ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Bakalářská práce

Metody řízení vývoje softwaru ve vybrané organizaci

Management methods of software development in a selected organization

Matěj Vyrut

Plzeň 2023

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma

Metody řízení vývoje softwaru ve vybrané organizaci

vypracoval/a samostatně pod odborným dohledem vedoucí/vedoucího bakalářské práce za použití pramenů uvedených v přiložené bibliografii.

Plzeň dne 24. dubna 2023 v. r Matěj Vyrut

Zásady pro vypracování práce

1. Vymezte základní metodiky vhodné pro řízení vývoje softwaru.
2. Představte vybraný podnikatelský subjekt a projekt.
3. Popiště nasazení postupu a metodiky do praxe a vyhodnoťte.

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce panu Ing. et Ing. Jiřímu Pešíkovi, za metodologickou pomoc při zpracovaní práce a za čas, který mi věnoval. Poděkování náleží také vývojovému týmu podnikatelskému subjektu, za obětavost a přízeň během naší spolupráce.

Obsah

Úvod

V průběhu neustálého vývoje softwarového průmyslu se různé metody řízení vývoje softwaru stávají běžně používanými navigačními mapami vedoucími k úspěchu v oblasti softwarových projektů. Mezi nejčastěji používané metodiky patří Waterfall a Agile, které se stávají hlavními směrovkami, spolu s méně známými, avšak stejně hodnotnými alternativami, jako jsou spirálové a hybridní metodiky. Při rozhodování o metodikách se manažeři často řídí preferencemi týmu nebo historickými vzory, jako by to byly cestovatelské rady. Avšak každý softwarový projekt je svým způsobem jedinečný a má vlastnosti, které se liší od předchozích projektů, na kterých tým pracoval. Pro každý projekt je klíčové zvolit vhodnou metodiku, která umožní týmům vyvíjet software splňující potřeby zákazníků v rámci časových a rozpočtových omezení, jako by to byl kompas ukazující správný směr. Otázkou, kterou by si měl každý manažer položit, je: "Jak vybrat tu nejvhodnější metodiku, která se stane průvodcem pro konkrétní projekt a jeho tým?"

Cílem této bakalářské práce je vymezit základní metodiky vhodné pro řízení vývoje softwaru, představit vybraný podnikatelský subjekt vyvíjející softwarový projekt a následně popsat nasazení vybrané metodiky do praxe a vyhodnotit celý postup.

Práce je rozdělena do pěti kapitol. První kapitola obsahuje vymezení pojmů jako je například projektové řízení, jeho proces, a hlavní cíl, jímž je projekt.

Druhá kapitola se zabývá metodami vývoje softwaru, především z čeho se skládají, jak jsou plánovány a organizovány. Dále je kapitola popisuje rozdíly mezi tradičním a agilním přístupem k vývoji softwaru. Obě kapitoly jsou zpracované z rešerše existující odborné literatury a jejím shrnutím.

Třetí kapitola je věnována agilní metodice, zejména metodice Scrum, Kanban, a Scrumban, která slučuje dvě zmíněné metodiky dohromady. Tato kapitola popisuje pojmy, postupy, cíle a uplatnění zmíněných metodik.

Čtvrtá kapitola popisuje implementaci metodiky Scrumban do probíhajícího projektu. Popisuje nástroje, které byly k implementaci Scumbanu použity, proces plánování, a kroky, které bylo nutné vykonat k udržení a vylepšení nově zavedené metodiky. Kapitola dále obsahuje rozhovory s odborníky v oblasti agilních metod, kteří popisují jejich zkušenosti a rady z praxe. Popsané události vycházejí z interních materiálů projektu a

osobních rozhovorů s členy vývojového týmu. Pátá kapitola obsahuje vyhodnocení procesu implementace metodiky Scrumban do výše zmíněného projektu a také uvádí návrhy na zlepšení procesů pro budoucí implementaci agilních principů.

1 Vymezení pojmů

1.1 Projektové řízení

Projektovým řízením *(project management)* se rozumí soubor norem, doporučení a best of practice zkušeností, popisujících, jak řídit projekt (Doležal, 2016).

Projektové řízení podle PMI (Project management institute) v sobě zahrnuje 5 hlavních oblastí.

