Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Systémové inženýrství a informatika

Diplomová práce

Metody vývoje software

Jakub Dudáš

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jakub Dudáš

Informatika

Název práce

Metody vývoje software

Název anglicky

Methods of software development

Cíle práce

Cílem diplomové práce je efektivní zavedení agilních metodik ve firmě na konkrétním projektu.
Dílčími cíli práce bude popsání všech metodik vývoje software s hlavním zaměřením na agilní metodiky.
Dále pak výběr vhodné agilní metodiky a vhodného nástroje pro podporu vývoje software.

Metodika

Teoretická část diplomové práce bude založena na studiu odborných informačních zdrojů.

Budou popsány jednotlivé metodiky vývoje software. Dále pak budou popsány nástroje pro podporu agilního vývoje software.

V praktické části bude vybrána vhodná metodika pro danou firmu a daný projekt.

Budou také vybrány vhodné nástroje pro podporu agilního vývoje software.

Dále bude popsáno, jak probíhá zavedení konkrétní agilní metodiky pro vývoj software ve firmě.

Následně bude konkrétním projektu zavedena a bude sledován průběh několika plánovacích a vývojových iterací s případným navrhnutím změn.

Doporučený rozsah práce

50-60 stran

Klíčová slova

agilní metodiky, tradiční metodiky, vývoj software, řízení projektů IT, agilní nástroje

---

Doporučené zdroje informací

AXELOS, PRINCE2 Agile® Guidance, 12 Jun 2015, ISBN 9780113314676

Eduard Kunce, Zuzana Šochová, Agilní metody řízení projektů, Computer Press, a.s. 19.05.2014, ISBN: 978-80-251-4194-6

Ian Sommerville, Softwarové inženýrství, Computer Press, a.s. 2013, ISBN: 978-80-251-3826-7

Lyssa Adkins, Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition, earson Addison Wesley Prof 2010, ISBN: 9780321637703

Zuzana Šochová, The Great ScrumMaster: #ScrumMasterWay, ADDISON WESLEY PUB CO INC, 2017, ISBN: 013465711X

Předběžný termín obhajoby

2018/19 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 24. 1. 2019

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 1. 2019

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 31. 03. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Metody vývoje software" jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2019 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Markovi Píckovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích v průběhu zpracování celé diplomové práce.

Metody vývoje software

Abstrakt

V současnosti jsou velmi populárním a vyhledávaným řešením pro vývoj softwaru, právě agilní metodiky. Agilních metodik je ale poměrně velké množství, a proto je nutné si mezi nimi vybrat tu nejvhodnější. Tato práce popisuje vybrané agilní a rigorózní metodiky, aby byl vidět vývoj a bylo možné si udělat představu, co dané metodiky doporučují. Dále je v práci vybrána ta nejvhodnější metodika, aby přesně uspokojovala potřeby daného podniku a týmu. Poté je vybraná metodika implementována a dále vylepšována podle potřeb týmu. Pro podporu agilních metodik existuje také velká spousta softwarových nástrojů. V práci je vybraný konkrétní nástroj, který bude nejlépe vystihovat potřeby daného podniku a týmu.

Klíčová slova: metodiky vývoje software, agilní metodiky, rigorózní metodiky, agilní nástroje, METES, Scrum, Kanban, Extrémní programování, Feature Driven Development, řízení projektů IT

Methods of software development

Abstract

Currently, agile methodologies are very popular and searched software development solution. However, agile methodologies are quite large group of methodologies, so it is necessary to choose the most appropriate one among them. This thesis describes selected agile and rigorous methodologies in order to see the development and to get an idea of what they recommend. Furthermore, the most suitable methodology is chosen in order to satisfy the needs of the particular company and the team. Then the chosen methodology is implemented and further improved according to the needs of the team. There are also a lot of software tools to support agile methodologies. In the thesis is chosen a specific tool that will best describe the needs of the team.

Keywords: Agile methodologies, Methodologies of software development, Rigorous methodologies, Agile tools, METES, Scrum, Kanban, Extreme programming, Feature Driven Development, Management of IT projects

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Agilní metodiky se v posledních letech stávají fenoménem v oblasti vývoje softwaru a stále více firem se snaží přizpůsobit tomuto trendu. Přesto, podle průzkumu je poměr mezi používanými rigorózními a agilními metodikami vyrovnaný. Není totiž možné tvrdit, že agilní metodiky se dají použít ve všech typech projektů a ve všech firmách. Rigorózní metodiky poskytují navzdory své velikosti a složitosti stále velké množství výhod.

Diplomová práce popisuje vybrané metodiky vývoje softwaru. I když se v práci nevybírá mezi rigorózními metodikami, tak jsou i přesto okrajově popsány, aby bylo možné vidět vývoj mezi nimi a agilními přístupy. Agilní metodiky jsou v teoretické části práce popsané více podrobně, protože je cílem mezi nimi zvolit tu nejvhodnější pro daný podnik a tým. Je zde také popsáno, jakým způsobem bude daná metodika vybrána a jakým způsobem bude vybraný konkrétní nástroj pro podporu vybrané metodiky.

Ve vlastní práci je jako první popsaný projekt a tým, v rámci kterého, byla daná metodika zaváděna. Popis je pouze obecný, aby bylo zabráněno vynášení citlivých dat z dané firmy. Dále jsou zhodnoceny jednotlivé metodiky podle systému hodnocení a výběru metodiky METES a je vybrána ta nejvhodnější pro daný tým.

Před samotným zavedením metodiky, je třeba definovat podpůrné nástroje pro snažší dodržení všech předepsaných postupů. Toho bylo ve vlastní práci docíleno pomocí definice kritérií příslušnými členy týmu a následně byl pomocí metody AHP vybrán ten nejvhodnější z nástrojů.

V poslední řadě je popsána samotná implementace metodiky na projektu a je postupně vylepšována na základě zpětných vazeb jak od týmu, tak z výstupů sledujících samotný průběh práce.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem diplomové práce je výběr vhodné agilní metodiky a její následné zavedení na konkrétním projektu.

Jedním z dílčích cílů práce je popsání vybraných metodik vývoje softwaru s hlavním zaměřením na agilní metodiky. Spolu s výběrem samotné metodiky, bude vybrán i nástroj na podporu vybraných agilních procesů, který bude nejvíce vyhovovat stanoveným kritériím. Nástroje jsou v krátkosti popsané na základě definovaných kritérií. Dále jsou popsány jednotlivé metody výběru jak samotné agilní metodiky, tak i metody výběru vhodného nástroje.

2.2 Metodika

V teoretické části práce jsou popsané jednotlivé metodiky vývoje softwaru. Kromě agilních metodik, jsou popsané i rigorózní metodiky, aby byl vidět rozdíl a vývoj mezi nimi. Dále je popsán systém hodnocení a výběru metodik METES a metoda AHP. Obě metody jsou použité k výběru mezi jednotlivými alternativami.

V praktické části jsou analyzovány současné problémy a potřeby projektu. Následně je vybrána vhodná agilní metodika pro danou firmu a daný projekt za pomocí systému hodnocení a výběru metodik METES, který bude upravený podle potřeb projektu a týmu. Dále jsou definována jednotlivá kritéria pro výběr vhodného podpůrného nástroje pro vybranou metodiku. Mezi jednotlivými nástroji bude vybrána kompromisní varianta za použití metody AHP.

Dále je popsáno, jak probíhá zavedení konkrétní agilní metodiky pro vývoj software ve firmě. Následně je na projektu sledován průběh několika iterací a budou zavedená jednotlivá vylepšení, která vzniknou ze zpětných vazeb.

3 Teoretická východiska

3.1 Proces vývoje softwaru

V softwarovém inženýrství, je proces vývoje softwaru proces, který dělí vývoje softwaru do několika odlišných fází, pro zlepšení designu, produktového řízení a projektové řízení, což je také známé jako životní cyklus vývoje softwaru. Metodika může zahrnovat předdefinování konkrétních výstupů a artefaktů, které jsou vytvořeny a dokončeny projektovým týmem pro vývoj nebo údržbu aplikace. [15]

3.1.1 Historie

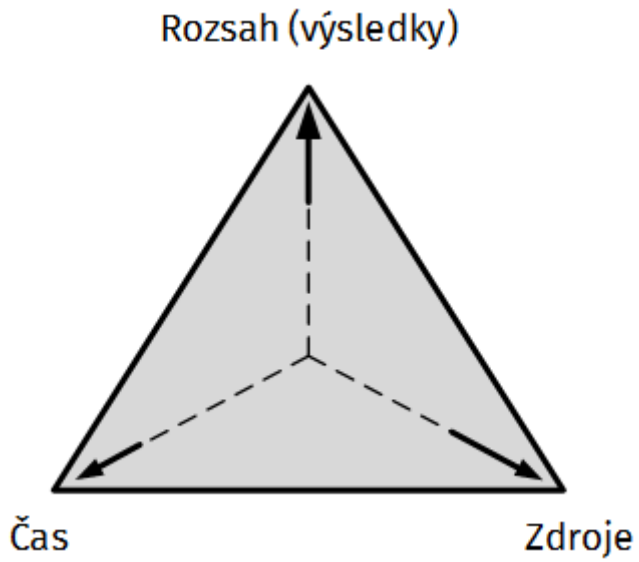
Metodologie vývoje softwaru (také známá jako SDM) se objevila v šedesátých letech 20. století. Podle Elliott (2004) lze životní cyklus vývoje systémů (SDLC) považovat za nejstarší formalizovaný metodický rámec, pro budování informačních systémů. Hlavní myšlenkou SDLC bylo „usilovat o rozvoj informačních systémů velmi úmyslným, strukturovaným a metodickým způsobem, který vyžaduje, aby každá etapa životního cyklu - od vzniku myšlenky až po realizaci konečného systému - byla prováděná přísně a postupně v kontextu uplatňovaného rámce.

Hlavním cílem tohoto metodického rámce v šedesátých letech bylo „vytvořit rozsáhlé funkční obchodní systémy ve věku velkých obchodních konglomerátů. Činnosti informačních systémů se pohybovaly kolem těžkého zpracování dat a superpočítačů“. Metodiky, procesy a rámce sahají od specifických kroků, které může organizace použít přímo v každodenní práci, až po flexibilní rámce, které organizace používá k vytvoření vlastního souboru kroků, přizpůsobených potřebám konkrétního projektu nebo skupiny. [15]

3.1.2 Projektový trojimperativ

Projektový trojimperativ definuje tři roviny, ve kterých se při realizaci projektu pohybujeme:

1. Rozsah projektu (Výsledky)
2. Náklady (Zdroje)
3. Časový rámec projektu (harmonogram)



Obrázek 1 - Projektový trojúhelník (Zdroj: [16])

Dále platí následující pravidla:

1. Kvalita práce je omezená rozpočtem, termíny projektu a rozsahem (vlastností).
2. Projektový manažer může v rámci projektového trojimperativu měnit zdroje.
3. Změny v jednom omezení vyžadují změny i v ostatních, aby neutrpěla kvalita

Například projekt lze dokončit rychleji, zvýšením rozpočtu nebo snížením rozsahu. Stejně tak rostoucí rozsah může vyžadovat rovnocenné zvýšení rozpočtu a času. Snížení rozpočtu bez přizpůsobení plánu nebo rozsahu, povede k nižší kvalitě. V praxi však vyměňování zdrojů není vždy možné. Například přidání lidí (peněz) nemusí znamenat zrychlení projektu, vzhledem k tomu že je nutné nové lidi zaškolit. Tím projekt ztrácí kapacity již zaškolených lidí. Projektový trojimperativ je obecně používán na analýzu projektu. [17]

Čas

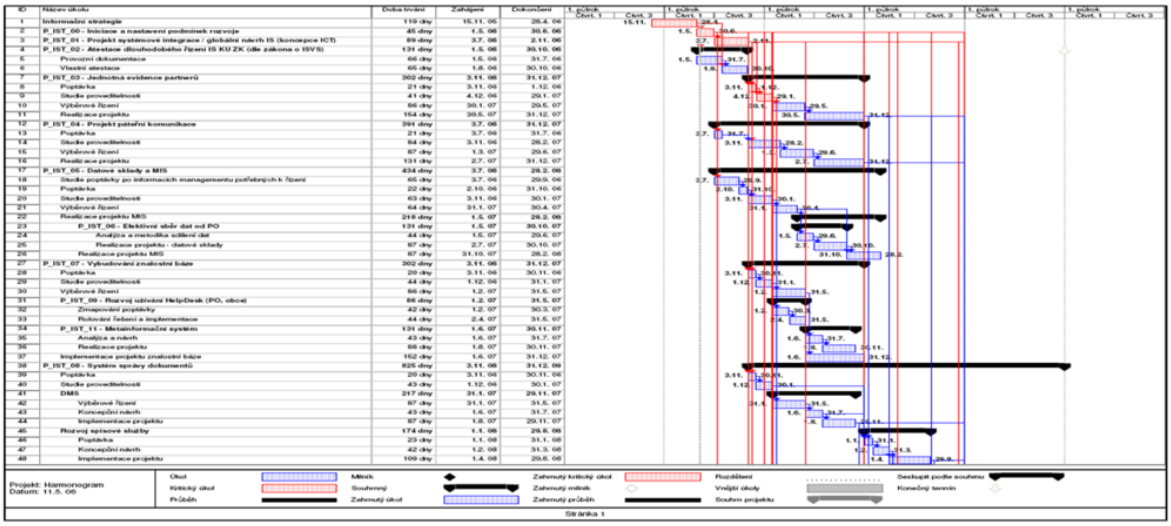
Pro analytické účely je doba potřebná k vytvoření produktu odhadnutá použitím několika metod. Jednou metodou je identifikace úkolů, potřebných k vytvoření výstupů dokumentovaných ve Work Breakdown Structure nebo WBS. Pracovní úsilí pro každý úkol je odhadnuto a tyto odhady jsou zařazeny do konečného odhadu pracnosti. Úkoly jsou také prioritizovány, jsou identifikovány závislosti mezi úkoly a tyto informace jsou dokumentovány v projektovém plánu. Závislost mezi úkoly, může ovlivnit délku celého projektu (omezení závislosti), stejně jako dostupnost zdrojů (omezení zdrojů). Čas závisí na všech ostatních kategoriích zdrojů a nákladů. Často se používají pro odhad zkušenosti z minulých podobných projektů [17]

Podle Project Management Body of Knowledge (PMBOK) procesy řízení projektu zahrnují:

1. Projektový plán
2. Definovaní aktivit
3. Sekvence jednotlivých aktivit
4. Odhad zdrojů pro aktivity
5. Odhad trvání aktivity
6. Vývojový plán
7. Řídicí plán

Ganttův diagram

Ganttův diagram je jedním z možných způsobů, jak vizualizovat plán projektuna časové ose, s vazbami závislostí úkolů a stanovenými trváními. Díky historickým údajům může pomoci pro přesnější odhady. Časové řízení je také důležité na úrovni člena týmu. Projektoví manažeři chtějí získat podporu od svého týmu, v této oblasti prostřednictvím nástrojů a procesů kolaborativního řízení času, aby byl projekt kolektivně schopen zůstat na správné cestě. [17] [36]



Obrázek 2 - Ganttův diagram (Zdroj: [36])

Náklady

Finanční závazek projektu závisí na několika proměnných. Existují zdroje, od materiálu k lidem, které zahrnují náklady na práci. Existují také fixní a variabilní náklady spojené s jakýmkoli projektem, jako jsou ekonomické náklady týmů s různými dovednostmi a produktivitou, které je třeba vypočítat.

Nákladové procesy zahrnují odhady nákladů, s cílem zjistit potřebný finanční závazek pro všechny zdroje potřebné pro dokončení projektu. Rozpočtování nákladů vytváří základní nákladovou základnu. Kontrola nákladů slouží k řízení změny nákladů v průběhu celého projektu.

Existuje několik metod pro odhad nákladů projektu:

1. Historické údaje: Použití nákladů podobných projektů pro srovnání.
2. Náklady na zdroje: Stanovení sazby nákladů na zboží a práci podle jednotlivých jednotek.
3. Bottom Up: Odhady od nejnižšího až po nejvyšší úroveň práce.
4. Parametrický: Měří statistický vztah mezi historickými daty a dalšími proměnnými.

Cena je jedním z komplikovanějších bodů trojúhelníku. [17]

Rozsah

Jak bylo uvedeno, rozsah projektu se zabývá konkrétními požadavky nebo úkoly nezbytnými pro dokončení projektu. Rozsah je důležitý pro řízení jakéhokoli projektu, ať už jde o agilní softwarové projekty nebo o dobře plánované projekty vodopádu, protože pokud nemůžete kontrolovat rozsah projektu, pravděpodobně nebude včas doručen nebo nebude doručen v požadovaném rozpočtu. Při spravování rozsahu je rozhodující, že jsou všechny úkoly prioritizovány, což umožňuje účinně naplánovat a přiřadit zdroje.

Dalším klíčovým faktorem je jasné definování dodávky a očekávání všech zúčastněných stran. Může se stát, že hlavně v dlouhodobých projektech se budou měnit požadavky, a proto je pak třeba ukázat všem, co bylo původně v plánu a nebo upravit všechny ostatní strany trojúhelníku

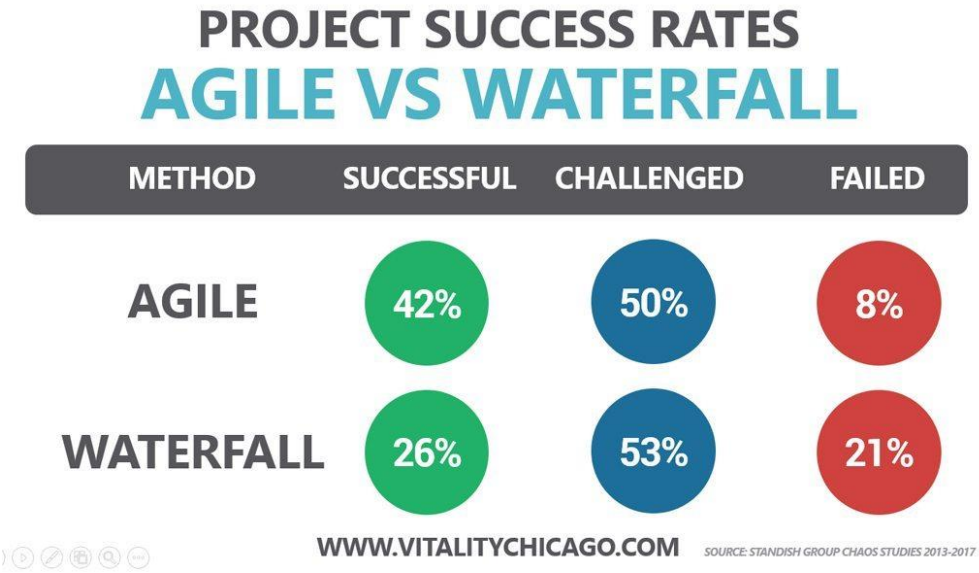
Aby se vyhovělo požadavkům zúčastněných stran a novým požadavkům, které přirozeně přicházejí při rozvíjení projektů, musí být projekt schopnen zvládat změny. [17]

3.1.3 Úspěšnost projektů

Nejnovější výsledky studie Standish Group Chaos ukazují úspěšnost a míru selhání projektu Waterfall a Agile. Nemělo by být překvapením, že Agilní projekty mají statisticky dvakrát větší pravděpodobnost úspěchu. Skupina Standish provádí průzkumy úspěšnosti a neúspěšnosti projektů IT každé 2 roky od roku 1994.

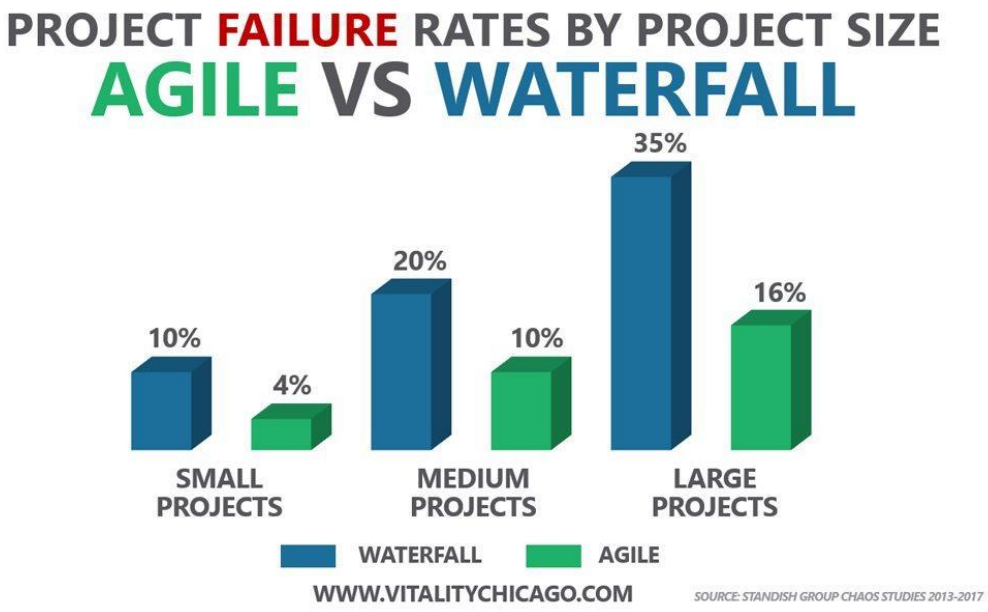
Standish Group Chaos hodnotila projekty jako úspěšné, problematické nebo neúspěšné.

1. Úspěšný projekt byl ten, který splňoval všechny tři hodnoty z projektového trojimperativu: plán, náklady a rozsah.
2. Problematický projekt by splňoval dvě ze tří omezení, například čas a rozpočet, avšak nikoli požadovaný rozsah.
3. Neúspěšný projekt je ten, který je zrušen před dokončením, nebo dokončen, ale nepoužit. [19]



Obrázek 3 - Srovnání úspěšnosti projektů (Zdroj: [19])

Velikost projektu také ovlivňuje úspěšnost projektu. Dalším hlavním zjištěním společnosti Standish Group je, že větší projekty mají vyšší míru selhání. [19]



Obrázek 4 - Neúspěšnost projektů (Zdroj: [19])

3.2 Porovnání agilních a rigorozních metodik

Rigorózní rozvojové přístupy existují již poměrně dlouhou dobu. Od svého zavedení byl model vodopádu (Royce 1970) široce využíván v rozsáhlých i malých softwarových projektech a byl prohlašován za úspěšný u mnoha projektů. Navzdory úspěchu má mnoho nevýhod, jako je linearita, nepružnost při měnících se požadavcích a vysoce formální procesy bez ohledu na velikost projektu. Kent Beck vzal tyto nedostatky v úvahu a představil Extreme Programming, první agilní metodiku.

Agilní metodiky se zabývají nestabilními a nepřesnými požadavky pomocí řady technik zaměřených na spolupráci mezi vývojáři a zákazníky a podporou včasného dodávání produktů. Shrnutí rozdílu mezi agilními a rigorózními metodikami je uvedeno v následující tabulce. [5] [7]

Tabulka 1 - Srovnání agilních a rigorózních metodik (Zdroj: Autor)

	Agilní metody	Rigorźní metody
Přístup	Adaptivní	Prediktivní
Měření úspěšnosti	Obchodní hodnota	Plnění plánu
Velikost projektu	Malý	Velký
Styl řízení	Decentralizováno	Autokratický
Perspektiva změny	Změna přizpůsobivosti	Změna udržitelnosti
Kultura	Vedení a spolupráce	Vést a kontrolovat
Dokumentace	Stručná	Podrobná
Důraz	Orientovaný na lidi	Procesně orientovaný
Cykly	Četné	Omezené
Doména	Nepředvídatelné / průzkumné	Předvídatelný
Počáteční plánování	Minimální	Obsáhlý
Návratnost investic	Brzy v projektu	Konec projektu
Velikost týmu	Malý / Kreativní	Velký

Agilní a rigorózní metodiky mají obě své silné a slabé stránky. Lidé obvykle používají některou z těchto metodik nebo se řídí svou vlastní metodikou. Existují hlavní faktory ovlivňující metodické rozhodnutí a výběr, která z nich je vhodná za daných podmínek. Tyto faktory lze rozdělit do velikosti projektu, lidí a rizika. [5] [7]

3.2.1 Velikost projektu

Jedním z omezení agilních metod je velikost projektu. Klíčovým prvkem velikosti projektu je rozpočet projektu, doba trvání a organizace projektového týmu. Čím větší tým nebo větší rozpočet potřebujete, tím větší je projekt. To znamená, že čím více požadavků, tím více je potřeba lidí a tím pádem je potřeba větší koordinace.

