtvrtek a končí ve středu.

4.4.2 Stand-upy

Ačkoliv stand-up týmu probíhá denně, jelikož jde o malý tým, tak se na něj pravidelně nescházejí. Ne vždy tak proběhne důležitá výměna informací týkajících se aktuálních projektů. Stand-upy jsou vždy naplánované na 8:30 a trvají 15 minut. Není pravidlem, že na nich všichni účastníci mluví. Mluví pouze ti, kdo mají důležité informace pro tým, což bývají většinou vývojáři. Při těchto schůzkách není Product Owner příliš aktivní, spíše poslouchá. Členové týmu nemají dané žádné pořadí a každý mluví, kdy chce.

4.4.3 Sprint planning

Sprint planning se koná vždy první den ve sprintu. Schůzka je vždy naplánována na 10:00 a je časově vymezená 60 minutami. Ve sprint planningu se probírá, jaké úkoly tým udělá a jakou mají hodnotu. Story pointy si každý vybírá sám.

Během sprint planningu se také vybírají úkoly z backlogu a dávají se do „To-do“ ve Scrum boardu.

4.4.4 Retrospektiva

Během kooperace autora s Týmem 2 autor retrospektivu nezažil, ale zjistil, že občas probíhá.

4.5 Tým 3

Tento tým vytváří interní aplikaci pro hodnocení zaměstnanců. Aplikace funguje tak, že si zaměstnanci mohou vybrat z předurčeného výběru dovedností a poté jim to musí nadřízený schválit. Díky tomu musí být pravidelně obnovována databáze zaměstnanců ve všech týmech. Aplikace je poměrně nová, protože byla do běžného provozu uvedena teprve na začátku února.

Tým má 10 členů a je také rozdělen do částí podle role zaměstnanců. Na rozdíl od ostatních týmů tento tým pracuje primárně dálkově. Pouze ve středy musí zaměstnanci chodit do kanceláře. Proto celá jejich komunikace je vedena přes MS Teams. V MS Teams má tento tým založených několik konverzačních kanálů a jiných kategorií, do kterých může člověk psát. Kvůli tomu není vždy pořádek v důležitých konverzacích.

Tým má členy na pozicích Product Owner, vývojáři, testeři, agilní kouč a analytici. Kvůli tomu, že je tento tým nový, jeho spolupráce není ještě doladěná, a proto to tým dohání přesčasy. Další chyba, kterou jsem v tomto týmu zaregistroval je absence automatizovaného testování. Díky tomu testeři mají při každém sprintu plno práce kontrolování dat a testování. Scrummaster byl z tohoto důvodu nucen začít testovat a díky tomu není jasné, když něco říká při schůzkách, jestli to říká z pozice scrummastera nebo testera.

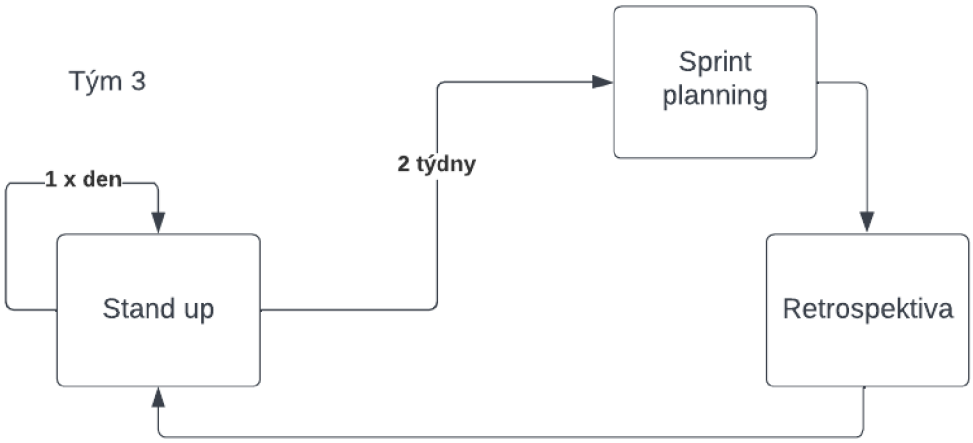
Také si dali měsíc na to, aby se tým stmelil bez tlaku, aby tým dodával ať už funkční nebo datové aktualizace.