Mezi tyto oblasti patří:

- Zahájení (definování) – definování projektových cílů a účelu, zahájení aktivit
- Plánování – naplánování, jak budou splněny požadavky a cíle projektu (které metody a postupy budou použity); specifikace provedení, časového plánu a finančního rozpočtu
- Vykonání – realizace výstupů a dodávek naplánovaným způsobem
- Sledování (monitorování) – kontrola stavu a postupu projektových prací, aby byly včas zjištěny odchylky od plánu, a ty mohly být zavčas korigovány
- Ukončení – ověření, že hotový úkol odpovídá aktuální definici toho, co se mělo udělat (odpovídá specifikaci v zadání), a uzavření všech nedokončených prací, např. dokumentace (včetně dokumentace vyhodnocení průběhu projektu) (Doležal, 2016)

1.2 Projekt

Projekt je jednoznačně nejdůležitějším prvkem celého projektového řízení. Jedná se o řízený a dočasný proces, který má přesná pravidla řízení a regulace. V opačném případě by se pak jednalo pouze o sled úkolů (Svozilová, 2016)

Nejdůležitějším prvkem projektového řízení je projekt. Projekt samozřejmě nemá pouze jednu všeobecnou definici. Svozilová (2016) uvádí dvě základní, přičemž první podle profesora Kerznera a druhou, která vychází z formulace podle PMI:

„Projekt je jakýkoliv jedinečný sled aktivit a úkolů, který má:

- *dán specifický cíl, který má být jeho realizací splněn*

- *definováno datum začátku a konce uskutečnění*
- *stanoven rámec pro čerpání zdrojů potřebných pro jeho realizaci.“*

„Projekt je dočasné úsilí vynaložené na vytvoření unikátního produktu, služby nebo určitého výsledku.“ (str. 64).

1.3 Software

Podle Myslína (2016) lze software definovat jako všechny programové prvky, které umožňují interakci s hardwarem na základě uživatelských pokynů. K této kategorii patří desktopové, mobilní a webové aplikace, stejně jako operační systémy. Dříve se vývoj softwaru často svěřoval jednotlivcům nebo malým skupinám programátorů, ale dnes se profesionální vývoj softwaru často řídí komplexními pravidly a cykly vedenými celými týmy odborníků. Tyto pravidla a cykly jsou navrženy tak, aby zajistily účinnost a kvalitu procesu vývoje softwaru.

1.4 Metoda

Metoda je předem stanovený postup, který má za cíl dosáhnout předem určeného výsledku. Abychom tento cíl dosáhli, je nutné mít podrobné znalosti o tom, jak postup použít. Tento proces je charakterizován záměrností (vztahující se k výzkumnému cíli) a systematičností (metoda je uplatňována v rámci teoreticky zdůvodněného postupu). Metodologie poskytuje východiska pro zdůvodnění postupu a hraje klíčovou roli při zaměření vědeckého zkoumání a výběru vědeckých výzkumných metod (Ochrana, 2010).

2 Metody vývoje softwaru

Co jsou to metody vývoje softwaru? (Vijayasarathy & Butler, 2016) uvedli následující definici: Metoda vývoje softwaru je rámec, který se používá pro strukturování, plánování a řízení procesu vývoje informačního systému. Rámec zahrnuje, ale není omezen na:

- Jak je tvořen vývojový tým
- Jak je plánována a prováděna fáze vývoje softwaru
- Jak jsou organizovány komunikace uvnitř týmu a mimo tým (k zákazníkům a externím týmům)

2.1 Tradiční přístup

Cílem této podkapitoly je představení dvou tradičních metod, kterými jsou Spirálová metoda a Vodopádová metoda. Tyto dvě metody můžeme zařadit mezi nejznámější tradiční metody řízení projektů a jsou i v současnosti stále používány ve firmách.

Tradiční metodiky jsou rigorózní, nebo bychom je také mohli nazvat metodikami „těžkými“ ve smyslu, že jsou velmi podrobné a formální. Vycházejí z přesvědčení, že se všechny procesy při vývoji dají popsat, plánovat, řídit a měřit. Z toho důvodu podrobně a přesně definují veškeré procesy, činnosti a jimi vytvářené produkty (Bruckner, 2012).