Rigorózní metodiky toto podporují poskytováním plánů, dokumentace a procesů pro lepší komunikaci a koordinaci mezi velkými skupinami. To ale neznamená, že se agilní metodiky nedají použít na velké projekty, je jen zapotřebí je trochu vice zformalizovat a projekt rozdělit do více týmů. [5] [7] [9]

3.2.2 Lidský faktor

Polovina hodnot agilního manifestu se zabývá lidskými faktory. Mít dovednosti a zkušené lidi v týmu je klíčový faktor agilních metodik. Povzbuzování odborníků z daného oboru k tomu, aby byli součástí týmu, dává vývojářům rychlou zpětnou vazbu od uživatele na jejich produkt. Přizpůsobivost zákazníků je dalším velkým faktorem, zákazník získá možnost kontrolovat pokrok a určovat směr vývoje softwaru během každé iterace. Získání této úrovně odhodlání od zákazníka činí agilní metodiku atraktivnějším procesem než rigorózní metodiku. [5] [7] [9]

3.2.3 Rizikové faktory

Nejdůležitějším rizikovým faktorem při vývoji softwarového procesu je kritičnost projektu a rychlost reakce na změnu. Agilní metodiky se používají v aplikacích, které lze rychle vyvinout a nevyžadují rozsáhlé zajištění kvality. Na kritické, spolehlivé a bezpečné systémy je vhodnější využít rigorózní metodiky. Pokud je projekt kritický, musejí být před vývojem softwaru dobře definovány všechny požadavky. [5] [7] [9]

Tabulka 2 - Charakteristika projektu podle typu metodiky (Zdroj: Autor)

Charakteristika projektu	Agilní metodiky	Rigorózní metodiky
Primární cíl	Rychlá hodnota	Vysoká jistota
Požadavky	Častý vznik urgentních požadavků, rychlá změna, neznámé	Znatelné brzy, převážně stabilní
Velikost	Menší týmy a projekty	Větší týmy a projekty
Architektura	Určeno pro současné požadavky	Určeno pro současné a očekávané požadavky
Plánování a řízení	Internalizované plány, kvalitativní řízení	Dokumentované plány, kvantitativní kontrola
Zákazníci	Aktivně spolupracují po celou dobu projektu	Definice požadavků na začátku projektu, kontrola na konci
Refactoring	Levný	Drahý
Rizika	Neznámá rizika, velký dopad	Dobře pochopená rizika, Menší dopad

Rozdíl mezi agilním a rigorózním pojetím vývoje softwaru můžeme také vidět na následujícím obrázku.

Obrázek 5 - Rigorózní a agilní trojúhelník (Zdroj: [20])

Rigorózní metodiky počítají s funkcionalitou jako s fixní veličinou. To znamená, že na počátku vývoje se upřesní přesné požadavky, které se musí dodržet, aby byl výsledný produkt akceptován. Čas a zdroje jsou proměnné, se kterými můžeme hýbat. Proto se u projektů vyvíjených pomocí rigorózní metodiky setkáváme často s překročením termínu dodání nebo s překročením nákladů, ať už u klienta nebo dodavatele.

Agilní metodiky mají opačný přístup. Jako fixní je na začátku projektu nastavený čas a zdroje a funkcionalita je proměnná. To znamená, že zákazník může dostat řešení, které zatím implementuje část funkcionality s nejvyšší prioritou, avšak díky agilnímu pojetí, je to v termínu, za dané peníze a podle skutečných požadavků zákazníka, které se v průběhu implementace mohly změnit. Důležité je, že nemusí být implementovány funkce, kterým zákazník spolu s dodavatelem definoval nejnižší prioritu. [20]

3.3 Rigorózní metodiky

Rigorózní metodiky vývoje kladou důraz na přesnou specifikaci všech požadavků na začátku projektu a na jejich podrobnou dokumentaci a smluvní odsouhlasení. Koncový zákazník není ve styku s vývojovým týmem, takže nemůže do procesu vývoje nijak zasahovat. V případě změn požadavků v průběhu vývoje začíná zpravidla celý proces od začátku. Vývojový tým se řídí pouze dokumentací. Zákazník dostává objednaný software po částech, nebo celý až na konci. [21]

Níže jsou popsané tyto vybrané rigorózní metodiky:

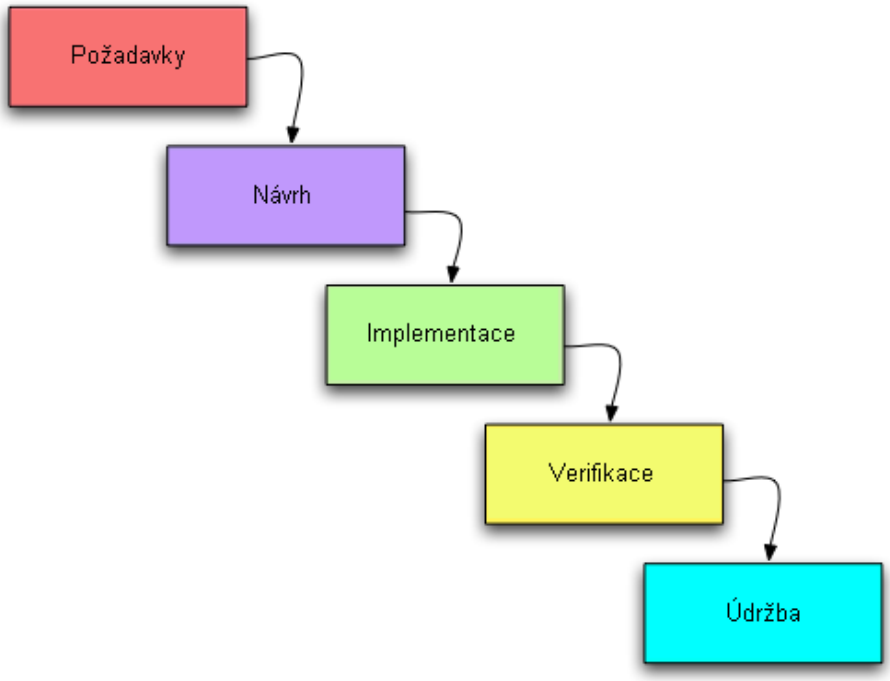
1. Vodopád
2. Racionální jednotný proces (RUP)
3. Spirálový model

3.3.1 Vodopád

Vodopád je rigorózní přístup, který odděluje proces vývoje produktu do skupin souvisejících úkolů, které musí být dokončeny před přesunem do další skupiny nebo fáze. Proto vyžaduje rozsáhlé plánování. Zavedení všech kroků před zahájením práce na vývoji produktu pomáhá minimalizovat nebezpečí a další chyby. Tím tým vždy ví, na čem by měli dále pracovat a co očekávat v budoucnu.

Základní fáze běžně používaných v metodice Vodopád:

1. Požadavky: zjistit a analyzovat, jaké jsou potřeby klienta a co by měl dělat konečný produkt.
2. Design: výběr správné technologie a vytvoření modelů produktu a detailní architektury.
3. Implementace: řešení problémů, implementace řešení
4. Ověření (testování): zjištění, zda produkt odpovídá stanoveným požadavkům na výkon a kvalitu.
5. Údržba: opravy chyb, aby se zajistilo, že produkt bude možné snadno používat [21]



Obrázek 6 – Vodopádový model (Zdroj: [22])

Vodopád poskytuje také jasné a podrobné časové lhůty a náklady. Ty pomohou tým vést k větší produktivitě. Nevýhodou vodopádu je, že je zastaralý pro požadavky moderního softwarového inženýrství. Psaní kódu a zároveň zajišťování kvality je poměrně obtížné, protože každá fáze této metodiky závisí na předchozí a žádné činnosti se nepřekrývají. S vodopádem musí týmy čekat na dokončení předchozích kroků. Pokud jsou opožděné, mohou být všechny ostatní úkoly odložené. [21]

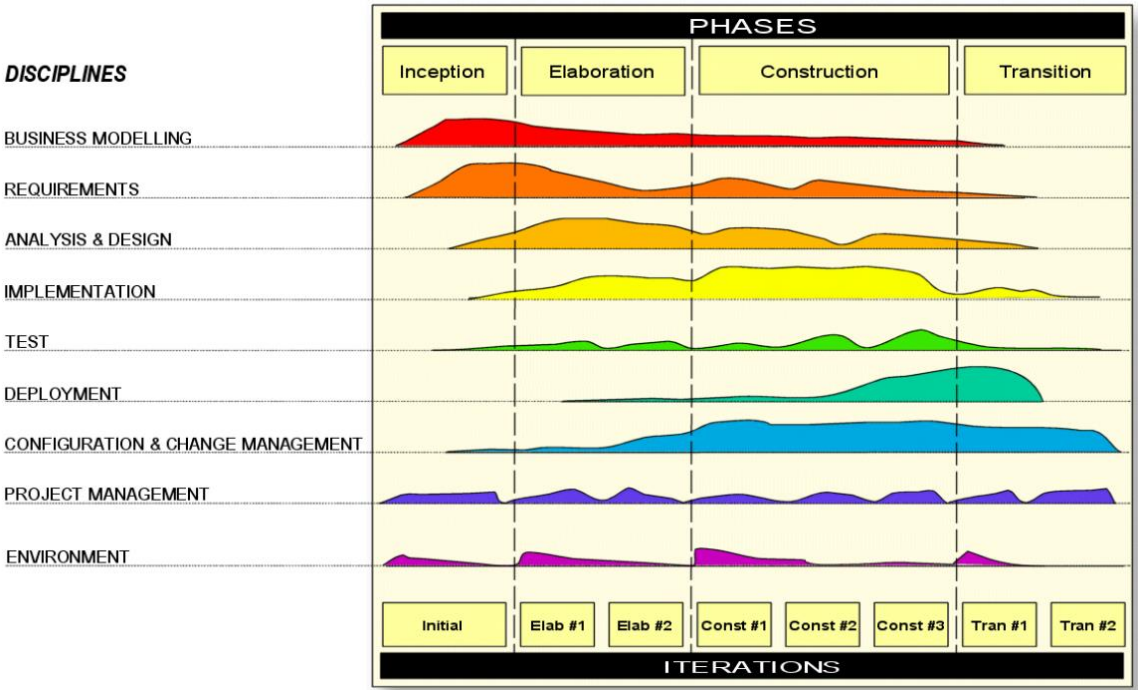
3.3.2 Rational Unified Process (RUP)

Rational Unified Process je procesní rámec, který podobně jako Extremní Programování poskytuje vhodnou osvědčenou praxi, standardy, šablony a vzory pro vývoj softwaru. Nabízí také organizovanější způsob přiřazování činností a rolí. Cílem tohoto procesu je vyvinout vysoce kvalitní software, který splňuje požadavky svých klientů a potřeby budoucích uživatelů v čase a v daném rozpočtu.

Rational Unified Process podporuje týmovou produktivitu tím, že nabízí všem členům skupiny přístup do znalostní oblasti, která obsahuje všechny informace a nástroje potřebné k tomu, aby jim pomohly provádět vývojové úkoly.
Rational Unified Process nemá pevný soubor procesů, které se musí za každou cenu dodržovat. Může být přizpůsoben tak, aby odpovídal požadavkům každého projektu. Každá fáze tohoto procesu je rozdělena do odlišných iterací, které je třeba dokončit předtím, než se přesunete do další fáze. [23]

Čtyři etapy, které projekt Rational Unified Process realizuje, jsou:

1. **Počátek:** vytvoření myšlenky za projektem a zjištění, zda máte správné zdroje pro jeho realizaci a zda odpovídá potřebám vaší organizace.
2. **Vypracování:** modelování architektury softwaru založené na dostupném rozpočtu a zdrojích, vyhodnocování rizik a příležitostí ke zjištění, jak mohou být změny nebo nové technologie přidány do projektu.
3. **Konstrukce:** provádí vývoj softwaru od jeho designu přes psaní kódu a testování.
4. **Přechod:** doručování konečného softwaru a provádění změn s cílem zlepšit výsledky nebo opravit jakékoliv problémy. [23]

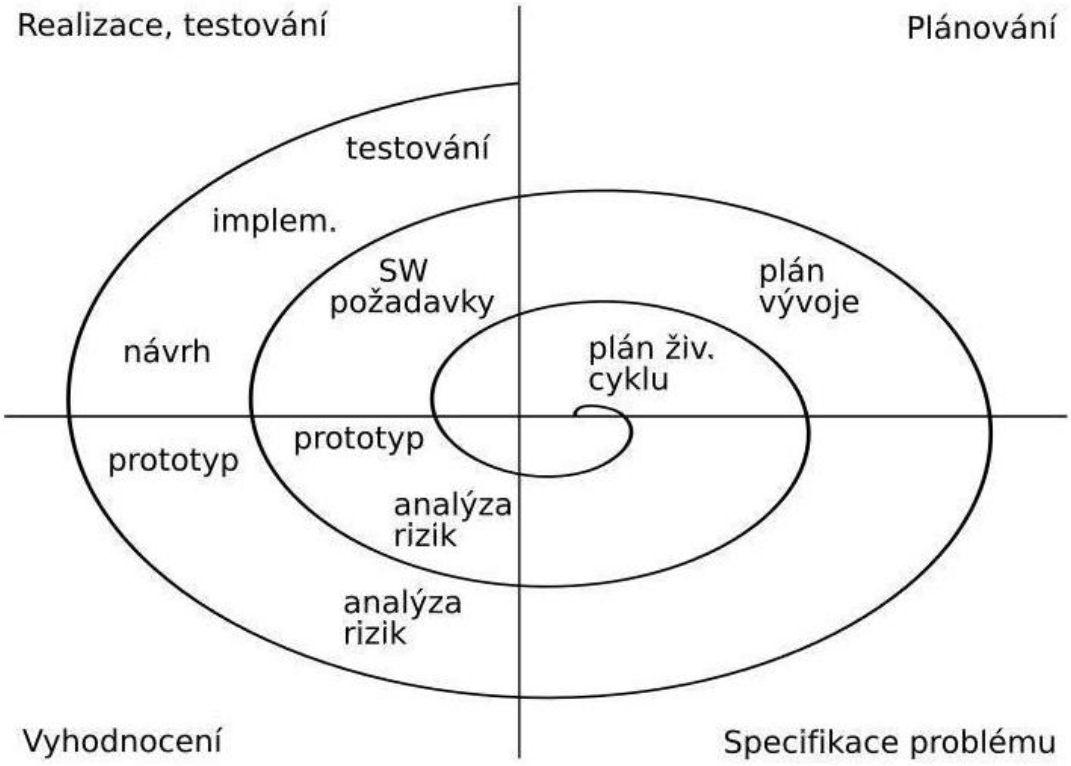


Obrázek 7 – Model RUP (Zdroj: [23])

3.3.3 Spirálový model

Spirálový model je řízený riziky, což znamená, že celkový úspěch projektu velmi závisí na fázi analýzy rizik. Analýza rizik vyžaduje specifickou odbornost při každé iteraci. Na první pohled se může zdát, že tento model je komplikovaný a neohrabaný a nejsou žádné důvody považovat tento přístup za jednu z možností, ale stejně jako všechny ostatní modely SDLC, má tento, kromě svých nevýhod, své jedinečné silné stránky.
Například je možné přidat některé další funkce v posledních fázích vývoje softwarových produktů.

Spirálový model může být charakterizován tím, že opakovaně v každé iteraci spouští sadu elementárních procesů vývoje a eliminuje riziko.
Spirálový model se skládá ze čtyř hlavních fází vývojového cyklu softwaru. Celý vývojový proces opakovaně prochází těmito fázemi. Každá iterace se nazývá Spirála. [4]



Obrázek 8 - Spirálový model (Zdroj: [24])

Čtyři hlavní fáze Spirálového modelu jsou:

Určení cílů

Zde všechno začíná. Členové týmu se snaží shromáždit cíle projektu, alternativy designu apod. V následujících spirálách jsou všechny požadavky generovány podle zpětné vazby zákazníka. Trvalá komunikace mezi zákazníkem a řízením projektu je proto zásadní.

Analýza rizik

Jedná se pravděpodobně o nejvýznamnější fázi. Rizika jsou možné podmínky a události, které brání rozvojovému týmu od splnění jeho cílů. Existuje široká škála rizik, od triviálních až po fatální rizika. Primárním úkolem vývojového týmu je eliminovat všechna možná rizika a prioritizovat je podle důležitosti. Dalším krokem je stanovení potenciálních strategií, které mohou pomoci rizika překonat. Vyhodnocení těchto parametrů může v následujících krocích způsobit změny. Na konci této fáze se vyrábí prototyp.

Vývojová fáze

V této fázi se plánovaný produkt vyvíjí a spolu s tím se spouští testy. Během první spirály, kdy celkové požadavky nejsou tak jasné, je vytvořen tzv. Proof of Concept (POF), který dostane zpětnou vazbu od zákazníka. Později v následujících spirálách, může být vyvíjena pracovní verze produktu s názvem „Build“, která je klientovi zaslána, aby dostala novou, podrobnější zpětnou vazbu. Tento přístup umožňuje dosáhnout vyššího detailu požadavků.

Hodnotící fáze

Tato fáze umožňuje vyhodnotit výstup projektu předtím, než projekt pokračuje do další spirály. Spirálový model se nazývá meta-model, protože využívá modely Vodopád i prototypování. Je však velmi důležité pochopit, že Spirálový model není jen řada přírůstků k vodopádu. Ve skutečnosti je tento model poměrně flexibilní. [4]

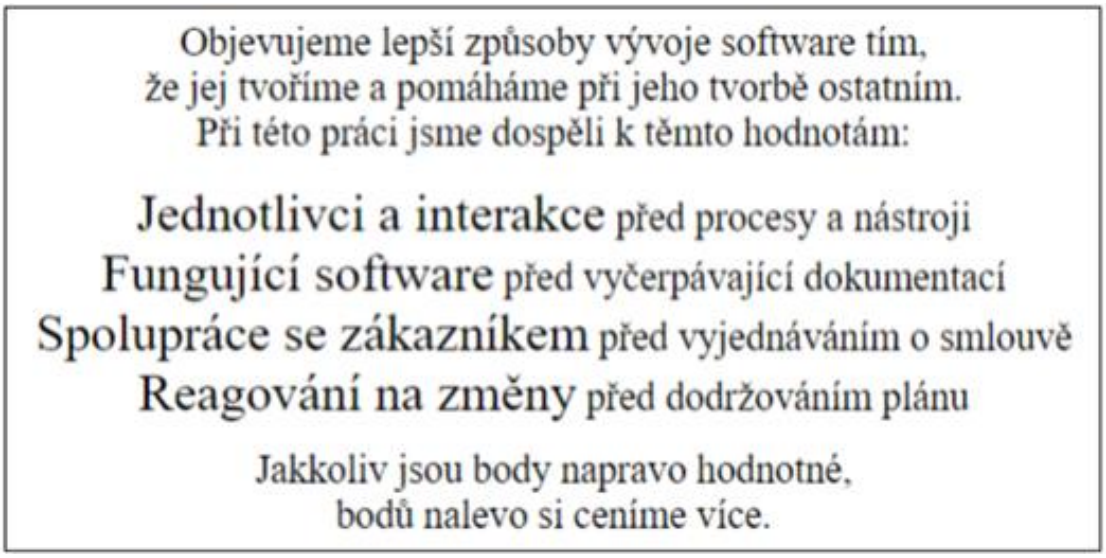
3.4 Agilní metodiky

3.4.1 Manifest agilního programování

Vznik agilního přístupu se datuje od roku 2001, i když většina technik byla používána již předtím. Tehdy se setkali nejvýznamnější představitelé nových přístupů při snaze oprostit vývojový proces od všech zbytečností a sestavili Manifest agilního programování, pod kterým se všichni rovnou podepsali. Současně byla založena Aliance pro agilní vývoj. Každá z agilních metodik je svým způsobem specifická, ale všechny jsou postavené na stejných principech. [5] [8]

Agilní manifest staví na dvou základních principech:

1. Umožnit změnu je mnohem efektivnější, než se jí snažit zabránit
2. Je třeba být připraven reagovat na nepředvídatelné události, protože ty nepochybně nastanou [5]



Obrázek 9 - Agilní manifest (Zdroj: [5])

Agilní manifest deklaruje 10 hlavních principů agilních metodik, které jsou v následujících odstavcích blíže charakterizované:

1. **Nejvyšší prioritou je včas a kontinuálně dodávat software, který zákazníkům přináší hodnotu.**

   Zákazníka nezajímají dokumenty, diagramy nebo integrace stávajících systémů, ale zajímá ho, zda dostane fungující software v každé iteraci a zda poskytovaná funkcionalita uspokojuje jeho potřeby. Rigorózní přístupy předpokládají, že splnění plánu = úspěch projektu = hodnota pro zákazníka. V dnešní době je však nutné, aby hodnota pro zákazníka byla neustále ověřována, protože je předmětem změny.

2. **Změnu požadavku je možné provést i v pozdějších fázích vývoje, protože tím může zákazník získat konkurenční výhodu.**

   Agilní přístup se snaží realizovat změny efektivně a řídit rizika negativních dopadů. Je třeba realizovat krátké iterace a na jejich konci dodávat fungující software. Iterativní vývoj je prosazován i rigorózními metodikami, agilní přístupy však kladou důraz na zkrácení cyklu dodání přírůstků.

3. **Uživatelé a vývojáři spolupracují na projektu denně.**

   Agilní přístupy radikálně mění koncept specifikace požadavku. Východiskem je přesvědčení, že nelze dohodnout a popsat všechny požadavky na začátku projektu. Proto definují na začátku jen hrubé požadavky, které se pak na základě každodenního jednání s klientem zpodrobňují, a dokonce i mění. To zdůrazňuje spoluúčast zákazníků na definování požadavku a přenesení části odpovědnosti za projekt na zákazníka.

4. **Motivovaní jedinci, kteří mají vytvořeny podmínky pro práci a mají podporu vedení, jsou klíčovým faktorem úspěchu projektu.**

   I když jsou využity všechny možné prostředky a nástroje, jsou to nakonec lidé, kteří rozhodují o úspěchu nebo neúspěchu projektu. Lidé musí být vhodně motivováni a projekt musí být podporován všemi zúčastněnými.

5. **Nejefektivnějším způsobem přenosu informací v rámci vývojového týmu je osobní komunikace.**

   Agilním metodikám bývá vytýkána nedostatečná dokumentace. Dokumentace ale není cílem projektu. Smyslem dokumentace je pochopení problému, kterého se mnohem lépe, rychleji a s menšími náklady dosáhne používáním přímých technik komunikace.

6. **Primární mírou úspěchu je fungující software.**

   Ani plnění plánu, ani existující dokumentace návrhu nezajistí úspěch projektu. Je potřeba mít fungující systém.

7. **Agilní procesy předpokládají "zdravý" vývoj.**

   Vývojáři běžně pracují přesčas a to i o víkendech, což zákonitě snižuje jejich produktivitu. Je také potřeba vymezit pracovní prostor, ve kterém zůstane tým v dobré kondici.

8. **Perfektní technické řešení i návrh.**

   Agilní přístupy zdůrazňují kvalitu návrhu, která je potřebná pro realizaci změn. Návrh není samostatnou etapou, která je zcela dokončena před zahájením implementace, ale je to iterativní činnost dělána v průběhu projektu.

9. **Zásadním požadavkem je jednoduchost řešení, tzn. umění maximalizovat množství neudělané práce.**

   Při vývoji softwaru můžeme použít množství postupů a metod. V agilních projektech je kladen důraz hlavně na jednoduché postupy.

10. **Nejlepší architektury, požadavky a návrhy vznikají ze samoorganizujících se týmů.**

    Tento princip zdůrazňuje kreativitu lidí, častou komunikaci a přizpůsobování metodiky. [5]

Níže jsou popsané tyto vybrané agilní metodiky:

1. Scrum
2. Kanban
3. Vývoj založený na testování
4. Extrémní programování
5. Vývoj rychlých aplikací (RAD)
6. Vývoj založený na funkcionalitě

3.4.2 Scrum

Scrum je metodika, v rámci které je možné řešit komplexní adaptivní problémy a zároveň produktivně a kreativně vytvářet produkty s nejvyšší možnou hodnotou.
Scrum je procesní rámec, který se využívá pro řízení práce na komplexních produktech od počátku 90. let.

Scrum není proces, technika či definitivní metoda. Je to spíše rámec, v němž je možné využívat různých procesů a technik. Scrum odhaluje relativní účinnost produktového řízení a pracovních postupů, což umožňuje nepřetržitě zlepšovat produkt, tým a pracovní prostředí.

Metodika Scrum sestává ze Scrum týmů a k nim přidružených rolí, událostí, dokumentů a pravidel. Každá složka v daném rámci splňuje specifický účel a je nezbytná pro úspěch a využívání Scrumu. [2] [8] [11]

Obrázek 10 - Scrum (Zdroj: [25])

Teorie Scrumu

Scrum je založen na teorii empirického řízení procesů. Empirismus tvrdí, že znalosti pocházejí ze zkušeností a rozhodnutí se dělají na základě toho, co už známe. Scrum využívá iterační a přírůstkový přístup k optimalizaci předvídatelnosti a k řízení rizik.