V tomto týmu chybí odhodlání. Má sprintové cíle, ty cíle ne vždy naplní a spoustu věcí přesouvají do dalšího sprintu.

Je vidět, že tento tým má hodně problémů, ale jelikož je ještě mladý, tak se učí správně fungovat. Cíle se tomuto týmu daří, ale je jich hodně.

Další problém je ten, že spolupráce primárně probíhá ve skupinách podle oboru. Díky tomu chybí celková kooperace. Zároveň ne všichni členové jsou zvyklí na práci v agilním týmu, a proto nejsou úplně zvyklí na práci bez rozkazů. To je pozůstatek vodopádu.

Obrázek 11 - Tým 3[11]



4.5.1 Sprinty

Sprinty jsou naplánovány stejně jako u Týmu 1 a 2. Sprint trvá čtrnáct dní, stand-up probíhá denně a sprint planning a retrospektiva probíhají každých čtrnáct dní. Všechny schůzky jsou povinné. Sprinty jsou nastavené tak, že začínají ve čtvrtek a končí ve středu. V tomto týmu se snaží dbát na časové vymezení schůzek, a proto se nikdy nepřetahuje. Z toho důvodu se ne vždy stihne probrat vše, co je potřeba.

4.5.2 Stand-upy

Jako všechny schůzky jsou i stand-upy povinné. Proto se na nich sejde vždy celý tým, jsou naplánované na 20 minut, a to vždy v čase od 8:40. I když by schůzky měl vést Product Owner, vede je agilní kouč. Každý během stand-upu musí říct ve zkratce, co dělal a co bude dělat. Navíc zde řeší i problémy, které nastaly. Věci, jež se čistě netýkají stand-upu, se musí řešit na soukromé schůzce mezi lidmi, kterých se to týká.

4.5.3 Sprint planning

Sprint planning navazuje hned na první stand-up ve sprintu a je naplánovaný na 60 minut. V tomto týmu si lidé nerozebírají úkoly jako v předešlých týmech, ale agilní kouč je pouze přesouvá z backlogu do „To-do“ ve Scrum boardu.

4.5.4 Retrospektiva

Tento tým teprve nedávno začal mít retrospektivy během retrospektivy tým probírá (stejně jako Tým 1), co se povedlo a co ne. Na rozdíl od ostatních týmů k tomu však používá online nástroj, díky kterému můžou zaměstnanci napsat anonymně, co se jim líbilo, co ne apod. Zásluhou anonymity mají pocit bezpečí pro vyjádření svých pocitů.

4.5.5 Grooming

Pouze nedávno začali v tomto týmu probíhat groomingy. Díky tomu se mohou členové týmu dozvědět o úkolech na nadcházející sprint a podle toho se připravit.

4.6 Řešení pro jednotlivé týmy

V této části práce budou navrženy změny pro jednotlivé týmy při vývoji softwaru. Díky těmto změnám by mohly týmy docílit zvýšené produktivity a lepší komunikace v týmu.

Tyto týmy používají nástroje stavěné pro jejich potřeby, proto není moc věcí, které je potřeba změnit. Jediné, co by se mělo upravit, jsou jejich návyky. Tyto změny jsou doporučeny po nastudování teorie a zároveň po vlastním sledování zvyků v jednotlivých týmech.

4.6.1 Tým 1

Tým 1 je nejlepší ze zvolených tří týmů. Dělají všechny ceremonie a díky tomu zatím funguje dobře. Tento tým má zdravě nastavenou flexibilitu mezi prací z domova a v kanceláři. Zároveň má povinný den, který pomáhá při domluvě na schůzkách na konci/začátku sprintu.