2.1.1 Vodopádová metoda

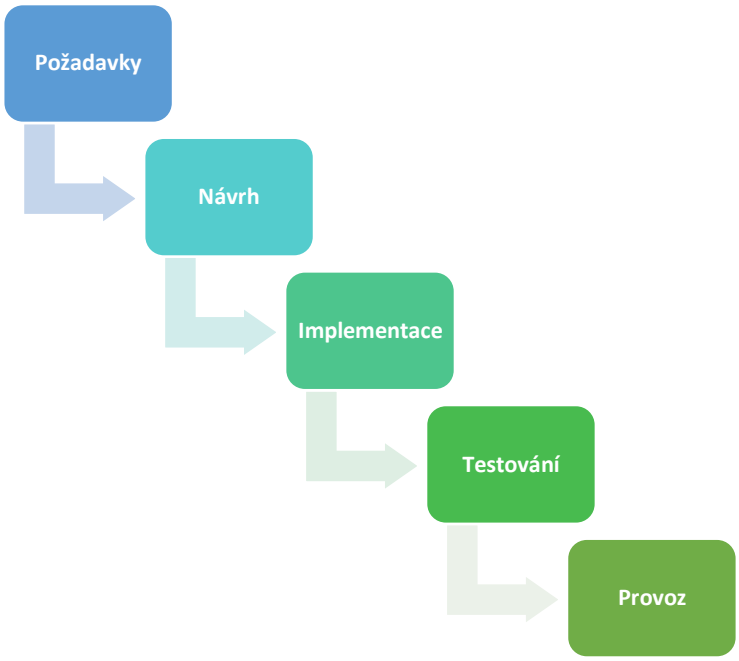
Nejstarší a nejznámější tradiční metodou řízení projektů je metoda Vodopádová (Testování Softwaru, n.d.).

Pojmenování Vodopádového modelu vychází z přirovnání posloupnosti jednotlivých fází k vodě protékající vodopádem, jak je znázorněno na schématu níže (Obrázek 2). Poprvé jej definoval Winston W. Royce ve svém článku „Managing the Development of Large Software Systém“ už v roce 1970. Vývoj tohohle modelu měl za cíl lepšího vyrovnání se s rostoucí složitostí produktů v leteckém průmyslu (Testování Softwaru, n.d.). Tato metoda je však pro současné projekty již značně nedostačující, ale pro svou jednoduchost je však nadále využívána (Myslín, 2016, str. 24). Základní myšlenka tohoto modelu vychází ze sekvenčního přístupu u jednotlivých fází. Jednotlivé fáze na sebe navazují postupně jedna za druhou. Celý proces má lineární postup a neopakuje se (Alshamrani &

Bahattab, 2015). Hlavní charakteristikou vodopádového modelu je to, že projekt může do další fáze vstoupit až tehdy, kdy je předchozí fáze kompletně dokončena a uzavřena. Jednou ze základních nevýhod, jak autor uvádí, je fakt, že v praxi, zejména u rozsáhlejších projektů, nelze prakticky dokončit jednu fázi a zahájit další, bez toho, aniž by se k ní v budoucnu vývojový tým opět vrátil, jelikož se můžou požadavky klientů v průběhu realizace projektu často měnit. V těchto případech by proto bylo nutné zapracovat změny jak do specifikace projektu, tak i do veškeré dokumentace (Testování Softwaru, n.d.).

Obrázek 1

Fáze Vodopádové Metody



Zdroj: (Myslín, 2016, str. 25), upraveno

Jak je na výše uvedeném diagramu patrné, jednotlivé fáze se na sebe postupně navazují jedna za druhou. Můžeme tedy tvrdit, že tento proces má lineární vývoj a neopakuje se. Následující seznam se snaží stručně popsat obsah těchto fází (Alshamrani & Bahattab, 2015):

1. **Požadavky:** V této fázi dochází ke stanovení toho, jaký by měl přesně být výsledek. Jelikož už není možné se k této fázi později vrátit, požadavky musejí

být specifikovány velice přesně a do jisté míry až „nadčasově“. Požadavky jsou poté nashromážděny, analyzovány a poté se připravuje dokumentace.