Každou implementaci empirické inspekce procesů podpírají tři pilíře:

1. Transparentnost
2. Kontrola
3. Adaptace

[2] [8] [11]

Transparentnost

Značné aspekty procesu musí být zřetelné pro všechny, kteří odpovídají za výsledek. Pod průhledností rozumíme to, že všechny aspekty budou definované za použití společného standardu. Tím dosáhneme to, že pozorovatelé budou stejně rozumět tomu, co vidí. [2] [8] [11]

Inspekce

Uživatelé Scrumu musí často kontrolovat Scrum dokumenty a pokrok směrem k cíli, aby mohli odhalit nežádoucí odchylky. Inspekce by však neměly být natolik časté, aby bránily výkonu práce. Inspekce jsou nejvíce prospěšné, když jsou svědomitě vykonávané zkušenými inspektory v místě výkonu práce. [2] [8] [11]

Adaptace

Pokud inspektor zjistí, že jeden nebo více aspektů procesu se odchyluje za hranici přijatelných hodnot, a že výsledný produkt bude nepřijatelný, tak se proces nebo produkt musí upravit. Úprava se provede co nejdříve, aby se minimalizovala další odchylka. [2] [8]

Scrum tým

Scrum tým se skládá z Produktového vlastníka (z angl. Product Owner), Vývojového týmu a Scrum Mastera. Scrum týmy jsou samoorganizující a interdisciplinární. Samoorganizující se týmy si sami zvolí, jak nejlépe svou práci provedou. Nejsou vedeny jinými lidmi mimo tým. Interdisciplinární týmy mají všechny potřebné dovednosti, aby práci provedly a nejsou závislé na jiných mimo tým. [2] [8] [11]

Produktový vlastník

Produktový vlastník je zodpovědný za maximalizaci hodnoty produktu, která je výsledkem práce Vývojového týmu. Způsob, jakým to bude dělat, se může výrazně lišit od organizace k organizaci. [2] [8] [11]

Produktový vlastník je jediná osoba odpovědná za řízení Produktového Backlogu. Řízení Produktového Backlogu zahrnuje:

1. Jasné definování položek v Produktovém Backlogu.
2. Seřazení položek v Produktových Backlogu tak, aby se cíle a poslání co nejlépe naplňovaly.
3. Optimalizace hodnoty práce vykonané Vývojovým týmem.
4. Zajištění, aby byl Produktový Backlog viditelný, transparentní a jasný všem a zobrazoval to, na čem bude scrum tým dále pracovat.
5. Zajištění, aby Vývojový tým rozuměl položkám v Produktovém Backlogu v potřebné hloubce.

Produktový vlastník může uvedené úkoly udělat sám, nebo může pověřit Vývojový tým. Odpovědným však zůstává Produktový vlastník. [2] [8] [11]

Vývojový tým

Vývojový tým sestává z odborníků, kteří pracují na dodání přírůstku "Konečného" produktu, který je potenciálně možné uvést na konci každého sprintu. "Hotový" přírůstek se požaduje pro Sprint Review. Pouze členové Vývojového týmu mohou vytvářet přírůstek. Vývojové týmy jsou tak strukturované a natolik zmocněné organizacemi, aby si sami mohly organizovat a řídit svou práci. Výsledná synergie optimalizuje celkovou účinnost a efektivitu Vývojového týmu. [2] [8] [11]

Vývojové týmy mají následující vlastnosti:

1. Organizují se sami. Nikdo, dokonce ani samotný Scrum Master, neříká Vývojovému týmu, jak přeměnit Produktový Backlog na přírůstky s funkcionalitou, kterou bude potenciálně možné nasadit.
2. Vývojové týmy jsou interdisciplinární a jako tým mají všechny dovednosti potřebné k vytvoření produktového přírůstku.
3. Scrum nerozeznává žádné tituly pro členy Vývojového týmu nehledě na to, jakou práci jednotlivec vykonává.
4. Scrum nerozeznává žádné podtýmy v rámci Vývojového týmu, nehledě na oblasti, které je třeba obsáhnout, jako například testování, architektura, operativa nebo podniková analýza.
5. Jednotliví členové Vývojového týmu mohou mít své specializace a oblasti zájmu, avšak zodpovědnost patří vývojovému týmu jako celku. [2] [8] [11]
6.

Scrum Master

Scrum Master je zodpovědný za podporu a pomoc při používání Scrumu. Scrum Masteři toho dosahují tím, že pomáhají každému pochopit teorii, postupy, pravidla a hodnoty Scrumu. Scrum Master je služebníkem a lídrem pro Scrum tým. Scrum Master pomáhá ostatním mimo Scrum tým pochopit, které z jejich interakcí se Scrum týmem jsou užitečné a které ne. Scrum Master pomáhá každému upravit tyto interakce, aby maximalizoval hodnotu vytvořenou Scrum týmem. [2] [8] [11]

Události Scrumu

Ve Scrumu se používají předepsané události k vytvoření pravidelnosti a minimalizaci potřeby setkání, které nejsou v Scrumu definovány. Všechny události jsou časově ohraničené tak, aby každá událost měla maximální délku trvání. Jakmile sprint začne, jeho trvání je pevně stanoveno a nemůže být zkrácen nebo prodloužen. Zbývající události se mohou ukončit kdykoliv, kdy se dosáhne účel daného eventu, čímž se zaručí, že se využije přiměřené množství času, aniž by se v procesu plýtvalo.

Kromě samotného Sprintu, který je souborem všech ostatních událostí, každá událost ve Scrumu je formální příležitostí pro inspekci a adaptaci. Tyto události jsou speciálně navrženy tak, aby umožňovaly rozhodující transparentnost a inspekci. Vynechání kterékoliv události povede ke snížené průhlednosti a promarněné příležitosti pro inspekci a adaptaci. [2] [8] [11]

1. **Sprint**

Srdcem Scrumu je Sprint. Je časově ohraničený, zpravidla na týden až měsíc. Během Sprintu je vytvořen "Hotový", použitelný přírůstek produktu, který je možné nasadit do produkce. Sprinty mají v průběhu vývoje pevnou délku trvání. Nový Sprint začíná okamžitě po skončení předchozího Sprintu.

Během Sprintu:

a) se neprovádějí žádné změny, které by ohrozily cíl sprintu,
b) cíle neklesají na kvalitě,
c) rozsah prací se může znovu upřesnit a prodiskutovat, mezi produktovým vlastníkem a vývojovým týmem. [2] [8] [11]

Zrušení Sprintu

Sprint je možné zrušit dříve, než uplyne jeho časové ohraničení. Pouze produktový vlastník má pravomoc Sprint předčasně ukončit, ačkoli by tak mohl učinit pod vlivem dalších zúčastněných stran, Vývojového týmu nebo Scrum Mastera.

Sprint by se mohl předčasně ukončit v případě, že by se cíl Sprintu stal zastaralým. K tomu může dojít v případě, že společnost změní směřování, nebo se změní podmínky na trhu, či technologie. Obecně by se Sprint měl ukončit v případě, že za daných okolností již nemá

smysl v něm pokračovat. Kvůli krátkému trvání Sprintu má však takové ukončení smysl pouze výjimečně. [2] [8] [11]

Plánování Sprintu

Plán Sprintu se vytváří za spolupráce všech členů Scrum týmu. Plánování Sprintu je časově ohraničené na maximálně osm hodin, v případě jednoměsíčního Sprintu. V případě kratších Sprintů jsou události obvykle kratší. Scrum Master zajistí, že se plánování uskuteční a všichni jeho účastníci rozumí jeho významu. Scrum Master učí Scrum tým dodržovat časové ohraničení. Při plánování jsou časově ohodnoceny všechny prioritizované úkoly. [2] [8] [11]

2. **Denní Scrum (Stand up)**

Denní Scrum je událost časově ohraničená na přibližně 15 minut a účastní se ho hlavně Vývojový tým. Denní Scrum se v průběhu Sprintu koná každý den. Během něj si Vývojový tým naplánuje práci na dalších 24 hodin. Optimalizuje se tím týmová spolupráce a výkon, protože se jednak zkontroluje práce provedená od posledního Denního Scrumu, a na straně druhé se odhadne nadcházející práce. Za účelem snížení komplexity se Denní Scrum koná každý den ve stejnou dobu a na stejném místě.

Vývojový tým používá Denní Scrum na kontrolu průběhu plnění cíle Sprintů a toho, jaký je trend v souvislosti s dokončováním prací v Sprint Backlogu. Denní Scrum zvyšuje pravděpodobnost toho, že vývojový tým splní cíl Sprintu. Každý den by měl mít vývojový tým představu, jak zamýšlí spolupracovat jako samoorganizující se tým, dosáhnout tím cíl sprintu a do konce Sprintu vytvořit očekávaný přírůstek. [2] [8] [11]

3. **Sprint Review**

Sprint Review se koná na konci Sprintu za účelem inspekce přírůstků a v případě potřeby, také adaptace Produktového backlogu. Během Sprint Review, Scrum tým a ostatní zájmové strany společně posuzují to, co se v průběhu Sprintu udělalo. Na základě toho, co se vyvinulo a změn provedených v Produktovém Backlogu, zúčastnění společně posoudí, co by se dále mělo udělat, aby se hodnota ještě více optimalizovala. Jde o neformální

setkání, nikoli status a přírůstek se zde prezentuje za účelem získání zpětné vazby a prohloubení spolupráce.

Výsledkem Sprint Review je revidovaný Produktový Backlog, který definuje položky Produktového Backlogu, které budou pravděpodobně vybrány do dalšího Sprintu. Produktový Backlog se zároveň může celkově upravit a to tak, aby se mohly využít nové příležitosti. [2] [8] [11]

4. **Retrospektiva Sprintu**

Retrospektiva Sprintu nabízí příležitost pro Scrum tým, aby se sám zkontroloval a připravil plán zlepšení, které zavede v následujících Sprintech.

Retrospektiva Sprintu se koná po Sprint Review a před dalším plánováním Sprintu. V případě jednoměsíčního Sprintu, toto setkání trvá nejvýše tři hodiny. V případě kratších sprintu jsou události obvykle kratší. Scrum Master zajistí, že se událost uskuteční a všichni jeho účastníci rozumí jeho významu.

Scrum Master vybízí Scrum tým, aby v rozsahu omezení procesního rámce Scrumu, zlepšil jejich proces vývoje a postupy, čímž by se pro další Sprint staly účinnější a příjemnější. Během každé Retrospektivy Sprintu, pokud je to vhodné, a ne v rozporu s produktovými či organizačními standardy, Scrum tým navrhne způsoby, jakými zvýší kvalitu produktu. Může jít o zlepšení pracovních procesů či adaptaci definice "Konečného".

Na konci každé retrospektivy by měl mít Scrum tým identifikována vylepšení, která bude v dalších sprintech implementovat. Implementace těchto vylepšení v následujících sprintech představuje adaptaci vyplývající z inspekce Scrum týmu jako takového. [2] [8] [11]

Scrum dokumenty

Dokumenty (z angl. Artifacts) ve Scrumu znázorňují práci nebo hodnotu a vytvářejí transparentnost a příležitosti pro inspekci a adaptaci. Dokumenty definované ve Scrumu jsou specificky navrhované tak, aby maximalizovaly transparentnost klíčových informací, díky čemuž všichni rozumějí dokumentům stejně. [2] [8] [11]

Produktový Backlog

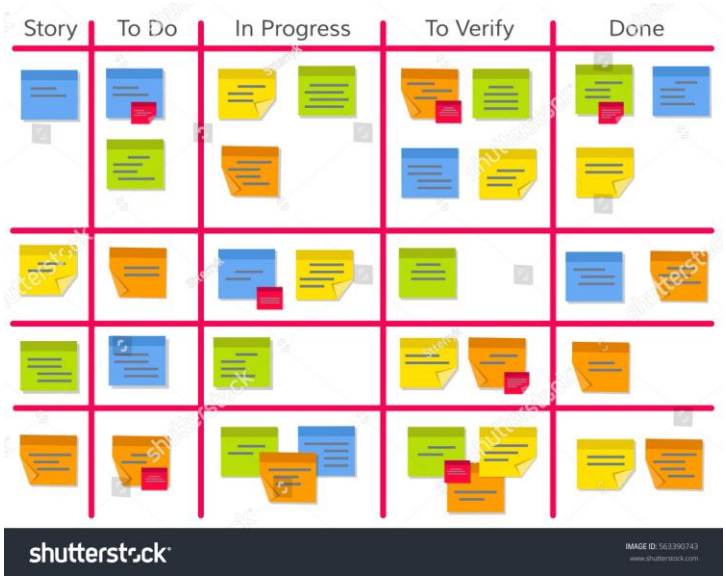
Produktový Backlog je uspořádaný seznam všeho, o čem víme, že bude v produktu zapotřebí. Je to jediný zdroj požadavků, vůči němuž by se dělaly jakékoli změny na produktu. Produktový vlastník je zodpovědný za Produktový Backlog včetně jeho obsahu, dostupnosti a uspořádání.

Produktový Backlog není nikdy úplný. Jeho počáteční verze je náčrtem prvotně známých a nejlépe pochopených požadavků. Produktový Backlog se vyvíjí tak, jak se vyvíjí produkt a prostředí, ve kterém se bude používat. Produktový Backlog je dynamický, neustále se mění, aby obsahoval všechno to, díky čemuž bude produkt vhodný, konkurenceschopný a užitečný. Pokud existuje produkt, existuje i jeho Produktový Backlog. [2] [8] [11]

Sprint Backlog

Sprint Backlog je soubor položek Produktového Backlogu vybraných pro Sprint, spolu s plánem na dodání produktového přírůstků a dosažení cíle Sprintu. Sprint Backlog je odhad vývojového týmu ohledně toho, jaká funkcionalita bude součástí dalšího přírůstku a prací potřebných na přeměnu dané funkcionality na "Hotový" přírůstek.

Sprint Backlog umožňuje vidět všechny práce, které Vývojový tým identifikuje jako nezbytné pro splnění Cíle Sprintech. Pro zajištění neustálého zlepšování se, zahrnuje nejméně jedno zlepšení procesu s vysokou prioritou identifikované během předchozí retrospektivy. [2] [8] [11]



Obrázek 11 - Scrum tabule (Zdroj: [26])

Přírůstek

Přírůstek (z angl. Increment) je součet všech položek Produktového Backlogu dokončených během příslušného Sprintu a hodnoty přírůstků všech předchozích Sprintů. Na konci Sprintu, musí být nový přírůstek "Hotový", což znamená, že musí být v použitelném stavu a splňovat definici "Konečného". Přírůstek je zkontrolovatelný hotový kus práce na konci Sprintu, který podpírá teorii empirismu. Přírůstek je krokem k vizi nebo cíli. Přírůstek musí být v použitelném stavu bez ohledu na to, zda se Produktový vlastník rozhodne ho použít. [2] [8] [11]

Definice "Konečného"

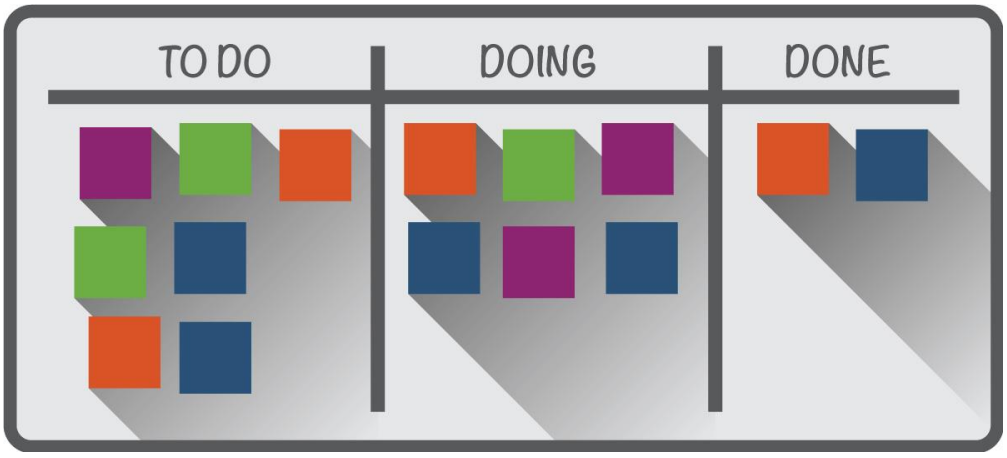
Když je položka Produktového Backlogu nebo přírůstek označený jako "Hotový", každý musí rozumět tomu, co "Hotové" znamená. Ačkoli vnímání "Konečného" se může značně lišit od jednoho Scrum týmu k druhému, členové Scrum týmu musí mít společné chápání toho, co znamená to, že práce je hotová, aby se zajistila transparentnost. Jde o definici "Konečného" daného Scrum týmu a používá se na zhodnocení toho, zda je práce na produktovém přírůstku hotová. [2] [8] [11]

3.4.3 Kanban

Kanban je vizuální systém pro správu práce v průběhu procesu. Kanban vizualizuje jak proces (pracovní postup), tak aktuální práci, která prochází tímto procesem. Cílem metodiky Kanban, je identifikovat potenciální překážky ve vašem procesu a opravit je tak, aby práce mohla protékat nákladově efektivním způsobem s optimální rychlostí nebo propustností. [8] [13] [27]

Kanban principy a praktiky

Metoda Kanban sleduje soubor zásad a postupů pro řízení a zlepšování toku práce. Jedná se o evoluční, nedestruktivní metodu, která podporuje postupné zlepšování procesů organizace. Ačkoli Kanban je označení pro samotnou tabuli, kde jsou zobrazeny jednotlivé úkoly, dá se na něj pohlížet i jako na metodiku, vzhledem k jeho doporučením o pracovním procesu. [8] [13] [27]



Obrázek 12 - Kanban tabule (Zdroj: [27])

Základní principy:

1. **Začíná tím, co se právě v organizaci dělá**

   Metoda Kanban zdůrazňuje, že okamžitě nezmění stávající nastavení ani proces. Kanban musí být aplikován přímo na aktuální pracovní postup. Jakékoli potřebné změny se mohou objevit postupně během používání. [8] [13] [27]

2. **Podporuje drobné evoluční změny**

   Kanban nabádá, aby byly provedeny spíše drobné přírůstkové změny než radikální změny, které by mohly vést k odporu v týmu a organizaci. [8] [13] [27]

3. **Z počátku respektuje současné role, odpovědnosti a názvy pracovních pozic**

   Na rozdíl od jiných metod, Kanban neukládá sám o sobě žádné organizační změny, takže není nutné provádět změny ve stávajících funkcích, které mohou fungovat dobře. Tým bude společně identifikovat a provádět všechny potřebné změny. Tyto tři zásady pomáhají organizacím překonat typický emocionální odpor a strach ze změn, které obvykle doprovázejí změny v organizaci. [8] [13] [27]

4. **Povzbuzuje vedení na všech úrovních**

   Kanban podporuje nepřetržité zlepšování na všech úrovních organizace a říká, že vůdčí akty nemusí pocházet pouze od vedoucích pracovníků. [8] [13] [27]

Základní postupy Kanban metody

1. **Vizualizace toku práce:**

Jedná se o zásadní první krok k přijetí a implementaci metody Kanban. Procesní kroky jsou zobrazeny, buď na fyzické tabuli, nebo na elektronické kanban tabuli. V závislosti na složitosti procesu a pracovního procesu může být Kanban tabule velmi jednoduchá až velmi komplikovaná.

V klasickém Kanban modelu jsou na Kanban tabuli minimálně tři sloupce

a) **To Do:** Tento sloupec uvádí úkoly, na kterých se ještě nezačalo pracovat.
b) **In progress:** Skládá se z probíhajících úkolů.
c) **Done:** Skládá se z dokončených úkolů. [8] [13] [27]

2. **Omezení WIP (Work in progress)**

Omezení nedokončených úkolů (WIP) je zásadní pro implementaci systému Kanban. Omezením WIP je doporučeno, aby tým nejprve dokončil daný úkol před zahájením nového. Úkol, který právě probíhá, musí být dokončen a označen jako „Hotový“. To vytváří kapacitu týmu v systému. Zpočátku nemusí být snadné rozhodnout, jaké by měly být vaše limity WIP. [8] [13] [27]

3. **Řízení toku**

Řízení a zlepšování toku je základem metody Kanban po zavedení prvních dvou bodů. Metoda Kanban pomáhá řídit tok zvýrazněním různých fází pracovního postupu a stavu práce v každé fázi. V závislosti na tom, jak je definován pracovní postup a že jsou nastaveny limity WIP, je možné sledovat buď hladký tok v rámci hranic WIP, nebo různé nesrovnalosti v daném procesu. [8] [13] [27]

4. **Zjednodušení procesu**

V rámci vizualizace procesu je smysluplné také explicitně definovat a vizualizovat zásady (procesní pravidla nebo pokyny) o tom, jak je práce vykonávaná. Formulováním

explicitních pokynů pro proces se vytváří společný základ pro všechny účastníky, aby pochopili, jak provádět jakýkoli druh práce v systému. [8] [13] [27]

5. **Implementujte zpětné vazby**

Zpětné vazby jsou nedílnou součástí každého dobrého systému. Metoda Kanban povzbuzuje a pomáhá provádět zpětné vazby různých typů, např. přehledy ve vašem pracovním postupu, metriky a výstupy a řadu vizuálních podnětů, které poskytují nepřetržitou zpětnou vazbu o postupu práce. [8] [13] [27]

6. **Zlepšení spolupráce**

Metoda Kanban je evoluční proces zlepšování. Pomáhá přijímat malé změny a postupně se zlepšovat tempem a velikostí objemu práce, které může tým snadno zvládnout. [8] [13] [27]

3.4.4 Vývoj založený na testování (TDD)

Je to proces vývoje softwaru, který se opírá o opakování velmi krátkého vývojového cyklu. Požadavky se změní na velmi specifické testovací případy, pak je software vyvinut, aby vyhověl pouze nově napsaným testům. To je v rozporu s vývojem softwaru, který umožňuje přidat software, který nemusí splnit požadavky. Programátoři také aplikují koncept na zdokonalení a ladění staršího kódu, vyvinutého staršími technikami. [28]

Cyklus vývoje založeného na testování:

1) Přidaní/vytvoření testu

Při vývoji řízeného testováním začíná každý nový požadavek psaním testů. Je napsán takový test, který definuje funkci nebo vylepšení funkce, která by měla být poměrně jednoduchá. Může to být i upravená verze stávajícího testu. [28]

2) Spuštění všech testů a zjištění, zda nový test selže

Tím že se pustí nový test, na který ještě dosavadní verze kódu není přípravená, se vývojář ujistí, že daná funkcionalita již není implementovaná. Je tedy očekáván neúspěch testu. [28]

3) Napsání kódu

Dalším krokem je napsat kód, který zapříčiní, že test již bude úspěšně procházet. Nový kód napsaný v tomto stádiu není dokonalý a může například projít testem nechtěným způsobem. To je přijatelné, protože v kroku 5 bude dále vylepšován. V tomto okamžiku je jediným účelem psaného kódu test úspěšně splnit. [28]

4) Opětovné spuštění všech testů

Pokud nyní proběhnou všechny testovací případy správně, tak si programátor může být jistý, že nový kód splňuje všechny požadavky a neporušuje žádné existující funkce. V opačném případě musí být nový kód přepsán do podoby, aby se tak stalo a vše prošlo bez problémů. [28]

5) Refactoring kódu

Rostoucí kódová základna musí být pravidelně vyčištěna během vývoje. Nový kód může být přesunut z místa, kde byl vhodně vložen pro absolvování testu, na místo, kam logicky patří. Existují specifické a obecné pokyny pro refaktorování a pro vytváření čistého kódu. Neustálým opětovným spuštěním testovacích případů během každé fáze refaktorování se může vývojář přesvědčit, že nejsou porušeny žádné existující funkce. [28]

6) Vše znovu opakovat

Počínaje dalším novým testem se cyklus opakuje, aby se funkce dále vylepšovala.