Bohužel tyto schůzky nejsou naplánovány s ohledem na velikost týmu, takže se často časově přetahuje. První varianta je změna prodloužit schůzky o 10 minut, což zajistí, že se nebude přetahovat a každý zaměstnanec bude moci počítat s časovou náročností schůzek. Další varianta je, že tým by se mohl rozdělit do dvou menších týmu, a tak zmenšit časovou náročnost schůzek. Díky tomu, že ne vždy vše stihnou ze sprintu, díky nepřesným odhadům, tak by měli začít dělat plánování pokerů. Tím by snížili riziko, které sebou přináší nepřesné plánování. Zároveň by bylo lepší začít používat časový odhad u úkolů, a ne ukazatel jejich komplexity.

Jako další bod bych doporučil přidání agilního kouče do týmu – pro zlepšení komunikace týmu a průběhu schůzek. Z důvodu, že zde není agilní kouč, je zbytečně moc zodpovědnosti položeno na Product Ownera, který pak často bývá v časovém skluzu.

Díky tomu, že tým nedělá review a také protože chodí týmu až s velkým zpožděním požadavky, tak tým musí po pár sprintech se vrátit a změnit to, co měli špatně. Kdyby ho dělali tak to můžou hned po sprintu změnit.

Nakonec bych dodal, že nemají sprint cíle, díky kterým nemají pocit tlaku na dokončení backlogu sprintu. Proto nejsou moc motivovaní ke spolupráci.

4.6.2 Tým 2

Největším problémem v Týmu 2 je komunikace, proto jeho členové nejsou tak produktivní, jak by mohli být. Kdyby se zavedly povinné schůzky a každý člen by řešil důležité věci se všemi členy, tak by se neztrácel čas zbytečnými nedorozuměními. Týmu by pomohla i zvýšená produktivita Product Ownera, ten se týmu příliš nevěnuje. Změnou k lepšímu by bylo také přidání agilního kouče, který by dohlížel na to, aby tým chodil na všechny schůzky.

4.6.3 Tým 3

Tým 3 má drobné problémy s komunikací mezi jednotlivými částmi týmu. Díky tomu občas není koordinovaný, a tudíž se místy zadrhává. Může to být tím, že není ještě zvyklý na spolupráci mezi členy, ale také tím, že nechodí moc do kanceláře do kanceláře a veškerá komunikace probíhá online. Zároveň je nutné dodat, že většina členů pracuje i v jiných týmech, nebo pracují na částečný úvazek. Proto se většinou nedokáží sejít. A i když se sejdou ve středu, tak většinu dne stráví na schůzkách. Proto by pro ně byla vhodná varianta docházky, kterou má nastavenou Tým 1 kombinující práci v kanceláři a home office.

Jako další problém, který je důležitý adresovat je ten, že vzhledem k pevně stanoveným deadlinům a málo informacím k úkolům se často tvoří přesčasy. Jako řešení je jediná možnost, a to získávat více informací a posunutí deadlinů na časově výhodnou dobu.

Jako řešení absence automatizovaného testování je možnost toho, kterou aktuálně dělají. Dají si měsíc na to, aby se tým stmelil a během této doby začít dělat na automatizovaném testování. Bohužel to nejde dělat během normálního sprintu, protože na to v té době není čas.

Díky tomu, že úkoly často přetékají do dalších sprintů je možnost si prodloužit sprinty ze 2 týdnů na 3.

5 Výsledky a diskuse

Nejlépe ze všech týmů funguje Tým 1, který má již zažité své návyky. V průzkumu byla schválně využita spolupráce se třemi odlišnými týmy, které jsou na sobě nezávislé, aby bylo možné porovnat jejich fungování. Díky tomu se dá pozorovat odlišnost, za níž může velikost a stáří týmu. Dle mého názoru Tým 3 má největší potenciál k tomu být časem nejlepší. I přesto, že zaměstnanci z Týmu 3 se znají pouze krátkou dobu, tak každým sprintem se jejich komunikace a celková koordinace výrazně zlepšuje.

Návrhy pro všechna řešení fungování týmů byly doporučeny na základě aktuální epidemické situace. Zároveň návrh řešení a praktická část byly ovlivněny koronavirem. Obecně platí, že navrhované řešení zahrnuje subjektivní připomínky autora práce a nemůže zaručit 100% úspěšnost. Téměř s každou navrženou změnou dochází ke zvýšení nákladů.