2. **Návrh:** V této fázi jsou domyšleny a rozpracovány úkony, které se povedou k převedení požadavků do výsledného softwaru. Jde o proces plánování a řešení problémů návrhu softwaru. Často se ovšem stává, že se při vývoji ukáže, že návrh je potřeba nějak pozměnit, což by přísně vodopádová metoda neměla umožňovat.
3. **Implementace:** V této fázi se implementují všechny stanovené požadavky.
4. **Testování:** Tato fáze se soustředí na testování a kontrolování vyvinutého softwaru, dále se řeší případné nalezené chyby, jejich následné opravování a vylepšování systému.
5. **Provoz:** Po finalizování realizace a řádného otestování je produkt předán zákazníkovi a následuje fáze, ve které se uvádí do provozu a zajišťuje se potřebná údržba vyvinutého softwaru. Tato fáze je jednoznačně z celého životního cyklu projektu nejdelší. V této fázi se také zavádějí úpravy na základě zákazníkovo požadavků. Ačkoliv fáze údržby nemusí být vždy nutnou součástí projektu, má-li produkt zůstat funkční, musí být pravidelně udržován (Dočkal, 2015; Sommerville, 2013).

Každá fáze vodopádové metody vývoje softwaru je dokončena, než se může přejít k následující fázi. Tento přístup může být užitečný pro velké a složité projekty, kde je třeba pečlivé plánování a řízení projektu. Nicméně, nese sebou řadu nevýhod:

1. **Nízká flexibilita:** Vodopádová metoda vývoje softwaru je velmi lineární a rigidní. To znamená, že jakmile začne každá fáze projektu, nelze ji zastavit nebo změnit směr. To může být problém, pokud se požadavky nebo potřeby uživatelů změní během procesu vývoje.
2. **Zpoždění objevení chyb:** Vodopádová metoda vývoje softwaru předpokládá, že všechny požadavky a specifikace jsou definovány před začátkem vývoje. To může vést k tomu, že některé chyby nebo nedostatky v návrhu nebo specifikacích nebudou objeveny, dokud se nedostanou do pozdějších fází projektu, což může být nákladné a zpozdit projekt.
3. **Vysoké náklady na změny:** Pokud jsou změny požadavků nebo specifikací nezbytné v pozdějších fázích projektu, mohou být velmi nákladné a časově náročné na implementaci. To může také způsobit zpoždění projektu.

4. **Malá možnost feedbacku od uživatelů:** Vodopádová metoda vývoje softwaru neumožňuje mnoho možností pro feedback od uživatelů v pozdějších fázích projektu. To může vést k tomu, že konečný produkt není v souladu s požadavky uživatelů (Singh, 2023).

2.1.2 Spirálová metoda

Spirálová metoda vývoje softwaru je iterativní model vývoje softwaru, který kombinuje prvky kaskádového modelu vývoje s principy prototypování a iterace. Tento model je vhodný pro vývoj sofistikovaných a komplexních softwarových systémů (Boehm, 1988).

Spirálová metoda vývoje softwaru se skládá z několika fází, které se opakují v každé iteraci. Tyto fáze jsou:

1. **Plánování**: V této fázi se identifikují cíle projektu, vytváří se plán a určuje se, jaké zdroje budou potřeba pro realizaci projektu.
2. **Analýza rizik**: V této fázi se identifikují potenciální rizika projektu a vyhodnocuje se, jaké jsou jejich důsledky a pravděpodobnost výskytu.
3. **Inženýrství**: V této fázi se vytváří prototypy a provádí se testování, aby se ověřila funkcionalita softwarového systému.
4. **Hodnocení**: V této fázi se zhodnocuje průběh projektu, vyhodnocuje se, zda jsou splněny cíle a jak se projekt vyvíjí (Boehm, 1988; JavaTpoint, n.d.).

Každá iterace zahrnuje plánování, analýzu rizik, inženýrství a hodnocení, a každá iterace se navrhuje tak, aby se využily poznatky a zkušenosti z předchozí iterace. Tento model umožňuje pružný přístup k vývoji softwarového systému a včasné odhalení problémů a jejich řešení (Boehm, 1988; JavaTpoint, n.d.).

I když má spirálová metoda několik výhod, existují také některé nevýhody, které by měly být vzaty v potaz:

1. **Náklady**: Vzhledem k opakovanému procesu v rámci každé iterace, může být spirálová metoda vývoje softwaru poměrně nákladná v porovnání s jinými metodami.
2. **Časová náročnost:** Kvůli opakovanému procesu v každé iteraci může být spirálová metoda vývoje softwaru časově náročná a může zpomalit tempo vývoje.