Obrázek 13 - Vývoj založený na testování (Zdroj: [28])

3.4.5 Extrémní programování (XP)

Extrémní programování (označované také zkratkou XP), je metoda agilního programování, která byla vytvořena počátkem 90. let. Tato metoda předepisuje určité činnosti všem účastníkům vývojového procesu. Jedná se o rigorózní činnosti, v "extrémní“ formě. Díky těmto vlastnostem je extrémní programování schopné přizpůsobit se a dodávat vysoce kvalitní software. [7] [29]

Základní principy

1) Jednoduchost

V XP se vytváří kód, který splňuje tzv. Minimální funkční implementaci. To znamená, že se tvoří pouze nutná část kódu potřebná pro splnění požadavků a nic víc. Nikdy se nevytváří kód, který je pouze předpokladem pro využití do budoucna, protože XP předpokládá, že se tyto požadavky mohou měnit každým dnem. [7] [29]

2) Komunikace

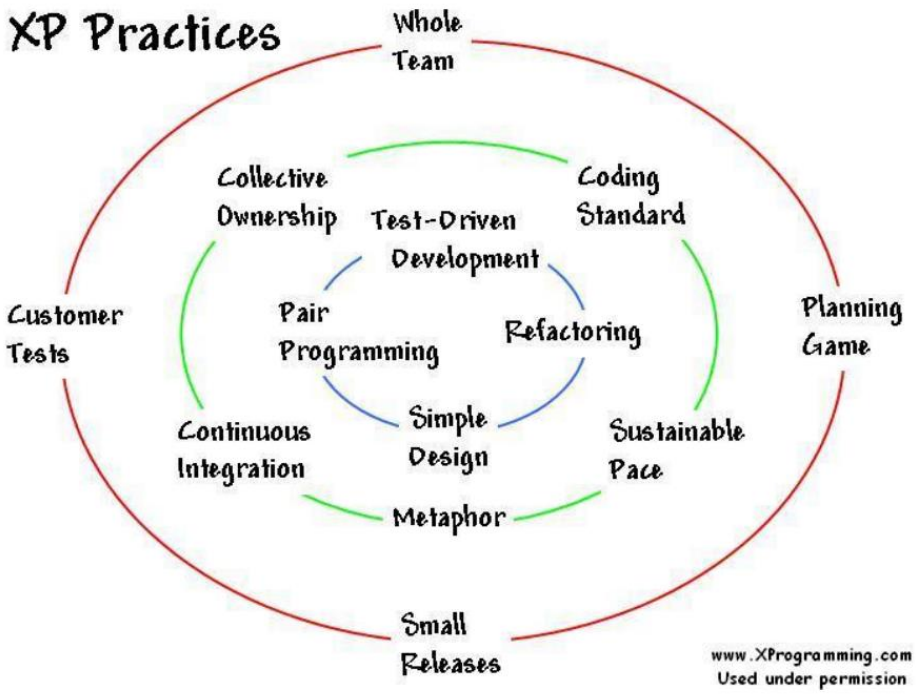
Kvalitní komunikace mezi všemi zúčastněnými stranami je důležitá. Komunikace mezi programátorem a zákazníkem zajistí software, který bude zákazníkovi vyhovovat, a který bude navržen přímo na míru. Programátor musí také komunikovat s manažerem, aby se zajistilo předání všech důležitých informací o stavu projektu a zjednodušilo se tak rozhodování. Komunikace programátor - programátor je nejdůležitější a věnuje se jí více principů XP. [7] [29]

3) Zpětná vazba

Zpětná vazba má v XP má více podob. Pro programátory jsou to jednotkové a integrační
testy prováděné v dostatečně krátkých časových intervalech. V podstatě se jedná o vývoj založený na testování. Tyto testy slouží k ověření funkcionality kódu a jako ověření toho, že změny, které byly v kódu učiněny, nenarušují funkcionalitu jiných částí programu. XP také podporuje další praktiky, jako je například refractoring. [7] [29]

4) Odvaha

„Odvaha“ programátorů se v XP metodice objevuje na více místech. Je běžnou praktikou, že oprava určitého komplexního problému způsobí další související chyby. Pustit se do takových významných změn vyžaduje odvahu a důvěru v dobře napsané jednotkové testy. Také je v XP běžnou praktikou zahození kódu, u kterého je problém s integrací, protože se věří, že následné nalezení optimálního řešení bude o hodně jednodušší. [7] [29]



Obrázek 14 - Extrémní programování (Zdroj: [29])

Základní praktiky:

1) **Obchodní praktiky**

a) **Plánovací hra**

Na procesu spolupracují zákazník a vývojový tým a společně plánují vývojový proces jednotlivých iterací. Jednotlivé zákaznické požadavky popisují tzv. Uživatelské scénáře. Vývojový tým tyto požadavky rozebere a odhadne implementační detaily (čas, cena), na základě čeho si zákazník určuje prioritu jednotlivých požadavků. [7] [29]

b) **Zákazník na pracovišti**

Na projektu pracuje i osoba (nebo tým) představující zákazníka, který úzce spolupracuje na řešení, určování priorit a objasňování požadavků.

c) **Vydávání malých verzi**

Nová verze programu se vydává tak často, jak je to jen možné. Jednotlivé funkcionality se po naprogramování a otestování okamžitě integrují do produkčního kódu. Získává se tak rychlá zpětná vazba od zákazníka. [7] [29]

d) Metafora

Při komunikaci v týmu se používají jednoduše zapamatovatelné metafory, které pomáhají ke snažšímu pochopení problému. [7] [29]

2) Týmové praktiky

a) Párové programování

Párové programování je charakteristická vlastnost XP. V jednu chvíli pracují na kódu dva programátoři. Řidič - programátor, který píše kód a navigátor - přísedící programátor, spoluanalyzuje a řeší problémy. Hledají nejlepší řešení a komunikují spolu o optimalizaci a implementačních detailech čímž se vytváří zpětná vazba již při vytváření kódu. Programátoři si tyto role střídají. [7] [29]

b) Společné vlastnictví kódu

V XP neexistuje rozdělení odpovědnosti za jednotlivé částí kódu. Každý programátor může měnit kteroukoli část kódu. Jediná potřebná věc je, držet se jednotkových testů. [7] [29]

c) Standardy kódu

Tým programátorů se dohodne na předem určených pravidlech týkajících se čistoty a kvality kódu. Všechny aspekty psaní kódu jako např. odsazování, umístění závorek apod. jsou schválené každým členem týmu. Tato dohoda zajistí jednotnost kódu vytvořeného více programátory, což v praxi znamená že programátoři nebudou mít problém pracovat na kódu někoho jiného, protože si nebudou muset zvykat na jeho styl programování. [7] [29]

d) Udržitelné tempo

Extrémní programování se opírá o tvrzení, že pokud jsou programátoři unavení, tvoří méně kvalitní kód a produkují více chyb. Proto je potřeba, již ze začátku nastavit takové tempo postupu, které bude možné udržet během celého projektu. Přibližná týdenní pracovní doba programátora je 40 hodin a přesčasy jsou nežádoucím jevem. [7] [29]

3) Programovací praktiky

a) Průběžná integrace

Nově naprogramované funkce jsou do produkčního kódu integrovány co nejdříve, minimálně jednou denně. Přidáním každé funkce následuje ověření funkčnosti a stability kódu pomocí jednotkových a integračnímu testů. [7] [29]

b) Jednoduchý návrh

Extrémní programování předpokládá změny požadavků zákazníka. Tato myšlenka tvrdí, že navrhovat funkce do budoucna není efektivní. Programátor by se při vývoji měl soustředit jen na implementaci funkcionality pro konkrétní iteraci. [7] [29]

c) Refactoring kódu

Neustálé vylepšování kvality kódu, který se opírá o kvalitně navržené a napsané testy. Ty umožňují uskutečnit rozsáhlejší změny bez obav o stabilitu a narušení funkcionality. Eliminují se duplicity, zvyšuje čitelnost, zlepšuje vnitřní návrh systému. Díky společnému vlastnictví je každý programátor oprávněn opravovat jakoukoli část kódu. [7] [29]

d) Testování

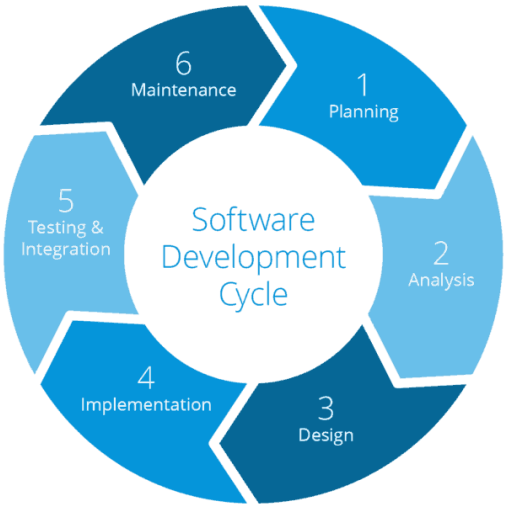
Při XP se využívá tzv. Programování řízené testy. [7] [29]

3.4.6 Rapid Application Development (RAD)

Rychlý vývoj aplikací je vývojový model, který upřednostňuje rychlé vytváření prototypů a rychlou zpětnou vazbu místo dlouhých vývojových a testovacích cyklů. Díky rychlému vývoji aplikací mohou vývojáři rychle provést více iterací a aktualizací softwaru, aniž by bylo nutné začít vytvářet vývojový plán od začátku. [30]

Kroky v rychlém vývoji aplikací:

Přestože se RAD v průběhu let změnil, tyto čtyři základní kroky se v průběhu let nezměnily.



Obrázek 15 - Rapid Application Development (Zdroj: [30])

1) Definování požadavků

Na začátku se metodika odlišuje od rigorózních modelů vývoje softwaru. Nevyžaduje, aby se komunikovalo s koncovými uživateli a získal se podrobný seznam přesně definovaných požadavků hned ze začátku. Místo toho se požaduje velmi málo specifikovaný rozsáhlý požadavek. Rozsáhlá povaha požadavků dovoluje, upřesnit konkrétní požadavky v různých bodech vývojového cyklu. [30]

2) Prototyp

Právě tam probíhá skutečný vývoj. Místo toho, aby se splnily přísné požadavky, vývojáři vytvářejí prototypy s různými funkcemi tak rychle, jak jen mohou. Tyto prototypy jsou pak ukázány klientům, kteří rozhodují, co se jim líbí a co by chtěli přidat. [30]

3) Zpětná vazba

V této fázi je udělena zpětná vazba na všechno co bylo doposud představeno koncovým uživatelům. Zpětná vazba se dává na každý aspekt prototypu a to včetně uživatelského rozhraní. Získávají se informace o tom, co v tuhle chvíli splňuje očekávání a co je třeba změnit. [30]

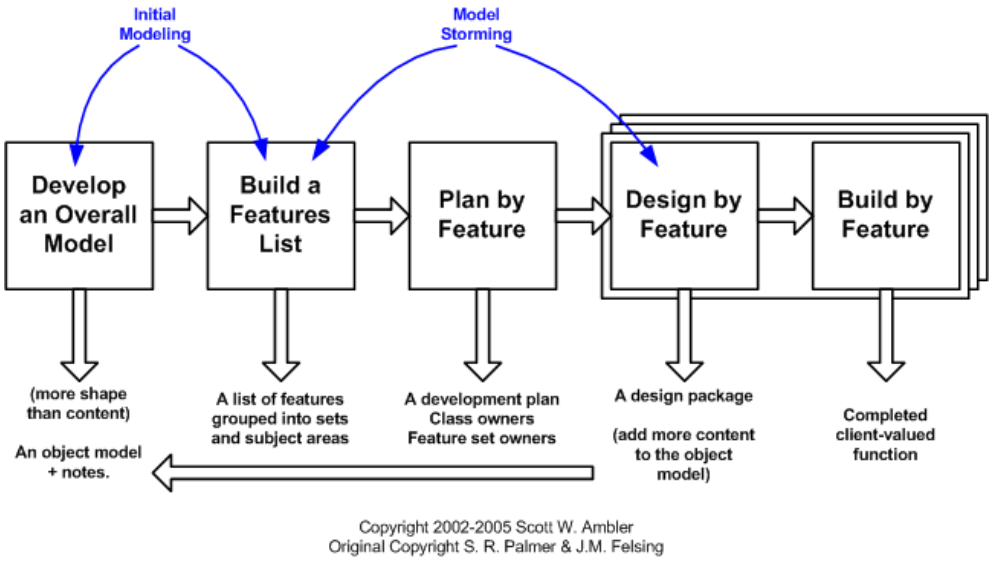
3) Dokončení

V tuhle chvíli jsou všechny funkcionality a vzhled dokončen a připravován na předání klientovi. Stabilita, použitelnost a udržovatelnost mají mimořádný význam před tím, než jsou doručeny klientovi. [30]

3.4.7 Feature driven development (FDD)

Vývoj založený na funkcionalitě je iterativní a přírůstkový proces vývoje softwaru. FDD kombinuje řadu osvědčených postupů uznávaných v průmyslu do soudržného celku. Tyto postupy jsou řízeny z hlediska klienta s ohledem na funkčnost. Jeho hlavním úkolem je poskytovat hmatatelný pracovní software včas a opakovaně.

FDD se skládá z pěti základních činností. Pro přesné hlášení stavu a sledování projektu vývoje softwaru jsou definovány milníky, které označují pokrok dosažený v každé funkci. Tato část poskytuje přehled o aktivitách na vysoké úrovni. Během prvních dvou sekvenčních činností je vytvořen celkový model. Poslední tři aktivity jsou iterovány pro každou funkcionalitu. [6] [31]



Obrázek 16 - Feature driven development (Zdroj: [31])

1) **Vytvoření celkového modelu**

V tomto prvním procesu, FDD vytváří tlak na týmy, aby vytvořily objektový model daného problému. Odlišně od ostatních, modelování FDD je skrze všechny funkcionality, iterativní a podporuje spolupráci. Členové týmu pracují společně na vytvoření modelu pro oblast domény a za pomoci hlavního architekta. Cílem je, aby různé týmy navrhovaly různé modely a po jejich kontrole byla vybrána ta nejlepší možnost. Nakonec bude model oblasti domény vložen do celkového modelu. [6] [31]

2) **Vytváření seznamu funkcionalit**

V této fázi dochází k porovnání seznamu funkcionalit s aktuálním Produktovým Backlogem. Po dokončení celkového modelu, založeného na znalostech získaných během této fáze, je nutné identifikovat funkcionality, které jsou pro klienta cenné a které v podstatě povedou směr projektu. Funkcionality by neměly trvat déle než dva týdny. [6] [31]

3) **Plánování podle funkcionalit**

Třetí fáze je o plánování, v jakém pořadí budou implementovány jednotlivé funkcionality. Sady funkcionalit jsou pak přiřazeny programátorům. Je zřejmé, že při plánování zohledňujeme různé aspekty, jako jsou rizika, závislosti, složitost, vytížení týmu atd. [6] [31]

4) **Návrh podle funkcionality**

Stejně jako u všech procesů využíváme znalosti získané z prvního modelového procesu. Hlavní programátor přebírá odpovědnost za výběr skupiny funkcionalit, které by měly být dále rozvíjeny. Odborníci na dané oblasti budou posléze zodpovědní za analýzu a navrhování řešení pro každou funkcionalitu. [6] [31]

5) **Vývoj podle funkcionalit**

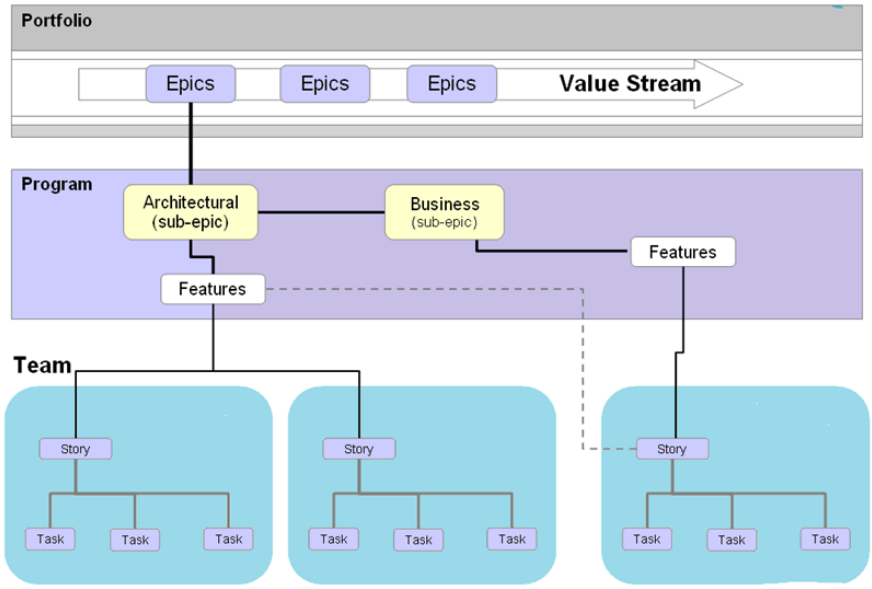
V posledním kroku přichází na řadu samotné programování funkcionality. Každý programátor poté zkontroluje svojí část kódu pomocí jednotkových testů a pokud vše proběhlo v pořádku, tak je funkcionalita schválena a nasazena.

3.4.8 Scaled Agile Framework (SAFe)

SAFe je komplexním přístupem k agilnímu vývoji produktu spolu se zásadami Lean vývoje. Podobně jako DSDM se snaží poskytnout odpovědi a návody, jak postupovat komplexně při malých i velkých projektech. často tedy staví na principech jako Kanban a Scrum. Implementace však některé prvky upravuje. [9] [32]

Přehled metodik a standardů projektového managementu je součásti metodiky. SAFe je koncipováno jako modulární a jednotlivé součásti mohou a nemusí společnost podle své potřeby implementovat. Jde o velmi mladou metodiku, její první verze se poprvé oficiálně objevila roku 2011 na základě myšlenek Deana Leffingwella.

Aktuálním je verze Safe označována jako 4.0. Safe deklaruje, že podle případových studií (Scaled Agile framework, 2017) zvyšuje produktivitu o 20-50%, snižuje čas potřebný na dodání produktu o 30-75%, o více než 50% snižuje chybovost a nakonec, ale ne v poslední řadě, motivuje zaměstnance. Množství kritiků studie zpochybňuje, právě kvůli krátké historické implementaci. [9] [32]



Obrázek 17 - Scaled Agile Framework (Zdroj: [32])

Existují 4 hodnoty, na kterých je SAFe postaveno:

1. **Srovnání (Alignment)**

Celková mise a vize společnosti by se měla rozpadat do jednotlivých týmů a úloh jednotlivců. Souhrnný přínos je důležitější než jednotlivé části. [9] [32]

2. **Vnořená kvalita (Built-in quality)**

Velké systémy jsou ekonomicky náchylnější než jejich jednotlivé části. Každá součást řešení musí dosahovat požadovanou kvalitu. [9] [32]

3. **Průhlednost (Transparency)**

Úplná transparentnost je nutná zejména při velkých projektech, kdy se jednotlivé funkcionality a týmy navzájem ovlivňují. Pokud nestavíme na faktech, ale na domněnkách, či neověřených informacích, je problémem dostatečně brzy zachytit riziko a eliminovat jej. [9] [32]

4. **Exekuce programu (Execute)**

Jen správně implementována programová úroveň Safe, která zahrnuje pravidelnou a spolehlivou dodávku, vede k úspěšnému projektu v SAFE. [9] [32]

Základní nekompromisní podmínky, které SAFE pro správné fungování vyžaduje, vycházejí ze známých Lean a Agile principů. celkově jejich uvádí 9. [9] [32]

1. **Aplikace ekonomického pohledu (Take an economic view)**

Při každém rozhodnutí je nutné vzít do kontextu i ekonomický pohled. Příkladem je cena za dodávku softwaru, cena za případné zpoždění, návratnost projektu a jiné. [9] [32]

2. **Aplikace systémového myšlení (Apply system thinking)**

Hlavní myšlenkou je, že nový software vzniká na základě potřeby automatizovat, zda nahradit pomalou a neefektivní práci pracovníků. Ti si v práci vyvinuli určitý systém, který je třeba vzít v úvahu při plánování a návrhu, nejen globální pohled. [9] [32]

3. **Předpoklad variability a zachování možností (assume variability; preserve options)**

Zřejmým agilním principem je snaha vyhnout se zaslepení původním návrhem, pokud nové informace žádají změnu produktu, a tedy i úpravu plánů, nákladů atd. Bez jisté míry akceptované variability by mohl být výsledný produkt nerentabilní, protože ztratí své opodstatnění kvůli změně okolního prostředí. [9] [32]

4. **Budování inkrementálně s rychlými integračními cykly (Build incrementally with fast, integrated learning cycles)**

Pravidelné malé inkrementy a jejich průběžné nasazení do produkce poskytuje projektovému týmu průběžnou zpětnou vazbu. Podmínkou je možnost na jejím základě produkt upravit v další iteraci. [9] [32]

5. **Objektívní zhodnocení po jednotlivých milnících (Base milestones on objective evaluation of working systems)**

Po každém inkrementu, který identifikujeme jako milník projektu, je třeba z každého hlediska kriticky zhodnotit pokrok projektu a případně upravit jeho další vývoj. [9] [32]

6. **Vizualizace a limitace množství rozpracovaných úkolů, jejich velikost a prostoje v procesu (Visualize and limit WIP, reduce batch sizes, and manage queue Lenghts)**

Cílem je dosáhnout kontinuální a vyrovnaný tok úkolů přes jejich životní cyklus od vytvoření požadavku až po úspěšnou integraci do produkčního systému. Úkoly by měly být co nejmenší, aby byly rychle zpracovatelné, čímž nebudou dlouho zdržovat daný zdroj i v době, když není využit celý jeho potenciál. Nakonec je třeba eliminovat čas, který úloha stráví čekáním na zdroj. [9] [32]

7. **Aplikace pravidelnosti a Synchronizace medzitýmového plánování (Apply cadence, synchronize with cross-domain planning)**

Pravidelný rytmus dodávání inkrementů pomáhá zjednodušit plánování a zpřehlední jednotlivé návaznosti mezi inkrementy i mezi týmy. [9] [32]

8. **Identifikovat a podpořit vnitřní motivaci (Unlock the intrinsic motivation of knowledge workers)**

Pracovníkům je žádoucí poskytnout optimální podmínky pro práci. Je třeba redukovat negativní vlivy, poskytnout autonomii v řešení konkrétních problémů, čímž dosáhneme větší zapojení a pocit spoluvlastnictví. [9] [32]

9. **Decentralizovat rozhodování (decentralizovat decision-making)**

Rychlá dodávka nových funkcionalit vyžaduje rychlé rozhodování. Jakýkoliv nutný eskalační proces způsobuje zdržení. Je třeba posílit autonomii a schopnost rozhodovat v rozumné míře i mezi běžnými členy týmů. [9] [32]

Existují dva druhy implementace SAFe:

1. **Tříúrovňový SAFe**

je vhodný pro projektové týmy o velikosti menší než 100 lidí, pracujících na stejném projektu nebo na více malých projektech, kde není nutná vysoká míra kolaborace.

2. **Čtyřúrovňový SAFe**

je vhodný pro projekty velkého rozsahu. SAFe zavádí pojem ART - Agile release train, (agilní vlak dodávky inkrementů). Je to "tým týmů", pracujících na projektu, které jsou spolu schopny plánovat, zavázat se k dodávce inkrementů a provést ji. Zahrnuje všechny potřebné role. [9] [32]

3.4.9 Large-Scale Scrum (LeSS)

LeSS začal být vyvíjen v roce 2005. Less je postaven na principech Scrumu a přináší návody pro aplikaci Scrumu pro větší organizace, velké projekty. LeSS je Scrum aplikován do prostředí vývoje velkých projektů. LeSS přináší řešení jak aplikovat principy, účel a části Scrumu do kontextu většího škálování tohoto procesu směrem k agilnímu řízení projektu co nejjednodušeji. LeSS využívá Scrum pro řízení jednotlivých týmů a následně řeší, jak jednotlivé Scrum týmy efektivně koordinovat a řídit jejich funkci skrz všemi komponentami a skrz všemi funkcemi. [33]

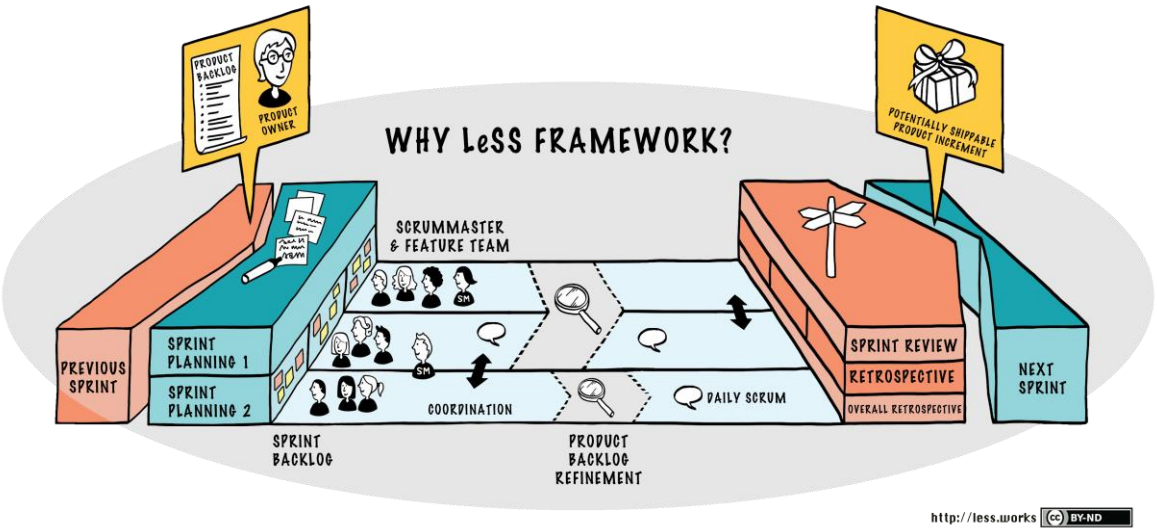
LeSS - Up to eight teams

První rámec LeSS je klasický princip Scrumu postavený pro jednoho vlastníka produktu (Product owner) určeného pro 2 až 8 týmů. Společně dodávají jeden produkt, který prezentují zákazníkům. Týmy jsou stejně jako v klasickém Scrumu stabilní a sebeorganizující se. Takto jednoduchý koncept platí pro prostředí do 8 spolupracujících týmů. 8 není pevně stanoveným číslem pro výběr mezi prvním a druhým rámcem LeSS. Bod, kdy půjde o první nebo druhý rámec Less, závisí na kontextu produktu. V určitém okamžiku, kdy první rámec less již nedokáže pochopit a podchytit celý produkt, řešitelský tým nedokáže efektivně komunikovat s ostatními týmy, nedokáže vyvážit vnější a vnitřní zaměření a produkt je tak velký, že pro jednu osobu se stává obtížné vše zvládnout, když se organizace producentů již nedokáže soustředit na správu produktů na vysoké úrovni, je třeba použít druhý rámec. [33]

LeSS Huge

Je použit hlavně u velkých firem a korporací. Principiálně je to stejné jako v malé firmě. Rozdíl je jen v implementaci, která je výrazně komplexnější a změna je náročnější. Stále jde o stabilní a sebeorganizující se, které společně dodávají jeden produkt.