Zároveň je důležité zmínit, že veškerý dodaný software pro jednotlivé týmy byl velice dobře přizpůsoben a není na něm nic, co by potřebovalo měnit. Díky tomu mohou týmy vykonávat svoji práci bez zbytečných zádrhelů.

Product owneři vidí jako nejzásadnější problém komunikaci a koordinaci v týmu.

6 Závěr

Hlavním cílem bakalářské práce je analyzovat agilní vývojové metody používané ve specifických skupinách pro vývoj aplikací a navrhnout následná řešení pro zefektivnění práce. Analýza a návrh řešení jsou založeny na teorii agilního řízení.

V teoretické části práce byly přiblíženy různé způsoby vývoje softwaru a popsány nejznámější a nejvíce využívané agilní metodiky. U každého způsobu vývoje jsou vykreslena základní pravidla, ale také funkce a vhodnost využití v realitě. Zároveň zde byla popsána historie agilních metodik.

V praktické části jsou zkoumány tři rozdílné týmy a představeno jejich řízení pomocí agilní metodiky Scrum. Díky tomu, že tyto týmy používají stejnou metodiku, se dají pozorovat rozdíly mezi jednotlivými týmy a jejich přístupem k fungování. I když jsou si tyto týmy poměrně podobné, najdeme mezi nimi značné diference v přístupu. Dále je v praktické části vypracováno řešení pro jednotlivé týmy, kde je popsána optimalizace jejich fungování pro zlepšení spolupráce.

Výsledky se však lišily od očekávání, protože Tým 3 vyvíjí software poněkud atypickým způsobem jen z prostředí domácí kanceláře bez osobního kontaktu. Na základě této skutečnosti je jednou z navrhovaných změn posouzení, zda setkání s osobami nezvyšuje efektivitu práce. Komunikace s Týmem 3 výhradně online také dala neočekávanou formu shromažďování informací prostřednictvím aplikace MS Teams.

I přesto, že tato firma pracuje s agilní metodikou poměrně krátkou dobu, přinesla do firmy mnoho kladů a možností práce v týmu. Zároveň se dá v ní pozorovat, že je ještě zvyklá pracovat vodopádově (například tím, že úkoly mají předem daný deadline místo dodávky, kde se nehledí na čas). Bohužel se agilní metodika nedostala do všech částí firmy, a proto to má negativní vliv na kooperaci mezi týmy agilními a neagilními.

7 Seznam použitých zdrojů

[1] AGILE ALLIANCE. Extreme Programming (XP). *Agilealliance.org* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://www.agilealliance.org/glossary/xp/

[2] ALTEXSOFT. Extreme Programming: Values, Principles, and Practices. In: *Hackernoon.com* [online]. 6. 3. 2018 [cit. 2022-02-13]. Dostupné z: https://hackernoon.com/
extreme-programming-values-principles-and-practices-6d270957fef7

[3] *Asana* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://asana.com/

[4] *Atlassian* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://www.atlassian.com/

[5] COCKBURN, Alistair. *Crystal clear a human-powered methodology for small teams.* Ebook, 2004. ISBN 9780132702492.

[6] COURSERA. The 3 Scrum Roles and Responsibilities, Explained. In: *Coursera.org* [online]. 20. 12. 2021 [cit. 2022-02-13]. Dostupné z: https://www.coursera.org/articles/
scrum-roles-and-responsibilities

[7] DCM. The Key Differences Between a Scrum Master & Product Owner. *Dcmlearning.ie* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://dcmlearning.ie/
product-owner-resources/the-key-differences-between-a-scrum-master-and-product-owner
.html

[8] DIGITE. What is a Kanban Board? *Digite.com* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://www.digite.com/kanban/kanban-board/

[9] DIGITE. What Is Extreme Programming (XP)? & It's Values, Principles, And Practices. *Digite.com* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://www.digite.com/agile/extreme-programming-xp/