Less Huge není až tak odlišný od klasického LeSS, co je ale v Less Huge složitější, je struktura Product backlog. Kvůli složitější struktuře Product Backlog vzniká podpůrná struktura takzvaných Area Product Owners, kteří pomáhají výše uvedenému Product Ownerovi s určením priority. Area Product Owners nejsou fixní, ale dynamicky se mění podle potřeb. Area Product Owner obvykle nemusí znát všechno, ale věnuje se nějaké oblasti (Area). Je třeba si uvědomit, že oblast není komponenta a týmy dodávají vždy kompletní funkcionalitu. [33]



Obrázek 18 - Large-Scale Scrum (Zdroj: [33])

3.5 Metoda AHP

Autor metody AHP Thomas L. Saaty (*1926, Irák) je americký matematik působící jako univerzitní profesor na Univerzitě v Pittsburghu. Je autorem, architektem a zakladatelem teorie analytického hierarchického procesu (AHP), rozsáhlé oblasti rozhodování, multikriteriální rozhodovací analýzy, analytického síťového procesu a jeho zobecnění na rozhodnutí se závislostí a zpětnou vazbou. Publikoval velké množství článků a vydal více než 12 knih na toto téma (Pittbusiness, 2008).

Analytický hierarchický proces je strukturovaná technika určená na řešení komplexních rozhodnutí. Je založena na matematickém postupu a lidské psychologii.
AHP poskytuje komplexní a logickou koncepci pro strukturování problému, pro kvantifikaci jeho elementů, které souvisejí s celkovými cíli a pro hodnocení alternativních řešení. AHP může být využita ve více různých oblastech. Je používána na celém světě v rozmanitých rozhodovacích situacích, v oborech jako státní správa, obchod, průmysl, zdravotnictví, vzdělávání. Je vhodnou metodou pro hodnocení firem, kde několik kritérií vede k objektivizaci jejich hodnocení. Byla použita v mnoha rozhodnutích v oblasti ekonomiky, energetiky, managementu, environmentalistiky, dopravy, zemědělství, průmyslu a armády.

Z faktorů, které činí metodu AHP snad nejpopulárnější rozhodovací metodou na světě je možno zdůraznit, že se přizpůsobuje pevným údajem, jako např. cena, rychlost dodávky, jakož i osobní zkušenosti a v neposlední řadě i intuici. Dovoluje tedy

matematicky odvodit váhu jednotlivých kritérií, namísto subjektivní volby váhy kritérií, jak to používají jiné rozhodovací metody. V první fázi, před samotnou aplikací metody, musí hodnotící subjekt (firma, podnik, organizace ...) definovat všechna kritéria a subkritéria, na základě kterých bude dané hodnocení probíhat. Výběr jednotlivých kritérií a subkritérií se uskutečňuje na základě dosavadních poznatků a zkušeností každého hodnotícího subjektu. Pokud se jedná o vůbec první hodnocení jistého subjektu, tak si musí kritéria vytřídit více méně podle vlastní intuice, resp. podle vzoru nějakého jiného hodnotícího subjektu. [34] [35]

Struktura metody AHP

Metoda AHP jako flexibilní model pro rozhodování, objasňuje problémy, které mají několik možných řešení. AHP je prováděna expertní a následně matematickou metodou, která rozděluje hlavní problém do menších a detailnějších prvků.
rozhodování podle metody AHP může být rozděleno do tří rozdílných stupňů (Saat, 1985):

1. **Hierarchičnost**
2. **Priority**
3. **Konzistentnost**

Hierarchičnost

Hierarchie je systém klasifikování a organizování lidí, věcí, myšlenek, kde každý prvek systému, kromě vrcholového, je podřízen jednomu nebo více prvkům. Hierarchické diagramy mají většinou tvar pyramid, ale není to vždy nezbytné. Existuje několik typů hierarchií. Nejjednodušší jsou tzv. "Hierarchie dominance", kde podobně jako obrácený strom jsou hlavní nadřazené atributy na vrcholu. Ty jsou dále následovány za sebou jdoucími stupni s postupně nižší a nižší dominancí.

"Holarchie" jsou v podstatě hierarchie dominance se zpětnou vazbou. "Čínské krabice" (nebo standardní hierarchie) narůstají postupně od nejjednodušších prvků nebo složek (vnitřní krabice) do větších a větších agregátů (vnější krabice). Lidské organizace jsou ve většině případů strukturované jako hierarchie, kde hierarchický systém je použit pro přidělení odpovědnosti, praktikování vůdcovství a usnadňování komunikace. Běžnou hierarchií "věcí" je např. stolní počítač na "vrcholu" s jeho podřízeným monitorem,

klávesnicí a myší. Hierarchická struktura je základem způsobu uvažování člověka, rozčlenit realitu na skupiny a podskupiny. Pomocí hierarchie máme možnost seřadit velké množství informací pro pochopení konkrétního rozhodovacího problému. Vytvářením takové informační struktury formujeme lepší a lepší obraz o problému jako celku.

Vysvětlení AHP hierarchie při vytváření strukturované hierarchie při metodě AHP se sestaví systém optimalizace sestávající z hlavního cíle, zvolené skupiny faktorů nebo kritérií a alternativ, uspořádaný podobně jako rodokmen. V nutných případech jsou dále kritéria rozepsaná na subkritéria a ty následně na další subkritéria atd., až do takového počtu úrovní, kolik jich problém vyžaduje. Nejpoužívanějším způsobem zobrazení hierarchie při této metodě je diagram, s cílem ve vrcholu, alternativami v dolní části a kritérii vyplňujícími prostor uprostřed.

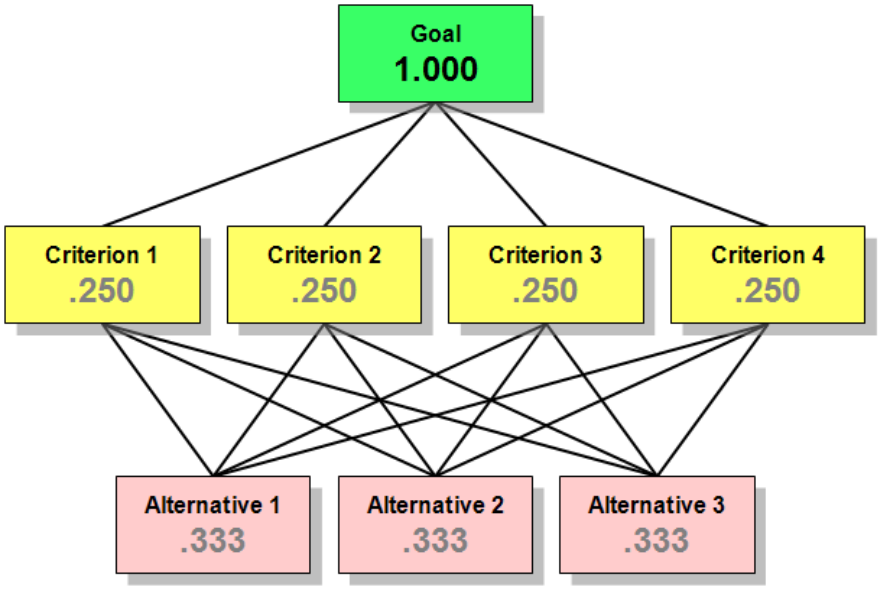
V takových diagramech se jednotlivé buňky nazývají uzly. Buňky vycházející z jakéhokoliv uzlu se nazývají děti - dceřiné uzly a uzly, z nichž vycházejí dceřiné uzly, jsou nazvány rodičovské uzly.

Proces sestavování hierarchie systému se nedělá jen proto, aby pomohl přesněji identifikovat všechny rozhodující prvky, ale také kvůli rozpoznání vazeb mezi nimi. Hlavní myšlenkou v modelu AHP je tedy rozčlenit hlavní problém na oddělené prvky (subkritéria) a ty navzájem mezi sebou porovnat.

Rozložení vlastností na menší podvlastnosti, které je vytvářejí, je velmi důležité z několika důvodů:

a) Hodnocení výsledků podle jednotlivých dílčích kritérií je podstatně jednodušší.
b) Jestliže tato hodnocení vyvolávají nějakou pochybnost je snazší si je ověřit.
c) Jednotlivé dílčí hodnotící kritéria mají přesnější smyslový obsah.
d) Při hodnocení podle jednotlivých kritérií je shoda stanovisek expertů mnohem větší než při hodnocení výsledku vcelku.

Nakolik je každý proces vytváření hierarchie jedinečný, není specifikován žádný postup, jak toto uspořádání vytvořit. Účastníci zkoumají aspekty problému od základní po nejdetailnější úroveň, což následně vyjádří víceúrovňovým způsobem, který AHP vyžaduje. Podle dosavadních přístupů je vhodnou metodou pro vytvoření hierarchie např. brainstorming. [34] [35]



Obrázek 19 - Metoda AHP (Zdroj: [34])

Priority

Po vytřídění vlastního souboru kritérií a sestavení hierarchické struktury se na všech úrovních hodnocení vzájemně porovnávají různé alternativy nebo kritéria, které mají vliv na hodnocení prostřednictvím slovního
vysvětlení a číselných hodnot. Výsledek je dán váhou v poměrné stupnici pro alternativy a kritéria. [34] [35]

Základní škála párového srovnání při metodě AHP:

Intenzita důležitosti

1 Stejná důležitost. Dva prvky se stejně podílejí na intervenci cíle.

3 Menší důležitost jednoho prvku vzhledem k druhému. Zkušenosti a názory jemně preferují jeden atribut před druhým.

5 Podstatná nebo silná důležitost. Zkušenosti a názory silně preferují jeden atribut před druhým.

7 Demonstrovatelná důležitost. Jeden atribut je velmi preferovaný a jeho dominance je demonstrována v praxi.

9 Absolutní důležitost. Evidentní favorizován jednoho atributu před druhým je na nejvyšším možném stupni vyjádření.

2, 4, 6, 8 Střední hodnoty mezi dvěma sousedními posouzeními.

Pokud je potřebný kompromis vzhledem k nejednoznačnosti přiřazení k uvedeným definicím důležitosti. Jako prostředek hodnocení jsou většinou použity dotazníky. Obvykle je dotazník číselný, také může být verbální - ovšem pro výpočty musí být verbálně výsledky převedeny na numerické. Všechny alternativy na úrovni kritérií jsou porovnány na základě párového přidělování vah. Při párovém porovnávání se dvě kritéria umístí do protilehlých konců řádku proti sobě a jsou porovnávány, které je důležitější. Uprostřed řádku je číslo 1, což znamená že porovnávané kritéria jsou stejně důležité. Podél řádku jsou čísla od 1 po 9, kde číslo 9 znamená, že kritérium na tomto příslušném konci bylo důležitější než kritérium na opačném konci. Pokud je n celkový počet prvků, které jsou porovnávány, pak platí, že počet porovnávání je: n · () n - 1/2.

Údaje o významnosti kritérií, získané na základě jejich párových porovnávání jsou hodnoty rij, udávající poměr významnosti hodnotícího kritéria k1 ke kritériu k2, kde i, j = 1,2, ..., m. Požaduje se, aby veličiny rij splňovaly pro všechny i, j = 1,2, ..., m, kde m je počet hodnotících kritérií, následující podmínky: rij> 0, -1 ij = ji rr, = 1 ii r. (2) Veličiny rij, relativní významnosti kritérií, se uspořádají do čtvercové matice relativních významností R. Základem Saatyho metody je výpočet maximálního charakteristického čísla λ matice relativních významnosti R (charakteristické čísla jsou řešením rovnice det (λ E - R) = 0), přičemž se předpokládá, že rozhodování o významnosti hodnocených kritérií je konzistentní nebo blízké konzistentnímu.

V případě větší nekonzistence Saaty doporučuje, aby expert své ocenění kritérií přehodnotil a matici relativních významností R upravil tak, aby zvýšil její konzistenci. [34] [35]

Přidělování vah

Správné a odpovědné určení vah jednotlivých dílčích hodnotících kritérií je jedna ze základních úkolů při řešení multikriteriálních úloh. Proto je třeba dobře znát řešenou problematiku a znát význam a dopad kritérií, kterými hodnotíme dosažený výsledek. Existuje řada metod, které pomáhají zpřesnit váhy kritérií určené na začátku expertům, resp. skupinou expertů z dané oblasti řešené problematiky.

Metody, umožňující realizovat kvantitativní (ale i kvalitativní) uspořádání na množině hodnotících kritérií, se nazývají expertními metodami. Expertní metody se používají nejen při určení vah kritérií, ale i při určování vah cílů a při určování vah

rozhodovacích kritérií, atd. Tyto expertní metody se rozlišují podle toho, zda jsou ocenění přidělovány na základě výroků jednoho experta nebo skupiny odborníků.

V praxi převládají skupinové ocenění, která jsou však doprovázeny dodatečnými problémy, jako je určení velikosti skupiny expertů, její složení, vytvoření podmínek zajišťujících objektivizované hodnocení expertů, určení postupu prací atd. Jsou to metody jako brainwriting, brainstorming, delfská metoda. Je zřejmé, že tak jak je zatíženo subjektivním činitelem samotné určení vah kritérií, tak je jím zatížen i výběr expertů. Někdy se za hlavní kritérium kompetentnosti expertů považuje jejich profesionální znalost, jindy se zase upřednostňuje originalita a intuice. Koeficient kompetence zohledňuje přirozeně i dosažený stupeň vzdělání, získanou praxi a zdroje, na základě kterých expert provádí hodnocení. Za základní vlastnost expertů lze označit jejich objektivitu. Tu lze dosáhnout soustředěním několika specialistů z oblastí, které mají profesionální vztah k řešenému problému. Je přirozené, že větší nároky jsou kladeny na výběr experta při individuálních expertízách.

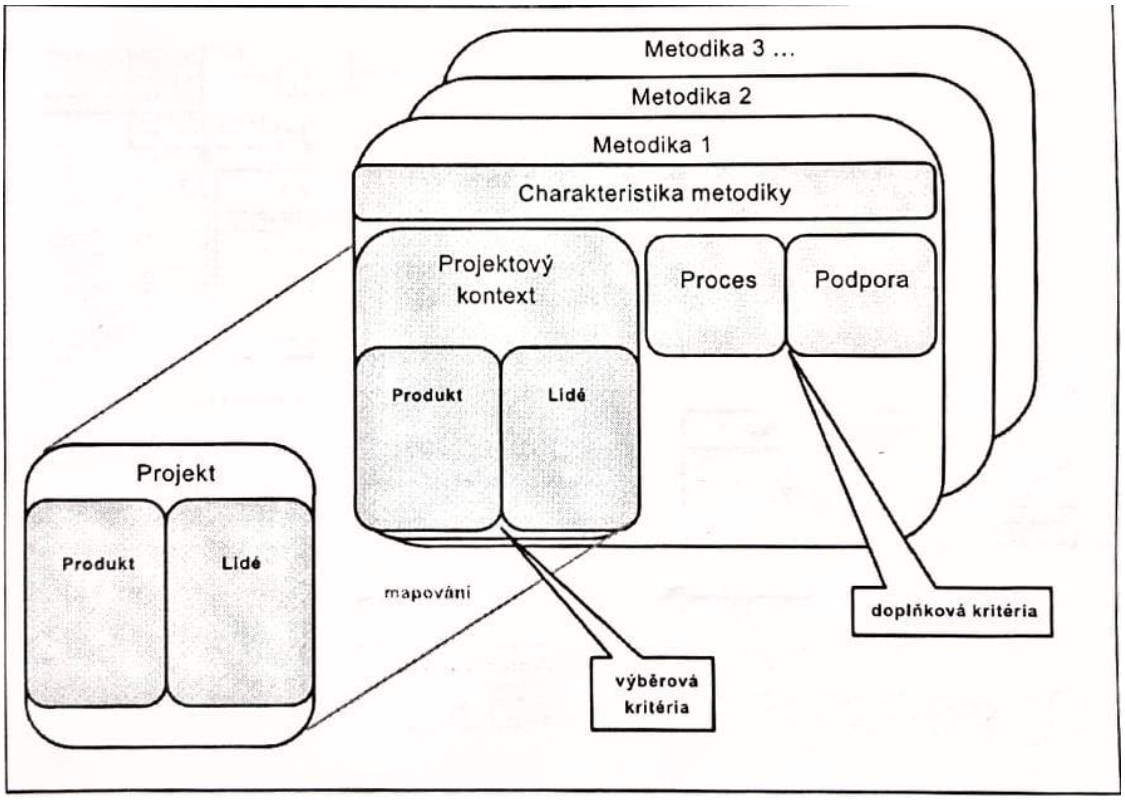
Konzistence

Alternativy rozhodování jsou setříděny v pořadí podle hodnocení.

Při používání metody AHP pro rozhodování musí být splněny čtyři axiomy:

1. Inverzní axiom: pokud alternativa A je n- krát preferovaná vůči B, pak alternativa B je 1 / n- krát preferovaná vůči A. Jde o pravidlo reciprocity, vyjádřené vztahem: ji ij rr 1 =.
2. Homogenní axiom: porovnávání párováním je významné, pouze pokud prvky jsou srovnatelné.
3. Závislý axiom: srovnávání na nižší úrovni (podkritérium) závisí na prvku na vyšší úrovni (na výšším kriréria). Platí tedy pravidlo tranzitivity, které se dá popsat vztahem: ij ik kj r = r ·r, kde i, kde jsou vybrané alternativy z matice R.
4. důsledkovými axiom: pokud nějaké kritérium v hierarchii bude změněno, je třeba očekávat nové ocenění pro novou hierarchii (Saat, Joyce, 1981). Splněním všech podmínek získáme kompletní matici párového srovnání. [34] [35]

3.6 Systém hodnocení a výběru metodik METES



Obrázek 20 - Struktura systému METES (Zdroj: [14])

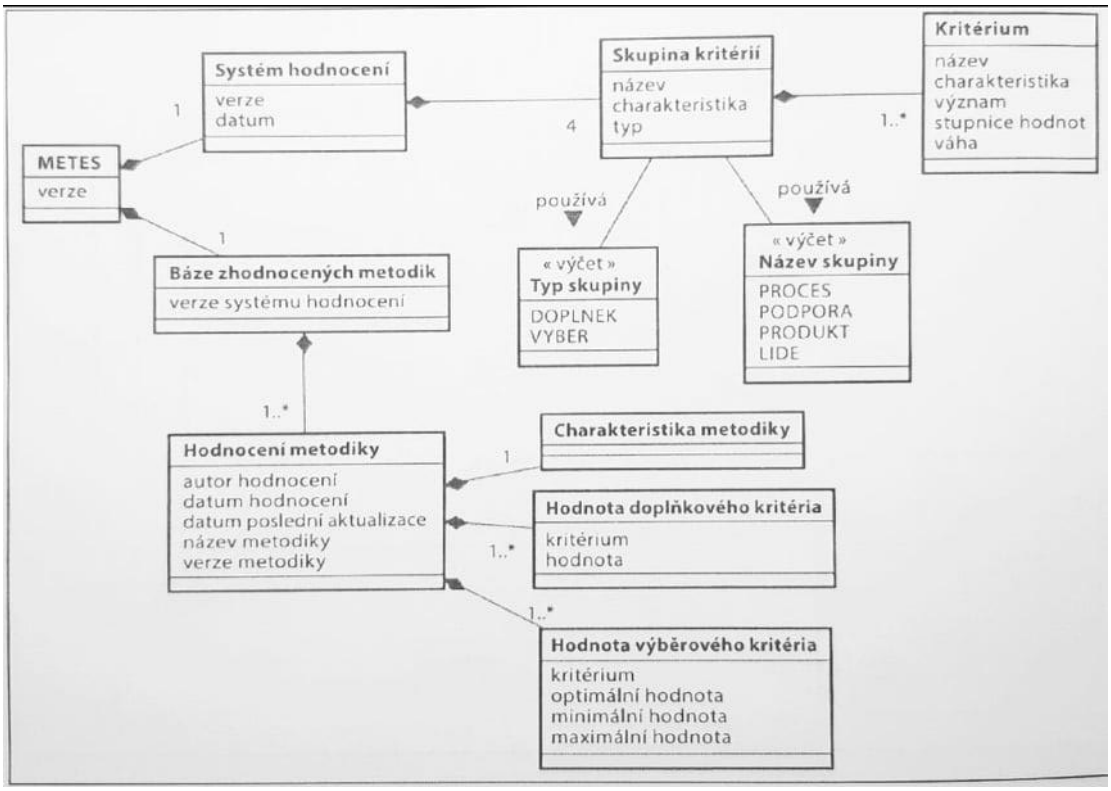
Každá z vybraných metodik je krátce hodnocena. Metodika se hodnotí podle 4 kritérií, z toho jsou 2 hlavní výběrové skupiny kritérií: Produkt a Lidé. Zbylé 2 jsou vedlejší doplňkové skupiny kritérií: Proces a Podpora. [14]

Kritéria skupiny Proces se zaměřují hlavně na procesní charakteristiky: způsob vývoje, role, procesy životního cyklu, metriky, model životního cyklu [14]

Kritéria skupiny Podpora hodnotí: dostupnost metodiky, podporu jejího zavedení a přizpůsobení, dostupnost kvalifikovaných lidí. [14]

Zbylé dvě hlavní výběrové skupiny kritérií Produkt a Lidé představují projektový kontext a jsou určující pro výběr metodiky. Skupina Produkt obsahuje kritéria spojení s produktem a skupina kritérií Lidé obsahuje charakteristiky daného projektového týmu. [14]

Každé kritérium ve výše uvedených skupinách se hodnotí na stupnici od 0 do 5, kde 0 až 1 představuje nízký stupeň splnění, hodnocení 2 až 3 představuje střední stupeň splnění a hodnoty 4 a 5 představují vysoký stupeň splnění. Jednotlivá kritéria jsou blíže specifikována v příloze A. [14]



Obrázek 21 - Konceptuální model systému METES (Zdroj: [14])

Systém METES obsahuje jednak popis systému hodnocení, alle i bázi zhodnocených metodik, ze které se vybírá vhodná metodika. U každé metodiky jsou vyhodnoceny její minimální, maximální a optimální hodnoty, které jsou pro danou metodiku přijatelné.

Jako první se definují váhy jednotlivých kritérií a to nejlépe Saatyho metodou, která je popsána v rámci metody AHP. Tyto váhy jsou pro každý projekt unikátní a měli by odrážet potřeby daného projektu.

Dále se určí hodnoty jednotlivých kritérií pro daný projekt. Ty se pak porovnají prvně s maximálními a minimálními hodnotami pro dané kritérium. Následně se vyberou nejvýznamnější kritéria podle předem určených vah jednotlivých kritérií a ty se pak porovnají s danými hodnotami. Pokud je některá z nich mimo stanovený interval, tak to znamená, že daná metodika není pro projektu použitelná.

Dále se výsledné hodnoty kritérií pro projekt porovnají s optimálními hodnotami pro dané metodiky. Určí se vzdálenost od optima, která se následně vynásobí váhou daného kritéria.

Jako poslední se vyberou ty metodiky, které součet těchto hodnot mají nejnižší. Pokud je pouze jedna metodiky, která splňuje všechny požadavky uvedené výše, tak je vybrána právě ona. Pokud je více než jedna vhodná metodika, vybírá se z nich následně ta nejvhodnější pomocí doplňkových kritérií. Jedná se o podobnou operaci jako u hlavních kritérií, jen s tím rozdílem, že se nehodnotí vzdálenost od předem stanovených hodnot, ale hodnotí se každá metodika zvlášť podle daných kritérií. Poté se jednotlivé hodnoty opět vynásobí váhou kritéria a vybere se ta metodika, co má největší součet těchto vážených hodnot. [14]

Tabulka 3 - Tabulka optimálních hodnot metodik podle systému METES (Zdroj: Autor)

	SCRUM	FDD	XP	Kanban	LeSS	SAFe
Důležitost projektu	3	3	3	2	4	4
Délka projektu	3	3	2	3	5	3
Stálost požadavků	0	1	0	0	1	1
Znovupoužitelnost	1	1	1	0	0	1
Velikost řešení	5	4	3	3	5	5
Zkušenost manažera projektu	1	3	2	3	1	1
Kvalifikace členů týmu	0	2	1	1	1	1
Motivace členů týmu	0	2	1	1	1	1
Dostupnost uživatelů	0	1	0	2	1	2
Velikost týmu	1	3	1	1	5	5
Rozmístění	3	1	1	1	5	5

4 Vlastní práce

Vlastní práce se zabývá hlavně samotným výběrem vhodné metodiky pomocí systémů hodnocení a výběru metodik METES a její implementace na projektu v konkrétním týmu. V následující části je také vybrán nástroj pro podporu vybrané agilní metodiky, pomocí vícekriteriální analýzy variant, a to přímo metodou analytického hierarchického procesu (AHP)

4.1 Popis projektu

Jedná se o větší projekt v korporátní firmě působící na trhu se správou financí. Tým, pro který se bude metodika vybírat se skládá z 15 lidí. Z toho jsou 2 vedoucí pracovníci, jedním z nich je i autor diplomové práce, který dostal za úkol vhodnou metodiku v týmu zavést, 12 odborníků na testování a jeden zástupce obchodního oddělení. Z technických a politických důvodů není možné týmy spojit v jeden, jak by to bylo v případě agilní metodiky nejlepší. Jedná se tedy pouze o testovací tým dané korporátní firmy. Všechny ostatní projektové role tvoří vlastní izolovaný tým, který se řídí svojí vlastní metodikou. Nicméně všechny týmy nově používají agilní metodiky, takže je poměrně snadné na ně navázat s plánováním.

V minulosti všechny týmy používaly k organizaci metodiku Vodopád. Z rozhodnutí vedoucích pracovníků firmy se ale všechna organizace v IT oddělení musí změnit v organizaci pomocí agilních metodik. Hlavními důvody jsou právě větší transparentnost práce, rychlé přizpůsobení k měnícímu se trhu a co možná největší a nejrychlejší přínos zákazníkům.

Cílem projektu je vývoj webové aplikace pro správu finančních prostředků určené zákazníkům dané firmy.

Dohromady se jedná o 4 oddělené týmy. První z týmů se zabývá analýzou webové aplikace, a to od business analýzy až po technickou specifikaci. Druhý tým se zabývá vývojem backendového systému pro danou webovou aplikaci. Třetí tým se zabývá vývojem samotného frontendu aplikace a pak je zde čtvrtý testovací tým, který je předmětem této práce.

4.2 Stávající problémy

Před snahou vedení společnosti zavést agilní metodiky pro vývoj výše zmiňované webové aplikace, byla největším problémem poměrně dlouhá doba zpracování nových požadavků a tím pádem pomalé dodání přidané hodnoty zákazníkovi. Jelikož je na trhu, kde vystupuje daná korporace velmi vysoká konkurence, je zapotřebí dobu, pro dodání přínosu zákazníkovi, zkrátit. V minulosti totiž trvalo dodání nové funkčnosti až šest měsíců, což je v oboru dané korporátní firmy v současnosti nepřijatelné.

Dalším problémem, který vedl ke změně metodiky, je neefektivní využití zdrojů a následné nedodržení termínů. Z pravidla se totiž stávalo, že se hned ze začátku vodopádového cyklu, některá část řetězce zpozdila. To následně vedlo k tomu, že týmy, které navazovaly na danou část, neměly přiřazenou žádnou práci a musely pouze čekat na dokončení předchozí fáze. Jelikož pak byla snaha o to, vše stihnout v termínu, tak byl tým zbytečně přealokován pracovníky a ani tak se většina požadavků nestačila dodat v termínech.

Při vyhodnocování takovýchto zpoždění se málokdy našel konkrétní důvod vzniku a tedy nebylo ani možné efektivně problém odstranit.

4.3 Volba agilní metodiky

Jako první je potřebujeme definovat jednotlivé metodiky, ze kterých budeme vybírat. Vybíráno bude pouze z agilních metodik, jelikož to bylo hlavním požadavkem vedení firmy. Dále již bylo na autorovi práce, aby zvolil přijatelné metodiky a zohlednil všechny požadavky vedení. Agilní metodiky byly vybrány na základě knihy Zlepšování procesů při budování informačních systémů (dát odkaz na knížku dolu), kde jsou popsány nejaktuálnější a doporučené agilní metodiky. Bude tedy vybíráno z těchto metodik: SCRUM, Kanban, Extrémní programování (XP), Vývoj založený na funkcionalitě (FDD), LeSS, SAFe.

Pro volbu z výše zmíněných metodik bude použit „Systém výběru a hodnocení metodik METES“, který autor práce zvolil kvůli jeho dlouhodobému používání a stálému vylepšování.

Systém výběru a hodnocení metodik hodnotí metodiky na základě čtyř skupin kritérií. Z toho jsou dvě hlavní skupiny kritérií, a to kritéria skupiny Produkt a kritéria

skupiny Lidé. Zbylé dvě vedlejší skupiny kritérií, kritéria skupiny Proces a kritéria skupiny Podpora, se používají pouze v případě, že z výsledků hlavních skupin kritérií je přijatelná více než jedna metodika. Stupnice hodnocení pro jednotlivá kritéria jsou uvedené v příloze A.

Systém výběru a hodnocení metodik METES dále doporučuje váhy pro jednotlivá kritéria. Ty byly mírně upravené, aby přesně vystihovaly potřeby firmy a týmu. V tabulce níže jsou vyznačena hlavní kritéria pro volbu metodiky. Pro tento tým jsou to tedy: Důležitost projektu, Délka projektu, Dostupnost uživatelů, Velikost týmu a Rozmístění

Tabulka 4 - Váhy výběrových kritérií systému METES (Zdroj: Autor)

Název kritéria	Váhy
Důležitost projektu	0,2925
Délka projektu	0,1498
Stálost požadavků	0,0469
Znovupoužitelnost	0,0352
Velikost řešení	0,0411
Zkušenost manažera projektu	0,0244
Kvalifikace členů týmu	0,0204
Motivace členů týmu	0,0200
Dostupnost uživatelů	0,1153
Velikost týmu	0,1442
Rozmístění	0,1102

Dále byly stanoveny jednotlivé hodnoty pro skupiny kritérií Produkt a Lidé, které jsou uvedeny níže.

Tabulka 5 - Hodnocení jednotlivých kritérií podle systému METES (Zdroj: Autor)

Název kritéria	Hodnocení (0-5)
Důležitost projektu	2
Délka projektu	5
Stálost požadavků	1
Znovupoužitelnost	1
Velikost řešení	3
Zkušenost manažera projektu	3
Kvalifikace členů týmu	0
Motivace členů týmu	2
Dostupnost uživatelů	2
Velikost týmu	2
Rozmístění	0

Na základě výše stanovených hodnot byl vypočítán vážený součet vzdáleností od optimálního řešení pro každou z vybraných metodik.

Tabulka 6 - Vypočítané vážené hodnoty pomocí systému METES (Zdroj: Autor)

Metodika	Vážené hodnoty
Kanban	0,6765
FDD	1,0437
XP	1,3385
SCRUM	1,5152
LeSS	1,8903
SAFe	2,0393

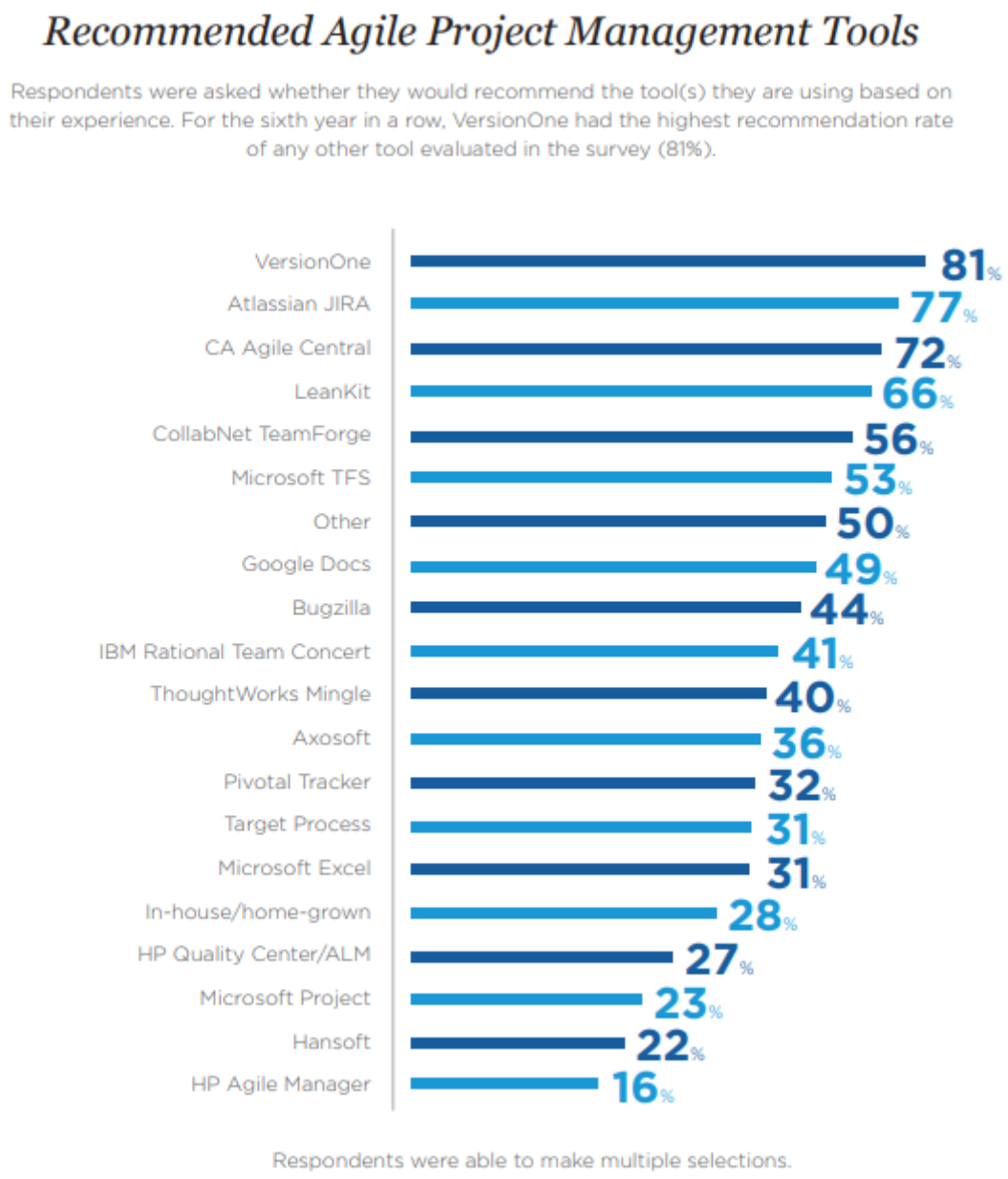
Na základě těchto hodnot byla nakonec vybrána metodika Kanban, jako jediná přípustná pro zmiňovaný tým. Ostatní metodiky totiž nesplnili rozsah hlavních výběrových kritérií, jak je vidět v tabulce níže. Kompletní srovnání kritérií s jejich optimem, maximální a minimální hodnotou je uvedeno v příloze B.

Tabulka 7 - Minimální, maximální a optimální hodnoty srovnané s hodnocením projektu dle systému METES (Zdroj: Autor)

FDD	minimum	maximum	optimum	Projekt
Délka projektu	0	4	3	5
Dostupnost uživatelů	0	1	1	2
SCRUM	minimum	maximum	optimum	Projekt
Dostupnost uživatelů	0	1	0	2
XP	minimum	maximum	optimum	Projekt
Délka projektu	0	4	2	5
Dostupnost uživatelů	0	1	0	2
Velikost týmu	0	1	1	2
LeSS	minimum	maximum	optimum	Projekt
Dostupnost uživatelů	1	1	1	2
Velikost týmu	5	5	5	2
Rozmístění	1	5	5	0
SAFe	minimum	maximum	optimum	Projekt
Velikost týmu	5	5	5	2
Rozmístění	1	5	5	0

4.4 Volba nástroje

Volba podpůrných nástrojů byla inspirována The State of Agile Report 2017, kde je uvedeno kromě jiného, i které nástroje pro podporu agilních metodik se používají, a které jsou odborníky doporučené.



Obrázek 22 - Doporučené nástroje pro podporu agilní metodiky (Zdroj: [2])

Z těchto doporučených nástrojů byly vybrány první čtyři a to: Atlassian JIRA, VersionOne, CA Agile Central a LeanKit. Dále byly přidány dva nástroje na doporučení od kolegů z firmy a to: Trello a Youtrack.

Z těchto nástrojů byl vybrán pomocí vícekriteriální analýzy variant, a to metodou Analytického hierarchického procesu, ten nástroj, který splňoval všechna stanovená kritéria nejlépe.

Kritéria byla stanovená autorem a druhým vedoucím pracovníkem v týmu. Jelikož už se jedná pouze o podpůrný nástroj týmu, vedení firmy kritéria obecně neurčovalo, ale byl zohledněn jejich prospěch například kritériem Cena.

Váhy jednotlivých kritérií byly určeny podobně jako u systému výběru a hodnocení metodik METES, tedy Saatyho metodou. Detailní zobrazení výpočetních tabulek se nachází v příloze C.

Tabulka 8 - Váhy kritérií pro volbu nástroje (Zdroj: Autor)

Název kritéria	Váhy
Cena	0,3374
Dostupnost na telefonu	0,0280
Nastavení oprávnění	0,0852
Automatické Měření času práce	0,1596
Podpora	0,1717
Retrospektiva	0,0465
Podpora testování	0,1717

4.4.1 Zvolená kritéria pro hodnocení nástroje

Kritérium „Cena“

Ohodnocení tohoto kritéria bylo přímo ze zdrojů jednotlivých nástrojů. Celková cena je přepočítána pro tým patnácti lidí. Cílem je, aby cena byla co nejnižší.

Kritérium „Dostupnost pro telefon“

Toto kritérium hodnotí možnost používání daného nástroje na telefonu. Bere v úvahu aplikace, které jsou přímo od dodavatele nástroje podporovány. Pokud nemá aplikaci přímo pro telefon, pak se bere do úvahy zdali aplikace podporuje responzivní design pro obrazovku telefonu. Cílem je, aby nástroj měl zařízenou podporu mobilní aplikací.

Kritérium „Nastavení oprávnění“

Inspirováno diplomovou prací Lucie Hefnerové [10]. Cílem je získat co nejlepší hodnocení, nejlepším je tedy 1 a nejhorším je 5.

Tabulka 9 - Popis kritéria - Nastavení oprávnění (Zdroj: Autor)

Hodnocení	Popis
1	Komplexní podpora řízení uživatelských oprávnění s možností vytváření skupin a nastavení práv ke všem dostupným entitám v nástroji.
2	Rozšířená podpora řízení uživatelských oprávnění s možností vytváření skupin - omezené nastavení práv k entitám.
3	Globální řízení uživatelských oprávnění.
4	Předdefinovaná uživatelská oprávnění - bez možnosti změny.
5	Žádná podpora řízení uživatelských oprávnění.

Kritérium „Automatické měření času práce“

Kritérium, které určuje, zdali je možné, aby nástroj automaticky měřil čas práce na jednotlivých úkolech. Hodnotí se i případné možnosti přidání dalšího externího nástroje, aby toto měření bylo umožněno. Cílem je, aby nástroj podporoval automatické měření času.

Kritérium „Podpora“

Opět inspirováno diplomovou prací Lucie Hefnerové [10]. Cílem je získat co nejlepší hodnocení, nejlepším je tedy 1 a nejhorším je 5.

Tabulka 10 - Popis kritéria - Podpora (Zdroj: Autor)

Hodnocení	Popis
1	Podpora dostupná 24/7 - více komunikačních kanálů.
2	Podpora dostupná 24/7 - jeden komunikační kanál.
3	Podpora dostupná pouze v pracovní dobu - více komunikačních kanálů.
4	Podpora dostupná pouze v pracovní dobu - jeden komunikační kanál.
5	Bez garantované podpory.

Kritérium „Retrospektiva“

Toto kritérium zohledňuje, zdali je v nástroji oficiálně možné nějakým způsobem evidovat výstupy z Retrospektivy. Cílem je, aby nástroj spravoval výstupy z retrospektivy.

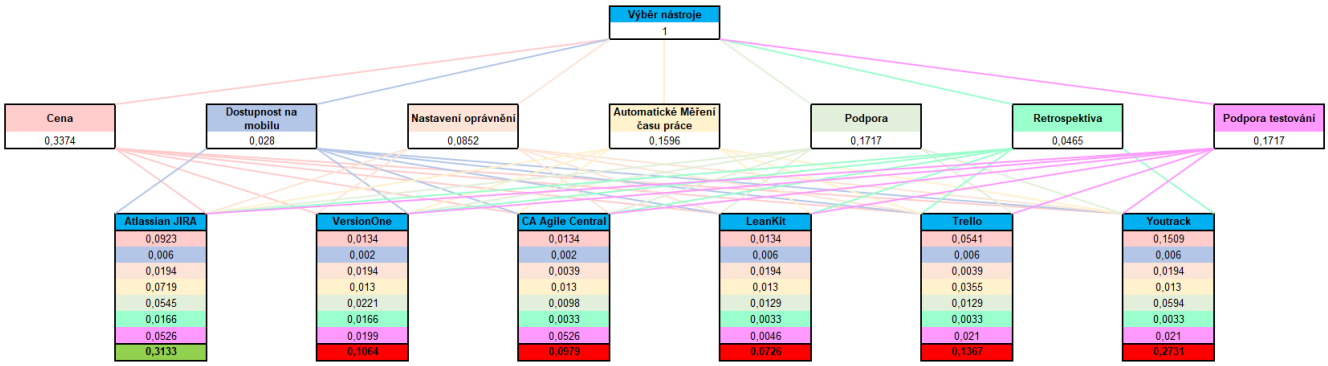
Kritérium „Podpora testování“

Kritérium, které určuje, zdali daný nástroj podporuje testování a to formou, že je možné v něm, jak evidovat samotné testovací scénáře. Opět jsou zohledněny rozšíření, ať už vlastní, nebo od externích dodavatelů.

Tabulka 11 - Hodnocení jednotlivých kritérií pro voilbu nástroje (Zdroj: Autor)

	Atlassian JIRA	**VersionOne**	**CA Agile Central**	**LeanKit**	**Trello**	**Youtrack**
Cena	$75 za měsíc	$435 za měsíc	$400 za měsíc	$435 za měsíc	$150 za měsíc	$60 za měsíc
Dostupnost pro telefon	Ano	Ne, pouze responzivní design	Ne, pouze responzivní design	Ano	Ano	Ano
Nastavení oprávnění	1	1	4	1	4	1
Automatické Měření času práce	Ano	Ne, pouze ruční	Ne, pouze ruční	Ne, pouze ruční	Ne, pouze externí nástroj	Ne, pouze ruční
Podpora	1	2	2	4	4	1
Retrospektiva	ano	ano	ne	ne	ne	ne
Podpora testování	Ano, formou rozšíření	Ano, formou rozšíření od jiných společností	ano	ne	Ano, formou rozšíření od jiných společností	Ano, formou rozšíření od jiných společností

Na základě výše stanovených hodnot byly vypočítané preferenční informace jednotlivých kritérií a následně sečteny pro získání nejvhodnější kompromisní varianty. Všechny informace o výpočtu jednotlivých kroků jsou uvedeny v příloze C.



Obrázek 23 - Metoda AHP (Zdroj: Autor)

Tabulka 12 - Součet preferenčních informací pro volbu nástroje (Zdroj: Autor)

Nástroj	Součet preferenčních informací
Atlassian JIRA	0,3133
Youtrack	0,2731
Trello	0,1367
VersionOne	0,1064
CA Agile Central	0,0979
LeanKit	0,0726

Jako nejvhodnější nástroj byl tedy vybrán Atlassian JIRA.

4.5 Popis reálného zavedení metodiky a nástrojů

Pro zavedení metodiky Kanban není obecně mnoho pravidel, jelikož se jedná o jednu z lehčích metodik. Je ale potřeba dát pozor, aby zavedení metodiky mělo nějaký reálný přínos a nejednalo se pouze o administrativní práci, která nebude dále nijak využívána. Některá odborná literatura nebere samotný Kanban jako metodiku, pohlíží na něj pouze jako pro možnost zobrazení toku práce. Kanban obecně definuje jen základní principy, větší část odborné literatury ho ale popisuje více dopodrobna, aby se jako metodika použít dal.

Definování rolí

Jednou z výhod metodiky Kanban je, že ze začátku nemění celé fungování týmu naráz, ale jedná se o postupné věci, které se ukáží jako smysluplné. Role v týmu tedy zůstanou stále víceméně stejné, pouze se usmění jejich zaměření.

V týmu zůstane na stejné pozici a se stále stejnou zodpovědností zástupce obchodního oddělení.

Dále se tým bude skládat stále ze dvou vedoucích pracovníků, kteří si ale nově rozdělí kompetence v týmu a to tak, že jeden z nich bude zodpovědný za samotnou přípravu a exekuci testovacích případů (tím je autor práce). Druhý bude zodpovědný za tvorbu automatizovaných testů, které jsou nutnou součástí testování.

Samotný testovací tým, bude rozdělený do dvou hlavních oblastí, tím je frontendové testování a backendové testování, dále jako frontendový tým a backendový tým. Ve frontendovém týmu se bude nacházet šest testovacích odborníků a oba dva vedoucí pracovníci, kteří také budou zapojeni do samotných testovacích aktivit, ale samozřejmě v omezeném rozsahu. V backendovém týmu bude zbývajících šest pracovníků. Automatizaci testovacích případů bude provádět každý jeden testovací odborník. Frontendový tým bude tedy automatizovat ty testovací scénáře, které slouží k testování frontendu a backendový tým, za se ty backendové testovací scénáře. Některých úkolů se ale budou účastnit oba týmy najednou. Samotné úkoly jsou rozebrány dále v práci. Samotný zástupce obchodního oddělení, není přiřazen do testovacích týmů. Jeho hlavní náplní práce je zastupování zákazníka a slouží tedy týmu spíše jako konzultant, který pomáhá prioritizovat jednotlivé úkoly týmu.

Definice Work in progress (WIP)

Definice WIP je při zavedení metodiky Kanban pravděpodobně nejtěžší určit hned. Ze začátku je třeba udělat pouze odhad, který bude upraven podle potřeb týmu. Je naprosto důležité, aby byla hodnota WIP co nejpřesnější. Cílem metodiky Kanban je minimalizovat čas průchodu a toho se dosáhne jen tehdy, když bude hodnota WIP určena co nejpřesněji. Nízká hodnota WIP může zapříčinit, že bude pracovat zbytečně moc členů týmu na jednom úkolu a budou tím pádem ztrácet čas. Příliš velká hodnota může zapříčinit nejasnost v úkolech a přecházení od jedné činnosti ke druhé, čím se zvýší doba průchodu jednotlivých úkolů. Ze začátku byla určena hodnota WIP na 12 úkolů.

Vizualizace toku práce

Vizualizace je základním kamenem metodiky Kanban, proto je velice důležité, aby bylo vše co možná nejvíce transparentní a veškeré definované stavy dávali smysl. Ze začátku byly nastaveny pouze tři základní stavy a to:

Stav „To Do“

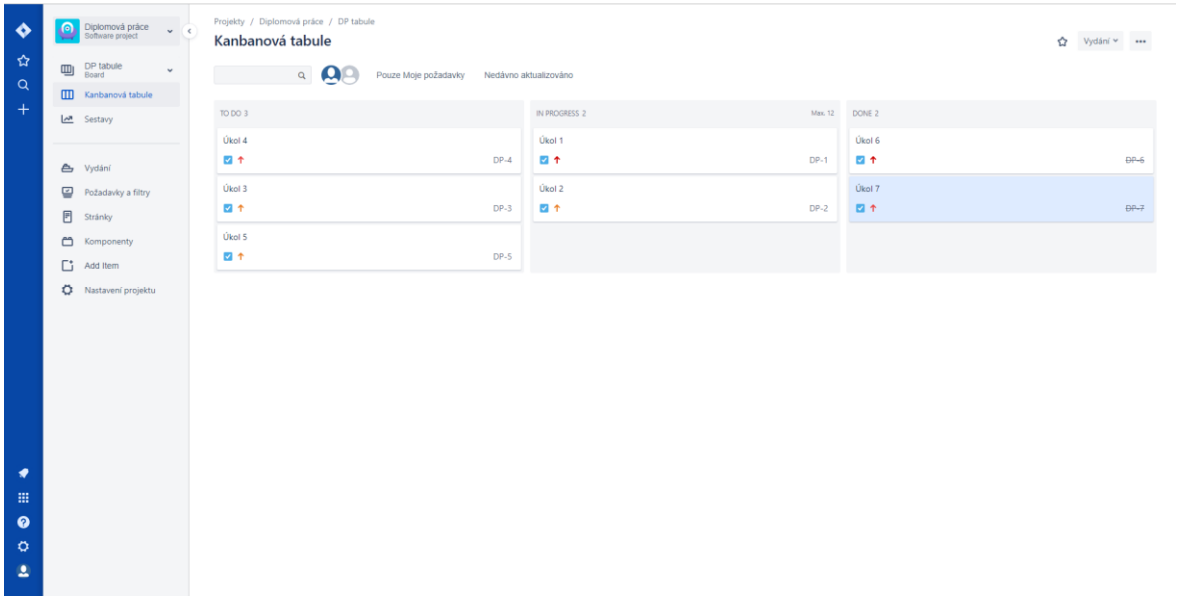
V tomto stavu jsou všechny úkoly, co je třeba udělat. Jsou seřazeny od těch nejdůležitějších po méně důležité, aby bylo jasné, na čem má tým pracovat.