[10] HERON, Joy. Simplicity – Fighting Complexity At All Costs. In: *Innoq.com* [online]. 3. 1. 2017 [cit. 2022-02-13]. Dostupné z: https://www.innoq.com/en/blog/simplicity-fighting-complexity/

[11] HOANG, Dat. The Fail-Fast Principle in Software Development. In: *Dzone.com* [online]. 5. 1. 2018 [cit. 2022-02-13]. Dostupné z: https://dzone.com/articles/fail-fast-principle-in-software-development

[12] KANBAN ZONE. What is Kanban. In: *Kanbanzone.com* [online]. 17. 1. 2016 [cit. 2022-02-13]. Dostupné z: https://kanbanzone.com/resources/kanban/

[13] *Kanbanize* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://kanbanize.com

[14] KNEAFSEY, Simon. A Short History Of Scrum. In: *Thescrummaster.co.uk* [online]. 5. 1. 2015 [cit. 2022-02-13]. Dostupné z: https://www.thescrummaster.co.uk/scrum/short-history-scrum/

[15] LYNN, Rachaelle. The History of Agile. In: *Planview.com* [online]. 22. 8. 2018 [cit. 2022-02-13]. Dostupné z: https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/history-of-agile/

[16] MISTRY, Ankur. What Is Extreme Programming(XP) And Core Values In Extreme Programming. In: *C-sharpcorner.com* [online]. 18. 12. 2019 [cit. 2022-02-13]. Dostupné z: https://www.c-sharpcorner.com/article/what-is-extreme-programming-xp/

[17] MRSIC, Maja. Crystal Methods. In: *Activecollab.com* [online]. 9. 8. 2017 [cit. 2022-02-13]. Dostupné z: https://activecollab.com/blog/project-management/crystal-methods

[18] NOVOSELTSEVA, Ekaterina. Extreme Programming; Tips & Advantages. In: *Apiumhub.com* [online]. 3. 2. 2020 [cit. 2022-02-13]. Dostupné z: https://apiumhub.com/
tech-blog-barcelona/extreme-programming-tips-advantages/

[19] *Project Manager* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://www.projectmanager.com/

[20] RYCHTECKÝ, Lukáš. Extreme programming: Values, Principles and Practices. In: *Flexiana.com* [online]. 20. 7. 2020 [cit. 2022-02-13]. Dostupné z: https://flexiana.com/2020/07/extreme-programming-values-principles-and-practices

[21] SARANGAM, Ajay. 8 Important Types Of Agile Methodology (2022). In: *Jigsawacademy.com* [online]. 23. 12. 2020 [cit. 2022-02-13]. Dostupné z:

https://www.jigsawacademy.com/blogs/product-management/types-of-agile-methodology/

[22] SHORE, Jim and Martin FOWLER, eds. Fail Fast. In: *Martinfowler.com* [online]. © 2004 [cit. 2022-02-13]. Dostupné z: https://www.martinfowler.com/ieeeSoftware/failFast.pdf

[23] SCHWABER, Ken a Jeff SUTHERLAND. *The Scrum Guide* [online]. PDF. 2020 [cit. 2022-02-13]. Dostupné z: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[24] SIMPLELEARN. What is Extreme Programming? XP Values, Principles and Practices. In: *Simplilearn.com* [online]. 18. 11. 2021 [cit. 2022-02-13]. Dostupné z: https://www.simplilearn.com/what-is-extreme-programming-article

[25] ŠOCHOVÁ, Zuzana. *Skvělý ScrumMaster*. Brno: Computer Press, 2018. ISBN 978-802-5149-270.

[26] WRIKE. Agile Guide. *Wrike.com* [online]. © 2022 [cit. 2022-02-13]. Dostupné z: https://www.wrike.com/agile-guide/

[27] ŠOCHOVÁ, Zuzana. Https://soch.cz/blog/management/agile/definition-of-done/. ZUZI´s blog [online]. 18.03.2018 [cit. 2022-03-12]. Dostupné z: https://soch.cz/blog/management/agile/definition-of-done/