Stav „In Progress“

V tomto stavu jsou právě ty úkoly, na kterých momentálně lidé z týmu pracují

Stav „Done“

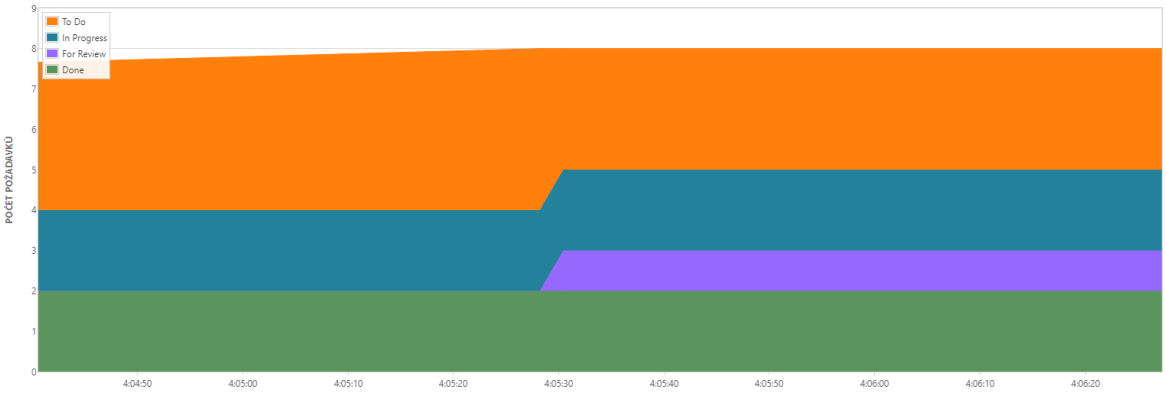
V tomto stavu se nachází úkoly, které jsou hotové. Je velmi důležité, aby každý z úkolů měl jasně stanovené tzn. Definici hotového, tedy co je potřeba udělat, aby se mohl úkol předat do stavu „Done“.



Obrázek 24 - Kanban tabule - reálná (Zdroj: Autor)

Vizualizace výstupů

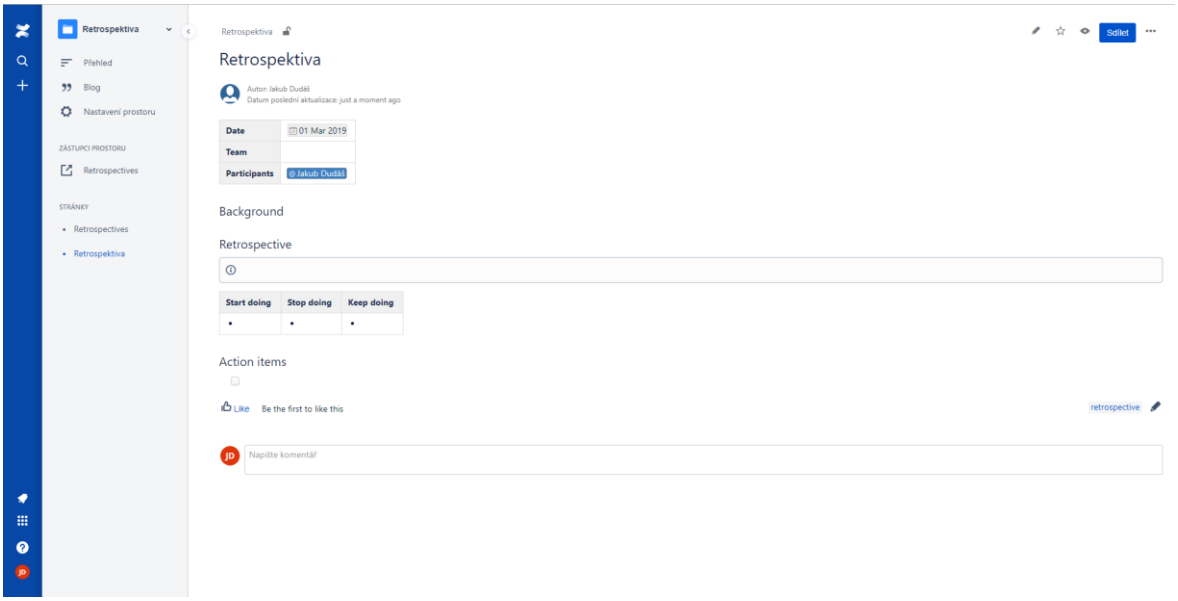
Vizualizací výstupu je myšlený hlavně graf Kumulativní diagram toků. Slouží hlavně pro zpětnou vazbu nastavení všech procesů a pro měření doby průchodu jednotlivých úkolů.



Obrázek 25 - Kumulativní diagram toků (Zdroj: Autor)

Retrospektiva

Retrospektiva slouží pro získání zpětné vazby od týmu a k řešením dalších různých problémů. Je zde prostor pro debatu o zavedeném procesu, návrhy pro zlepšení a tak dále. Ze začátku byla zavedena retrospektiva jednou týdně, aby bylo možné co nejrychleji odstranit všechny nedostatky.



Obrázek 26 - Retrospektiva - reálná (Zdroj: Autor)

Určování úkolů týmu

Jak je zmíněno výše, daný tým navazuje na práci ostatních týmů a je třeba mezi nimi zvýšit úroveň komunikace. Některé z týmů používají také Kanban, a to analytický tým a další používají Scrum, jedná se o vývojové týmy.
Samotnou prioritu úkolů budou určovat oba dva vedoucí pracovníci a zástupce obchodního oddělení na základě komunikace s vedoucím pracovníkem analytického týmu a scrum mastery vývojových týmů.

Jednotlivé typy úkolů týmů byly definovány na skupiny:

Příprava testovacích scénářů pro frontend

Úkoly týkající se přípravy testovacích případů budou mít zpravidla menší prioritu než samotná exekuce testovacích scénářů. Tyto úkoly vznikají v závislosti na výstupu od týmu analytiků. V případě že je dokončená samotná specifikace frontendových úkolů, je možné začít psát testovací případy. Tyto úkoly se dají odložit až do té doby, než si je vývojové týmy, zařadí do Sprintu. Čím více se blíží konec Sprintu, tím důležitější jsou tyto úkoly.

Exekuce testovacích případů pro frontend

Exekuce testovacích případů bude mít v ideálním stavu nejvyšší prioritu a to hlavně v případě, že budou dokončené vývojové Sprinty. Vstupem pro tyto úkoly bude tedy výstup ze sprintů vývojových týmů.

Příprava testovacích scénářů pro backend

Tento typ úkolů je totožný s přípravou testovacích dat pro frontend.

Exekuce testovacích případů pro backend

Tento typ úkolů je totožný s Exekuce testovacích případů pro frontend.

Exekuce regresních testů

Exekuce regresních testů je potřeba vždy před nasazením aplikace do produkce. Z pravidla je potřeba nasadit v co možná nejkratším čase, po obdržení vývojové verze. Proto tento typ úkolů má velmi vysokou prioritu, hned po exekuci testovacích případů nových funkčností.

Vývoj automatizovaných testů

Vývoj automatizovaných testů je na seznamu zpravidla méně důležitý, než ostatní typy úkolů, i když je obecně jeden z nejdůležitějších pro správné fungování týmu. Očekávaným cílem v tuto chvíli je pomocí optimalizace rozdělení práce, najít více času pro tento typ úkolů.

Ostatní

Mezi tyto úkoly se řadí všechny ostatní aktivity týmu. Jejich priorita je posuzována individuálně.

4.6 Reálné zavedení a provedení změn a vylepšení

Plán pro zavedení změn je získávat zpětnou vazbu od členů týmu hlavně v rámci týdenní retrospektivy a také sledovat výstupy ze sledovaných ukazatelů. Ze začátku byla doba nastavena na každých čtrnáct dní, kdy bude vyhodnocen postup a sjednocena všechna obdržená zpětná vazba. Je důležité, ze začátku dát trochu více prostoru členům týmu, aby si zvykli na nový systém práce. Následně bude vymyšleno zlepšení.

Před úplným začátkem je třeba sjednotit všechny rozdělané úkoly. Ty, co jsou prioritní tak hned ze začátku rozdělit do týmu a vložit do sloupce „In Progress“. Je však velmi důležité neporušit stanovený WIP.
Všech úkolů bylo dohromady 55. Z toho bylo 8 vybraných jako kritických, které se musejí za každou cenu dostat ihned do stavu „in progress“, aby se práce na nich nezastavila. U zbylých 47 byla definována priorita a následně byly všechny úkoly zařazeny do „To Do“.

První iterace

Během prvních čtrnácti dní, z toho deseti pracovních dní, bylo tedy zařazeno 8 kritických úkolů, které byly doplněny čtyřmi nejvíce prioritními úkoly, aby byla naplněna hodnota WIP.

Jako první se objevil problém, že většina členů týmu si nebyla jistá definicí hotového. To se projevovalo tak, že na sobě měli jednotlivé úkoly moc dlouho a často se chodili ptát vedoucích pracovníků, zda mohou úkol považovat za splněný.

Dalším problémem u některých členů byla jejich pasivita v samostatnému odebírání nových úkolů z „To Do“, ať už kvůli jejich nízkému pracovnímu nasazení, nebo hlavně kvůli nejistotě na čem mohou dále pracovat.

Jako řešení těchto problémů bylo navrženo pravidelné setkání. Podobně jako tomu bývá u metodiky Scrum, tak i v tomto týmu byly zavedeny krátké schůzky, kde byl shrnutý stav jednotlivých úkolů. Tyto schůzky se konaly dvakrát týdně, a to v úterý a ve čtvrtek během dopoledne.

Dalším problémem bylo, jak evidovat drobné náhodné úkoly, které se pravidelně objevovaly. Jednalo se například o rychlé přetestování defektu, výpomoc vývojovým týmům s hledáním dat a tak dále.

Tento problém byl vyřešen navrhnutím dvou různých přístupů, jak s těmito náhodnými úkoly pracovat.

První z nich byl, že pokud se bude jednat pouze o jeden úkol, který bude trvat méně než půl hodiny, tak se evidovat nebude. Toto rozhodnutí bylo provedeno na základě toho, že pokud by se musel psát každý takový úkol, zabralo by evidování více času, než by bylo třeba

Druhým řešením bylo tyto úkoly sjednotit do jednoho delšího a evidovat je jako jeden úkol. Tento přístup je tím preferovaným, aby vše bylo správně evidované, ale ne vždycky možným.

Druhá iterace

V druhé iteraci se objevily i problémy, které nejsou spojené s nasazením nové metodiky či zavedením nových procesů, ale byly to obecné chyby týmu, které dříve nebyly tak viditelné.

Jeden z problémů byla nepřesná a neaktualizovaná specifikace od analytiků. Kvůli tomu potom docházelo k nepochopení, nebo k jinému pochopení dané problematiky, což poté vedlo k chybné testovací analýze a tedy poté byly špatně napsané testovací scénáře.

Dalším problémem byla opět forma komunikace, ale tentokrát mezi samotnými testovacími odborníky. Konkrétně to byl případ předávání informací mezi členy, kteří se věnují automatizovaným testům a členy, kteří se věnují manuálnímu testování. Docházelo potom tedy opět k vzájemnému nepochopení a automatizované testy pak byly napsané nespolehlivě, což se ukázalo na počtu chyb, které se dostaly na produkci.

Tento problém byl vyřešen přidáním kontroly jak samotných připravených manuálních testů, tak i nově vzniklým automatizovaným testům. V prvním případě byla tato kontrola prováděná jiným členem týmu, kterému v tom pomáhal analytik za danou oblast. Tak bylo zajištěno, že bylo správně pochopené zadání a nebylo nic omylem vynecháno z testování.

Kontrola nově vytvořených automatizovaných testů probíhala za pomocí odborníka na testování, který zkontroloval, že automatizovaný test opravdu kontroluje to, co je předmětem daného testovacího případu. V rámci těchto kontrol byl na Kanban tabuli přidaný sloupec „For Review“, kde čekaly jednotlivé úkoly právě na tuto kontrolu.

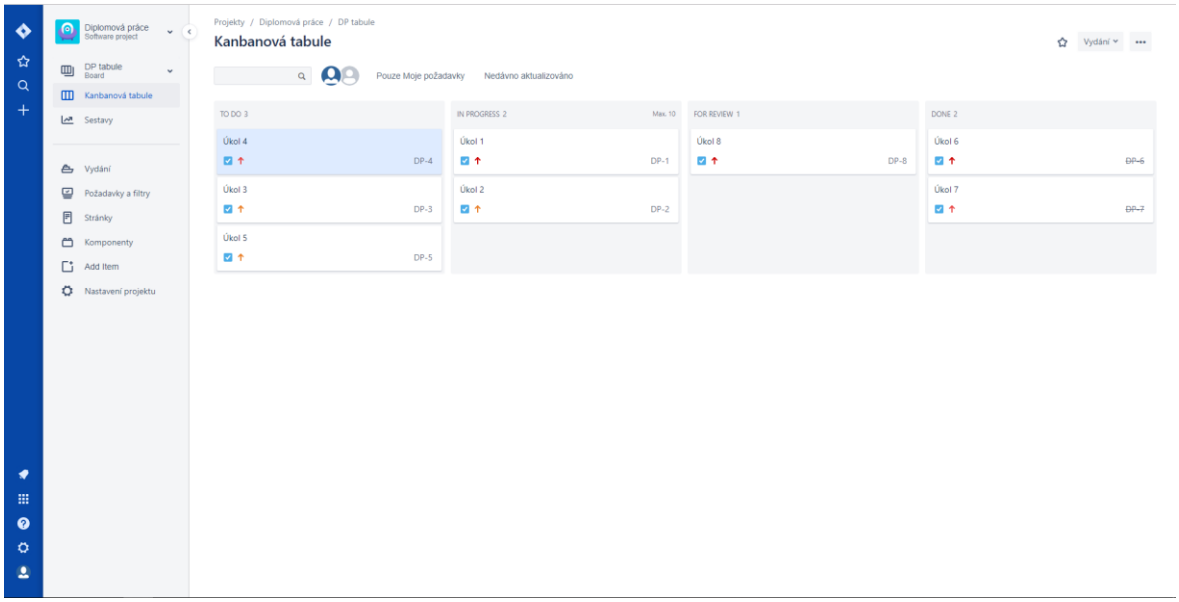
Třetí iterace

Po poměrně dlouhém čase se začaly projevovat první výsledky zavedení nových procesů a také první dlouhodobá zpětná vazba, ze které se dal odvodit nějaký trend.

Prvním problémem byla moc velká hodnota WIP. To se projevovalo hlavně na ranních schůzkách, kdy někteří členové týmu začali používat spojení jako „Na to jsem neměl čas, protože jsem dělal tento úkol“. Oba úkoly však byly ve sloupci „In Progress“
V rámci toho se ukázalo, že nebyl přesně definován sloupec „For Review“ a nebylo jasné zda ho počítat do hodnoty WIP či nikoli. Pak se stávalo, že se některým členům hromadily úkoly v tomto stavu a nebyli si jistí určením priorit jednolivých úkolů.

Řešením výše uvedených problémů bylo v první řadě přesnější definování stavu „For Review“ a následné úpravy hodnoty WIP. Po domluvě se všemi členy týmu byla hodnota WIP byla snížena na 10.

Stav „For Review“ byl tedy definován tak, že se počítal do celkové hodnoty WIP, tedy nebylo možné mít ve sloupcích „In Progress“ a „For Review“ více úkolů než 10. Úkolů je sice méně než členů v týmu, ale na některých úkolech, hlavně na validaci odvedené práce, jich z pravidla pracuje více než jeden. Tímto bude zajištěno, co možná nejrychlejší dodání hodnoty zákazníkovi.



Obrázek 27 - Kanban tabule - reálná nová (Zdroj: Autor)

Dále se ukázalo, že již není nutné mít Retrospektivu každý týden, protože poslední dobou se tam neprojednávalo zdaleka tolik věcí jako ze začátku, proto byla tato pravidelná schůzka jednou za čtrnáct dní.

Mimo výše popsané problémy byl také sledován trend, že do týmu přichází méně práce, než je schopný udělat a má tedy více členů, než by bylo potřeba. Tento vývoj bude nadále sledován a budou proti němu udělána řádná opatření, a to buď ve formě získání více práce od dalších sdružených projektů ve firmě, nebo případným přealokováním členů týmu na jiné projekty.

4.7 Shrnutí

Po zavedení metodiky Kanban a následných iteracích, byly základní pravidla a procesy víceméně nastavené.

V týmu jsou 2 vedoucí pracovníci. Jeden z nich je zodpovědný za průběh testování a druhý je zodpovědný za automatizaci. Dále byl tým rozdělený na frontendový tým, který měl 8 členů včetně vedoucích pracovníků a backendový tým, který obsahoval 6 pracovníků

Kanban tabule obsahovala 4 sloupce:

Stav **„To Do“**
Stav **„In Progress“**
Stav **„For Review“** – v tomto stavu musí zkontrolovat správnost řešení jiný pracovník, než ten, který daný úkol řešil
Stav **„Done“**

Hodnota WIP byla nastavená na 10 úkolů a byla nastavená pro sloupce „In Progress“ a „For Review“ dohromady.

Retrospektiva byla nastavena na každých 14 dní a trvala přibližně hodinu.

5 Výsledky a diskuse

5.1 Výsledky

Zavedení metodiky Kanban v daném týmu, proběhlo bez větších komplikací. Ze začátku byly nastavené teoretické procesy, které byly v následujících iteracích postupně odstraňovány. Důležitou součástí zavedení metodiky také bylo, aby bylo zcela jasně vidět, na čem tým momentálně pracuje a kolik je toho schopný udělat. Což zavedením této metodiky po jejím přizpůsobení bylo přesně možné určit. Z obecných doporučení ohledně metodiky Kanban, je třeba se nikdy nespokojit se současným nastavením procesů a snažit se je pořád zdokonalovat. Toto doporučení bylo možné sledovat na vývoji procesů ve sledovaném týmu.

Největším problémem při zavádění nové metodiky, byla hlavně neochota týmu. Bylo to hlavně kvůli pověsti agilních metodik, které jsou sice velmi oblíbené u vedoucích pracovníků, ale méně oblíbené u samotných specialistů. Tato neochota je způsobená hlavně strachem ze změn a nedostatkem informací. V tomto případě bylo zcela nezbytné přesně vysvětlit veškeré důvody zavádění nové metodiky a jasně vysvětlit všechny nové procesy. Ze začátku jsou v každém týmu členové, kteří nové nápady podporují a členové, kteří je vyloženě odmítají. Členové, kteří novou metodiku ze začátku odmítali, jí po jasném vysvětlení nakonec naštěstí přijali. V horším případě takovéto odmítání končí demotivací.

Bohužel nebylo možné udělat přesné zhodnocení zlepšení po zavedení nové agilní metodiky, protože výstupy, které byly k dispozici z předchozího fungování, nebyly zdaleka tak měřitelné, jako po zavedení nové metodiky. S jistotou ale můžeme říct, že nová metodika znatelně zlepšila přehlednost a výkon jednotlivých členů týmu. To jsme nejlépe mohli vidět na zvýšeném pracovním nasazení, které se projevilo tím, že tým stíhal udělat více pracovních úkolů, než do něj přicházelo. Tento trend bude nadále sledován a následně bude upravena velikost týmu, nebo zajištění většího množství práce.

5.2 Obecná doporučení

V diplomové práci bylo možné sledovat pravdivost obecných doporučení a také přijít s vlastními doporučeními.

Snad nejdůležitějším doporučením je zajištění kvalitní komunikace mezi vedoucími pracovníky napříč celou firmou a jednotlivými specialisty. Správná a kvalitní komunikace může odstranit velkou část obav specialistů a tím pomoci zavedením nové metodiky. V rámci této komunikace je třeba sledovat i takový jev, že někteří členové týmu nebudou chtít přijmout novou metodiku za žádnou cenu. To může způsobit to, že tito členové budou demotivovat ostatní členy týmu, a to následně velmi zhorší celý proces zavedení nové metodiky. Pokud opravdu bude vyzkoušeno vše a situace se nezmění, tak je takovýchto případech doporučeno tyto členy přesunout na jiný projekt v organizaci, pokud to bude možné, nebo tyto členy propustit.

Dalším doporučením je postupný přechod k agilním metodikám. Ať už je to v rámci zavedení metodiky v organizaci či týmu, nebo zavedení samotné metodiky jako takové. Při zavedení metodiky jako takové je doporučená metodika Kanban hned z několika důvod, které jsou detailněji popsány v teoretické části práce. Jedním z hlavních důvodů je, že metodika není destruktivní. To znamená, že se snaží zachovat co nejvíce již fungujících procesů a postupně je upravovat. Dalším důležitým důvodem je její jednoduchost v oblasti týmových setkaní. Ve srovnání s metodikou Scrum, nemá ani zdaleka tolik týmových setkání.

Výše uvedené řešení výběru a zavedení nové agilní metodiky je z větší části zobecnitelné. Nicméně, je třeba pokaždé udržet všechny důležité zájmy dané organizace a daného týmu.

5.3 Diskuse

Z diplomové práce vzešla dvě hlavní témata k diskuzi.

V první řadě, je tu možnost sjednocení jednotlivých podtýmů ve sledovaném týmu. Jedná se o backendový a frontendový tým. Obecně je v agilních metodikách doporučováno sjednocení znalostí skrz celý tým, a to hlavně z důvodů, aby mohli členové týmu pracovat na nejdůležitějších úkolech, a nejen na těch, které díky jejich současným schopnostem zvládnou. Občas totiž nastala situace, kdy tři nejdůležitější úkoly, dokázali udělat pouze dva členové týmu. V takovémto případě se prodlužuje dodání nejdůležitějších funkcionalit zákazníkovi, což je přesný opak, než o co se agilní metodiky snaží. Tomuto sjednocení již velmi napomáhá skutečnost, že jsou všechny úkoly uvedené na jednom místě a už teď je možné si brát takové úkoly, které daní členové uznají za vhodné vzhledem k jejich schopnostem. Toto téma bude dále probíráno s ostatními vedoucími pracovníky.

Druhým tématem k diskuzi je zavedení jedné z metodik, které jsou vhodné pro celou organizaci. V práci jsou zmíněné dvě takovéto metodiky, LeSS a SAFe. Problémem při zavedení těchto celoorganizačních metodik je, hlavně politika dané organizace. V tomto případě je potřeba sjednotit celou organizaci, která má tisíce zaměstnanců. Jedná se o velkou finanční zátěž a je třeba tomu věnovat velké množství času. Výhodami těchto metodik je, že celá organizace pracuje stejným způsobem, se stejnými cíli a se stejnými prioritami. To vede k možnosti rychlé adaptace současnému trhu a tím pádem k velké konkurenční výhodě. Dobrým začátkem by bylo, alespoň sjednocení výše popisovaného projektu, který sám o sobě je rozdělen do čtyř výše zmíněných týmů.

6 Závěr

Hlavním cílem bylo vybrat vhodnou agilní metodiku a zavést jí v týmu na konkrétním projektu. Tento cíl byl úspěšně splněn. Vhodná metodika byla vybrána pomocí systému hodnocení a výběru metodik METES, který byl upraven podle požadavků daného projektu a týmu. Byla vybrána metodika Kanban. Při zavádění metodiky byly řešeny problémy, jako například zvolení vhodné hodnoty WIP, definování struktury Kanban tabule, napojení testovacího týmu na ostatní týmy atd. Všechny problémy se podařilo úspěšně odstranit a dále se pracuje na dalších možných vylepšeních.

Dílčími cíli byla analýza dostupných zdrojů, popis jednotlivých metodik a výběr vhodného nástroje pro podporu vybrané agilní metodiky. Všechny dílčí cíle práce byly úspěšně splněny.

Vhodný nástroj pro podporu agilní metodiky byl vybrán pomocí kritérií, které definovali vedoucí pracovníci daného týmu. Kritéria byla následně zhodnocena pro každý nástroj. Poté byla vybrána kompromisní varianta za pomocí metody AHP. Byl zvolen nástroj Atlassian Jira Software od firmy Atlassian, Inc.

Při zavádění nové agilní metodiky se prokázalo, že zavedení není jednorázovou činností, ale jedná se spíš o postupný proces, který pomáhá k co nejlepšímu dosažení cílů týmů.

7 Seznam použitých zdrojů

[1] *The State of Agile Report 2017* [online]. [vid. 2017-01-28]. Dostupné z: https://agilebb.nl/wp-content/uploads/2018/04/versionone-12th-annual-state-of-agile-report.pdf

[2] *2017 State of Scrum Report* [online]. [vid. 2017-01-28]. Dostupné z: http://www.pm-progetti.it/download/scrum-alliance-state-of-scrum-2015.pdf

[3] JAROSLAV, Ráček. *Agilní metodiky vývoje SW* [online]. [vid. 2017-01-28]. Dostupné z: https://is.muni.cz/el/1433/podzim2013/PA017/um/SWE2_07_agilni.pdf

[4] *Spirálový model* [online]. [vid. 2017-01-28]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/spiralovy-model/

[5] *Manifest Agilního vývoje software* [online]. [vid. 2017-01-28]. Dostupné z: http://agilemanifesto.org/iso/cs/manifesto.html

[6] *FDD: Processes* [online]. [vid. 2017-01-28]. Dostupné z: http://www.step10.com/SoftwareProcess/FeatureDrivenDevelopment/FDDProcesses.html

[7] *ExtremeProgramming.org home* [online]. [vid. 2017-01-28]. Dostupné z: http://www.extremeprogramming.org/map/project.html

[8] ŠOCHOVÁ, Zuzana. *Zuzi's blog* [online]. 2015. Agile and Lean, Scrum, Kanban, XP @ Business. Dostupné z: http://soch.cz/blog/

[9] *AGILNĚ VE VELKÝCH ORGANIZACÍCH* [online]. [vid. 2017-01-02]. Dostupné z: https://blog.think-forth.com/2016/07/14/agilne-ve-velkych-organizacich/#comments

[10] HEFNEROVÁ, Lucie. *Kanban a možnosti jeho využití v bankovním prostředí*. Praha, 2015. b.n.

[11] SUTHERLAND, Jeff a Ken SCHWABER. *The Scrum Guide*. The Definitive Guide to Scrum: The Rules of the Game

[12] DALALAH, Ahmad. Extreme Programming: Strengths and Weaknesses [online]. 2013. Dostupné z: http://www.acit2k.org/ACIT/2013Proceedings/132.pdf

[13] ANDERSON., David J. *Kanban*. 1. vydání. Sequim, Wash: Blue Hole Press, 2010. successful evolutionary change in your software business. ISBN 0984521402.

[14] BUCHALCEVOVÁ, Alena. *Zlepšování procesů při budování informačních systémů*. Praha: Oeconomica, 2018. ISBN 9788024522357.

[15] KO, andrew J. *A brief history of software engineering* [online]. Nedatováno. Dostupné z: https://faculty.washington.edu/ajko/books/cooperative-software-development/history.html

[16] *Trojimperativ projektu a jeho význam pro praxi* [online]. Nedatováno. Dostupné z: https://www.pmconsulting.cz/pm-wiki/trojimperativ-projektu/

[17] HIGHSMITH, James A. *Agile project management*. 2nd ed. Upper Saddle River, NJ: Addison-Wesley, nedatováno. creating innovative products. ISBN 0321658396. Chaos Report - Q&A with Jennifer Lynch. In. Nedatováno.

[18] HIGHSMITH, Jim. Beyond Scope, Schedule, and Cost: The Agile Triangle. nedatováno.

[19] MERSISO, Anthony. *Agile Projects are More Successful than Traditional Projects* [online]. [vid. 2018-04-01]. Dostupné z:

https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/
[20] SÍKA Jaroslav. *Projektové řízení v cloudu* [online]. [vid. 2014-04-01]. Dostupné z: https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/
[21] BRUCKNER Tomáš *Tvorba informačních systémů: Principy, metodiky, architektury*. Praha: Grada, 2012. ISBN 9788024522357.
[22] *Vodopádový model* [online]. Nedatováno. Dostupné z: https://cs.wikipedia.org/wiki/Vodop%C3%A1dov%C3%BD_model
[23] ANWAR, Ashraf. *A Review of RUP (Rational Unified Process)* [online]. [vid. 2014] Dostupné z: https://www.cscjournals.org/manuscript/Journals/IJSE/Volume5/Issue2/IJSE-142.pdf
[24] *Životní cyklus softwaru* [online]. [vid. 2016] Dostupné z: http://szz.g6.cz/doku.php?id=temata:30-zivotni_cyklus_softwaru:main
[25] *Project Management Methods, Methodologies, and Frameworks* [online]. [vid. 2018] Dostupné z: https://www.paymoapp.com/academy/project-management-methodologies/
[26] Dostupné z: https://www.shutterstock.com/cs/image-vector/whiteboard-post-notes-agile-software-development-563390743
[27] *What is a Kanban Card in lean manufacturing?* [online]. [vid. 2017-10-05] Dostupné z: https://blog.v-comply.com/kanban-card/
[28] GILL, Navdeep Singh. *TDD, Test and Behavior Driven Development and Unit Testing in Python* [online]. [vid. 2017-12-08] Dostupné z: https://www.xenonstack.com/blog/test-tdd-bdd-unit-testing-python/
[29] LANGROVÁ, Kamila. *Přehled rolí v jednotlivých metodikách* [online]. [vid. 2014]. Dostupné z: https://spicenter.vse.cz/wp-content/uploads/2018/08/semestralni_prace/15_02/Langrova-4IT421_Langrova_Kamila_xlank10.pdf
[30] *Rapid Application Development* [online]. Nedatováno. Dostupné z: https://www.mendix.com/rapid-application-development/
[31] AMBLER, Scott W. *Feature Driven Development (FDD) and Agile Modeling* [online]. Nedatováno. Dostupné z: http://agilemodeling.com/essays/fdd.htm
[32] RANGARAJ,Jyothi. *SAFe Methodology Tutorial: What is Scaled Agile Framework* [online]. Nedatováno. Dostupné z: https://www.guru99.com/scaled-agile-framework.html
[33] *Large-Scale Scrum* [online]. Nedatováno. Dostupné z: https://less.works/
[34] TUCKER, Greg. *Analytic Hierarchy Process* [online]. [vid. 2017-01-03] Dostupné z: https://www.itsminfo.com/analytic-hierarchy-process/
[35] SAATY, R. W. *The analytic hierarchy process—what it is and how it is used* [online]. Nedatováno. Dostupné z: https://www.sciencedirect.com/science/article/pii/0270025587904738
[36] *Ganttův diagram (Gantt Chart)* [online]. Nedatováno. Dostupné z: https://managementmania.com/cs/ganttuv-diagram

8 Přílohy

Příloha A: Hlavní výběrová kritéria systému METES

Důležitost produktu	
Stupnice hodnocení:	**Popis**
0	jen pilotní projekt
1	doplňkový systém
2	systém podporující fungování organizace
3	systém kritický pro poslání (národní organizace)
4	systém kritický pro poslání (nadnárodní organizace)
5	systém, na kterém závisí životy lidí
Délka projektu	
Stupnice hodnocení:	**Popis**
0	do 1 měsíce
1	do 3 měsíců
2	do 6 měsíců
3	do 12 měsíců
4	do 24 měsíců
5	nad 24 měsíců
Stálost požadavků	
Stupnice hodnocení:	**Popis**
0	požadavky není předem možné detailně stanovit
1	požadavky se více než z 50% mění
2	procento změn požadavků je cca 30%
3	požadavky lze definovat předem, mění se jen priority požadavků
4	požadavky lze definovat předem, mění se, ale snahou je změny potlačovat 5 – požadavky lze definovat předem a nemění se
5	požadavky lze definovat předem a nemění se

Znuvupoužitelnost	
Stupnice hodnocení:	**Popis**
0	cílem není znuvupoužitelnost
1	snaha používat již hotové komponenty
2	snaha vytvářet znovupoužitelné třídy v rámci projektu
3	snaha vytvářet znovupoužitelné spustitelné komponenty v rámci projektu
4	snaha vytvářet znovupoužitelné spustitelné komponenty v rámci organizace
5	cílem je maximální znuvupoužitelnost v rámci organizace

Velikost řešení	
Stupnice hodnocení:	**Popis**
0	1 až 10 případů užití
1	11 až 40 případů užití
2	41 až 100 případů užití
3	101 až 200 případů užití
4	201 až 300 případů užití
5	300 a více případů užití

Zkušenost manažera projektu	
Stupnice hodnocení:	**Popis**
0	5 a více
1	4 až 5 let
2	3 až 4 roky
3	2 až 3 roky
4	1 až 2 roky
5	0 až 1 rok

Kvalifikace členů týmů	
Stupnice hodnocení:	**Popis**
0	více než 70% členů týmu je kvalifikovaných, s širokým zaměřením
1	více než 70% členů týmu je kvalifikovaných, ale specializovaných
2	cca 50% členů týmu je málo kvalifikovaných
3	více než 60% členů týmu je málo kvalifikovaných
4	více než 70% členů týmu je málo kvalifikovaných
5	více než 80% členů týmu je málo kvalifikovaných

Motivace členů týmu	
Stupnice hodnocení:	**Popis**
0	motivovaní jedinci vysokých morálních kvalit, sdílí znalosti, sami se organizují
1	aktivní, motivovaní jedinci, sdílí znalosti
2	plní zadané úkoly, sdílí znalosti
3	plní zadané úkoly, nesdílí znalosti
4	jedinci jsou špatně motivováni a snaží se vyhýbat úkolům, nesdílí znalosti
5	žádná motivace

Dostupnost uživatelů	
Stupnice hodnocení:	**Popis**
0	uživatel je součástí týmu, má odpovědnost za požadavky
1	uživatel je k dispozici denně
2	uživatel je k dispozici kdykoliv na vyžádání
3	uživatel je k dispozici na začátku, konci a v průběhu projektu, v předem určených milnících
4	uživatel je k dispozici jen na začátku a na konci projektu
5	uživatel není dostupný

Velikost týmu	
Stupnice hodnocení:	**Popis**
0	1 až 4
1	4 až 10
2	11 až 20
3	21 až 50
4	51 až 100
5	více než 100
Rozmístění	
Stupnice hodnocení:	**Popis**
0	0 – v jedné místnosti
1	1 – v jedné budově
2	2 – více míst v jednom městě
3	3 – dvě místa v jedné zemi
4	4 – více míst v jedné zemi
5	5 – mimo jednu zem

Příloha B: Detailní výpočetní tabulky pro volbu metodiky podle systému METES

	Důležitost projektu	Délka projektu	Stálost projektu	Znovupoužitelnost	Velikost řešení	Zkušenost manažera projektu	Kvalifikace členů týmu	Motivace členů týmu	Dostupnost uživatelů	Velikost týmu	Rozmístění	Geometrický průměr	VÁHY
Důležitost projektu	1	5,00	7,00	7,00	5,00	7,00	7,00	7,00	3,00	3,00	5	4,5868	0,2925
Délka projektu	0,20	1	7,00	7,00	5,00	5,00	7,00	7,00	1,00	1,00	1,00	2,3488	0,1498
Stálost požadavků	0,14	0,14	1	1,00	1,00	3,00	5,00	7,00	0,33	0,33	0,14	0,7354	0,0469
Znovupoužitelnost	0,14	0,14	1,00	1	1,00	3,00	1,00	3,00	0,20	0,20	0,20	0,5527	0,0352
Velikost řešení	0,20	0,20	1,00	1,00	1	1,00	3,00	3,00	0,20	0,33	0,33	0,6447	0,0411
Zkušenost manažera projektu	0,14	0,20	0,33	0,33	1,00	1	1,00	1,00	0,20	0,20	0,20	0,3822	0,0244
Kvalifikace členů týmu	0,14	0,14	0,20	1,00	0,33	1,00	1	0,33	0,20	0,20	0,20	0,3202	0,0204
Motivace členů týmu	0,14	0,14	0,14	0,33	0,33	1,00	3,00	1	0,14	0,14	0,14	0,3131	0,0200
Dostupnost uživatelů	0,33	1,00	1,00	5,00	5,00	3,00	3,00	3,00	1	1,00	3,00	1,8080	0,1153
Velikost týmu	0,33	1,00	3,00	5,00	3,00	5,00	5,00	7,00	1,00	1	3,00	2,2605	0,1442
Rozmístění	0,20	1,00	7,00	5,00	3,00	5,00	5,00	7,00	0,33	0,33	1	1,7273	0,1102
SUMA	2,9810	9,9714	28,6762	33,6667	25,6667	35,0000	41,0000	46,3333	7,6095	7,7429	14,2190	15,6797	1,0000

FDD	váhy test	min	max	opt	Můj projekt	od opt.	vážené hodnoty
Důležitost projektu	0,2925	0	3	3	2	-1	0,2925
Délka projektu	0,1498	0	4	3	5	2	0,2996
Stálost požadavků	0,0469	1	3	1	1	0	0,0000
Znovupoužitelnost	0,0352	0	2	1	1	0	0,0000
Velikost řešení	0,0411	0	4	4	3	-1	0,0411
Zkušenost manažera projektu	0,0244	0	3	3	3	0	0,0000
Kvalifikace členů týmu	0,0204	0	3	2	0	-2	0,0408
Motivace členů týmu	0,0200	0	2	2	2	0	0,0000
Dostupnost uživatelů	0,1153	0	1	1	2	1	0,1153
Velikost týmu	0,1442	0	4	3	2	-1	0,1442
Rozmístění	0,1102	0	1	1	0	-1	0,1102
							1,0437

SCRUM	váhy test	min	max	opt	Můj projekt	od opt.	vážené hodnoty
Důležitost projektu	0,2925	0	3	3	2	-1	0,2925
Délka projektu	0,1498	1	5	3	5	2	0,2996
Stálost požadavků	0,0469	0	3	0	1	1	0,0469
Znovupoužitelnost	0,0352	0	1	1	1	0	0,0000
Velikost řešení	0,0411	0	5	5	3	-2	0,0822
Zkušenost manažera projektu	0,0244	0	2	1	3	2	0,0487
Kvalifikace členů týmu	0,0204	0	1	0	0	0	0,0000
Motivace členů týmu	0,0200	0	1	0	2	2	0,0399
Dostupnost uživatelů	0,1153	0	1	0	2	2	0,2306
Velikost týmu	0,1442	1	4	1	2	1	0,1442
Rozmístění	0,1102	0	3	3	0	-3	0,3305
							1,5152

XP	váhy test	min	max	opt	Můj projekt	od opt.	vážené hodnoty
Důležitost projektu	0,2925	0	3	3	2	-1	0,2925
Délka projektu	0,1498	0	4	2	5	3	0,4494
Stálost požadavků	0,0469	0	3	0	1	1	0,0469
Znovupoužitelnost	0,0352	0	1	1	1	0	0,0000
Velikost řešení	0,0411	0	3	3	3	0	0,0000
Zkušenost manažera projektu	0,0244	0	2	2	3	1	0,0244
Kvalifikace členů týmu	0,0204	0	1	1	0	-1	0,0204
Motivace členů týmu	0,0200	0	1	1	2	1	0,0200
Dostupnost uživatelů	0,1153	0	1	0	2	2	0,2306
Velikost týmu	0,1442	0	1	1	2	1	0,1442
Rozmístění	0,1102	0	1	1	0	-1	0,1102
							1,3385

Kanban	váhy test	min	max	opt	Můj projekt	od opt.	vážené hodnoty
Důležitost projektu	0,2925	0	3	2	2	0	0,0000
Délka projektu	0,1498	1	5	3	5	2	0,2996
Stálost požadavků	0,0469	0	3	0	1	1	0,0469
Znovupoužitelnost	0,0352	0	1	0	1	1	0,0352
Velikost řešení	0,0411	0	3	3	3	0	0,0000
Zkušenost manažera projektu	0,0244	0	3	3	3	0	0,0000
Kvalifikace členů týmu	0,0204	0	3	1	0	-1	0,0204
Motivace členů týmu	0,0200	0	3	1	2	1	0,0200
Dostupnost uživatelů	0,1153	0	3	2	2	0	0,0000
Velikost týmu	0,1442	1	4	1	2	1	0,1442
Rozmístění	0,1102	0	3	1	0	-1	0,1102
							0,6765

LeSS	váhy test	min	max	opt	Můj projekt	od opt.	vážené hodnoty
Důležitost projektu	0,2925	0	5	4	2	-2	0,5851
Délka projektu	0,1498	2	5	5	5	0	0,0000
Stálost požadavků	0,0469	0	5	1	1	0	0,0000
Znovupoužitelnost	0,0352	0	3	0	1	1	0,0352
Velikost řešení	0,0411	3	5	5	3	-2	0,0822
Zkušenost manažera projektu	0,0244	0	2	1	3	2	0,0487
Kvalifikace členů týmu	0,0204	0	1	1	0	-1	0,0204
Motivace členů týmu	0,0200	0	1	1	2	1	0,0200
Dostupnost uživatelů	0,1153	1	1	1	2	1	0,1153
Velikost týmu	0,1442	5	5	5	2	-3	0,4325
Rozmístění	0,1102	1	5	5	0	-5	0,5508
							1,8903

SAFe	váhy test	min	max	opt	Můj projekt	od opt.	vážené hodnoty
Důležitost projektu	0,2925	0	5	4	2	-2	0,5851
Délka projektu	0,1498	2	5	3	5	2	0,2996
Stálost požadavků	0,0469	0	5	1	1	0	0,0000
Znovupoužitelnost	0,0352	0	3	1	1	0	0,0000
Velikost řešení	0,0411	3	5	5	3	-2	0,0822
Zkušenost manažera projektu	0,0244	0	2	1	3	2	0,0487
Kvalifikace členů týmu	0,0204	0	1	1	0	-1	0,0204
Motivace členů týmu	0,0200	0	1	1	2	1	0,0200
Dostupnost uživatelů	0,1153	1	2	2	2	0	0,0000
Velikost týmu	0,1442	5	5	5	2	-3	0,4325
Rozmístění	0,1102	1	5	5	0	-5	0,5508
							2,0393

Příloha C: Detailní výpočetní tabulky pro volbu nástroje pomocí metody AHP

	Cena	Dostupnost na mobilu	Nastavení oprávnění	Automatické Měření času práce	Podpora	Retrospektiva	Podpora testování	Geometrický průměr	VÁHY
Cena	1	7	3	3	3	5	3	3,1133	0,3374
Dostupnost na mobilu	0,14	1	0,20	0,20	0,20	0,33	0,20	0,2580	0,0280
Nastavení oprávnění	0,33	5,00	1	0,33	0,33	3	0,33	0,7859	0,0852
Automatické Měření času práce	0,33	5,00	3,00	1	1,00	3	1,00	1,4724	0,1596
Podpora	0,33	5,00	3,00	1,00	1	5	1	1,5838	0,1717
Retrospektiva	0,20	3,00	0,33	0,33	0,20	1	0,20	0,4288	0,0465
Podpora testování	0,33	5,00	3,00	1,00	1,00	5,00	1	1,5838	0,1717
SUMA	2,6762	31,0000	13,5333	6,8667	6,7333	22,3333	6,7333	9,2261	1,0000

Cena	Atlassian JIRA	VersionOne	CA Agile Central	LeanKit	Trello	Youtrack	geometrický průměr	VÁHY	Upravené podle vah kritérií
Atlassian JIRA	1	7,00	7	7	3	0,33	2,6458	0,2735	0,0923
VersionOne	0,14	1	1	1	0,20	0,11	0,3834	0,0396	0,0134
CA Agile Central	0,14	1,00	1	1,00	0,20	0,11	0,3834	0,0396	0,0134
LeanKit	0,14	1,00	1,00	1	0,20	0,11	0,3834	0,0396	0,0134
Trello	0,33	5,00	5,00	5,00	1	0,33	1,5504	0,1603	0,0541
Youtrack	3,00	9,00	9,00	9,00	3	1	4,3267	0,4473	0,1509
SUMA	4,7619	24,0000	24,0000	24,0000	7,6000	2,0000	9,6730	1,0000	0,3374

Dostupnost na mobilu	Atlassian JIRA	VersionOne	CA Agile Central	LeanKit	Trello	Youtrack	geometrický průměr	VÁHY	Upravené podle vah kritérií
Atlassian JIRA	1	3,00	3	1	1	1,00	1,4422	0,2143	0,0060
VersionOne	0,33	1	1	0,33	0,33	0,33	0,4807	0,0714	0,0020
CA Agile Central	0,33	1,00	1	0,33	0,33	0,33	0,4807	0,0714	0,0020
LeanKit	1,00	3,00	3,00	1	1,00	1,00	1,4422	0,2143	0,0060
Trello	1,00	3,00	3,00	1,00	1	1,00	1,4422	0,2143	0,0060
Youtrack	1,00	3,00	3,00	1,00	1	1	1,4422	0,2143	0,0060
SUMA	4,6667	14,0000	14,0000	4,6667	4,6667	4,6667	6,7305	1,0000	0,0280

Nastavení oprávnění	Atlassian JIRA	VersionOne	CA Agile Central	LeanKit	Trello	Youtrack	geometrický průměr	VÁHY	Upravené podle vah kritérií
Atlassian JIRA	1	1,00	5	1	5	1,00	1,7100	0,2273	0,0194
VersionOne	1,00	1	5	1	5,00	1,00	1,7100	0,2273	0,0194
CA Agile Central	0,20	0,20	1	0,20	1,00	0,20	0,3420	0,0455	0,0039
LeanKit	1,00	1,00	5,00	1	5,00	1,00	1,7100	0,2273	0,0194
Trello	0,20	0,20	1,00	0,20	1	0,20	0,3420	0,0455	0,0039
Youtrack	1,00	1,00	5,00	1,00	5	1	1,7100	0,2273	0,0194
SUMA	4,4000	4,4000	22,0000	4,4000	22,0000	4,4000	7,5239	1,0000	0,0852

Automatické Měření času práce	Atlassian JIRA	VersionOne	CA Agile Central	LeanKit	Trello	Youtrack	geometrický průměr	VÁHY	Upravené podle vah kritérií
Atlassian JIRA	1	5,00	5	5	3	5,00	3,5116	0,4507	0,0719
VersionOne	0,20	1	1	1	0,33	1,00	0,6368	0,0817	0,0130
CA Agile Central	0,20	1,00	1	1,00	0,33	1,00	0,6368	0,0817	0,0130
LeanKit	0,20	1,00	1,00	1	0,33	1,00	0,6368	0,0817	0,0130
Trello	0,33	3,00	3,00	3,00	1	3,00	1,7321	0,2223	0,0355
Youtrack	0,20	1,00	1,00	1,00	0,33	1	0,6368	0,0817	0,0130
SUMA	2,1333	12,0000	12,0000	12,0000	5,3333	12,0000	7,7907	1,0000	0,1596

Podpora	Atlassian JIRA	VersionOne	CA Agile Central	LeanKit	Trello	Youtrack	geometrický průměr	VÁHY	Upravené podle vah kritérií
Atlassian JIRA	1	3,00	3	5	5	1,00	2,4662	0,3177	0,0545
VersionOne	0,33	1	1	3	3,00	0,33	1,0000	0,1288	0,0221
CA Agile Central	0,33	1,00	1	0,33	0,33	0,20	0,4415	0,0569	0,0098
LeanKit	0,20	0,33	3,00	1	1,00	0,20	0,5848	0,0753	0,0129
Trello	0,20	0,33	3,00	1,00	1	0,20	0,5848	0,0753	0,0129
Youtrack	1,00	3,00	5,00	5,00	5	1	2,6854	0,3459	0,0594
SUMA	3,0667	8,6667	16,0000	15,3333	15,3333	2,9333	7,7627	1,0000	0,1717

Retrospektiva	Atlassian JIRA	VersionOne	CA Agile Central	LeanKit	Trello	Youtrack	geometrický průměr	VÁHY	Upravené podle vah kritérií
Atlassian JIRA	1	1,00	5	5	5	5,00	2,9240	0,3571	0,0166
VersionOne	1,00	1	5	5	5,00	5,00	2,9240	0,3571	0,0166
CA Agile Central	0,20	0,20	1	1,00	1,00	1,00	0,5848	0,0714	0,0033
LeanKit	0,20	0,20	1,00	1	1,00	1,00	0,5848	0,0714	0,0033
Trello	0,20	0,20	1,00	1,00	1	1,00	0,5848	0,0714	0,0033
Youtrack	0,20	0,20	1,00	1,00	1	1	0,5848	0,0714	0,0033
SUMA	2,8000	2,8000	14,0000	14,0000	14,0000	14,0000	8,1872	1,0000	0,0465

Podpora testování	Atlassian JIRA	VersionOne	CA Agile Central	LeanKit	Trello	Youtrack	geometrický průměr	VÁHY	Upravené podle vah kritérií
Atlassian JIRA	1	3,00	1	7	3	3,00	2,3956	0,3062	0,0526
VersionOne	0,33	1	0,33	5	1,00	1,00	0,9067	0,1159	0,0199
CA Agile Central	1,00	3,00	1	7,00	3,00	3,00	2,3956	0,3062	0,0526
LeanKit	0,14	0,20	0,14	1	0,14	0,14	0,2090	0,0267	0,0046
Trello	0,33	1,00	0,33	7,00	1	1,00	0,9590	0,1226	0,0210
Youtrack	0,33	1,00	0,33	7,00	1	1	0,9590	0,1226	0,0210
SUMA	3,1429	9,2000	3,1429	34,0000	9,1429	9,1429	7,8248	1,0000	0,1717