3. **Náročnost na řízení:** Pro správné provedení spirálové metody vývoje softwaru je potřeba silné a efektivní řízení projektu, které může být náročné na zdroje a zkušenosti.
4. **Není vhodná pro jednoduché projekty:** Spirálová metoda vývoje softwaru je navržena pro vývoj složitých a rozsáhlých softwarových systémů, takže není ideální pro menší a jednodušší projekty (JavaTpoint, n.d.).

2.2 Agilní přístup

V roce 2001 organizace Agile Alliance vytvořila manifest a prohlášení o principech, známé také jako „manifesto of Agile Alliance“. Agilní projektový management se řídí těmito principy a agilní postupy se odrážejí od těchto zásad

Agilní manifest vypadá takto:

„Odhalujeme lepší způsoby vývoje softwaru tím, že to sami praktikujeme a pomáháme praktikovat ostatním, aby to také dokázali. Skrze tuto práci jsme začali oceňovat a upřednostňovat:

- **Jednotlivce a interakce** před procesy a nástroji
- **Fungující software** před komplexní dokumentací
- **Spolupráce se zákazníkem** před vyjednáváním smlouvy
- **Reakce na změnu** před následováním plánu

To znamená, že zatímco položky napravo mají svoji hodnotu, přesto dáváme položkám nalevo hodnotu větší " (Agile Aliance, 2001).

První hodnotu lze chápat jako, že špatně dokumentovaný projekt s dobrými interakcemi je lepší, než projekt s dobrou dokumentací, ale nepřátelskými interakcemi mezi jednotlivci. Druhá hodnota vyjadřuje, že fungující kód je jediným viditelným důkazem o progresi projektu. Dokumentace je něco, co může sloužit k podpoře týmové práce. Třetí hodnota zvýrazňuje spojení mezi týmem a zákazníkem k provádění rozhodování. Smlouva může být potřebná k navázání spolupráce, ale poté už potřebná není. Čtvrtá hodnota může být vysvětlena tak, že každý projekt potřebuje plán, ale plánování v rámci agilního přístupu má krátký vývojový cyklus, aby bylo možné rychle reagovat na změny (Cockburn, 2006).

Dvanáct principů agilního přístupu prohlubuje význam těchto hodnot v praxi:

- Naší nejvyšší prioritou je včasné uspokojení zákazníka a nepřetržité dodávání cenného softwaru.
- Vítáme měnící se požadavky, a to i v pozdním vývoji. Agilní procesy využívají změny ke konkurenční výhodě zákazníka.
- Dodávejte pracovní software frekventovaně, od několika týdnů do několika měsíců, s upřednostněním kratšího časového horizontu.
- Obchodníci/Zákazníci a vývojáři musí denně spolupracovat během celém projektu.
- Vytvářejte projekty s pomocí motivovaných jednotlivců. Dejte jim prostředí a podporu, kterou potřebují, a důvěřujte jim, že práci dokončí.
- Nejúčinnější a nejefektivnější způsob předávání informací s vývojovým týmem a v rámci něj je komunikace z očí do očí.
- Funkční software je primárním měřítkem pokroku.
- Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři a uživatelé by měli být schopni udržovat konstantní tempo po dobu neurčitou.
- Neustálá pozornost věnovaná technické dokonalosti a dobrému designu zvyšuje obratnost.
- Jednoduchost – umění maximalizovat množství neudělané práce – je esenciální.
- Nejlepší architektury, požadavky a návrhy vycházejí ze samostatně se organizujících týmů.
- V pravidelných intervalech tým přemýšlí o tom, jak se stát více efektivní a poté podle toho vyladí a upraví svoje chování (Agile Aliance, 2001).

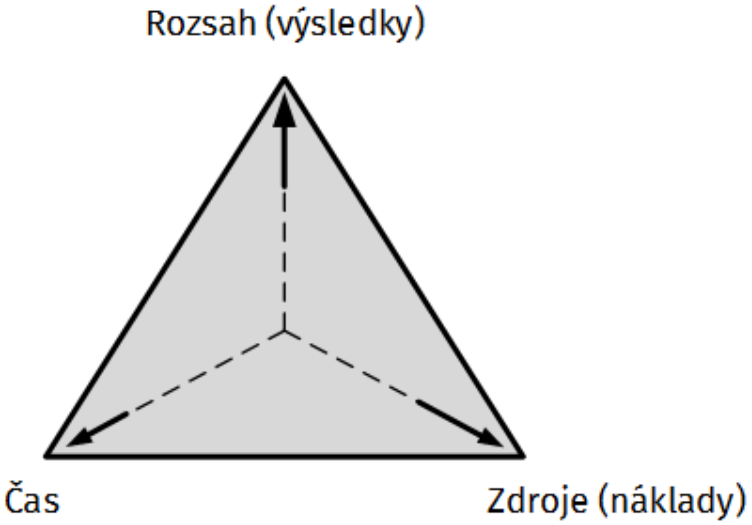
2.3 **Trojimperati**v

Trojimperativem projektového řízení můžeme rozumět jednoduchou reprezentaci klíčových prvků potřebných k úspěšnému plánování projektu. Tyto klíčové prvky jsou:

- **Rozsah** – určuje všechny práce v projektu. Zároveň určuje výsledek, který zákazník očekává
- **Čas** – definuje harmonogram projektu
- **Zdroje/Náklady –** za kolik se má co udělat (Doležal, 2016)

Obrázek 2

Trojimperativ projektu



Zdroj: (PM Consulting, n.d.)

Trojimperativ projektového řízení můžeme příhodně použít právě pro nalezení odpovědi při volbě metodiky pro náš konkrétní projekt. Metodiku Waterfall je nejvhodnější využít v takovém projektu, jehož rozsah je přesně definován, a je klíčovým prvkem projektu, tedy například plánování veletrhu, konference, stavbě domu či implementaci jednoduchého softwaru. S pevně daným rozsahem je hlavním úkolem projektového manažera naplánovat zdroje na časové ose s ohledem na požadovanou posloupnost úkolů. Agilní přístup je vhodný využívat v projektech, kde je pevně definovaný čas, a naopak rozsah je předmětem plánování. Ideálním příkladem je právě vývoj softwaru. Hlavní důraz se zde klade na správu priorit (Lizner, 2018).

3 Agilní metodiky

Agilní metodiky se již několik let rozvíjejí a stávají se nezbytným nástrojem pro efektivní a rychlou realizaci projektů. Tato kapitola se zaměřuje na klíčové agilní metody a frameworky, které zásadním způsobem formovaly současný i budoucí vývoj IT a technologických projektů. V průběhu této kapitoly se seznámíme s nejdůležitějšími agilními metodikami a rámci, jako jsou Extreme Programming (XP), Kanban, Crystal, a nakonec si podrobně rozebereme dnes nejpoužívanejší agilní metodu: Scrum.

3.1 Extrémní programování (XP)

Extrémní programování (XP) patří mezi agilní metodiky softwarového vývoje, které vznikly jako reakce na neefektivitu tradičních vodopádových metodik. Klade důraz na rychlou a flexibilní reakci na změny požadavků zákazníka a na maximalizaci hodnoty pro zákazníka. XP se skládá z několika praktik, jako například společné vlastnictví kódu, párové programování, testování v průběhu vývoje, jednoduchost návrhu a kontinuální integrace (Beck & Andres, 2004).

XP využívá iterativního a inkrementálního přístupu k vývoji softwaru a klade důraz na spolupráci a komunikaci mezi vývojáři a zákazníkem. Podle Becka (2004) je XP založeno na předpokladu, že změny v požadavcích zákazníka jsou přirozenou součástí vývoje softwaru. Důraz na kvalitu kódu a jeho čitelnost pomáhá minimalizovat technický dluh a usnadňuje údržbu softwaru v budoucnu.

Jednou z hlavních praktik XP je společné vlastnictví kódu, kdy všichni vývojáři mají odpovědnost za kód a podílí se na jeho vytváření a úpravách. Další klíčovou praktikou je testování v průběhu vývoje, což umožňuje včasnou identifikaci chyb a minimalizaci rizika vzniku technického dluhu (Beck & Andres, 2004).

Jednoduchost návrhu je další důležitou praktikou XP, která klade důraz na použití co nejjednodušších řešení, které splňují požadavky zákazníka (Beck, 2004). Párové programování je další praktikou XP, při které dva vývojáři spolupracují na řešení problému a sdílí si odpovědnost za výsledný kód (Beck & Andres, 2004).

3.2 Crystal

Crystal je model, který vytvořil člen Agile Alliance, Alistair Cockburn několik let před setkáním Agile Alliance. Nejedná se pouze o jednu metodiku, ale o rodinu metodik. První z nich, Crystal Clear, je vytvořena pro menší týmy o maximálně osmi lidech. Druhá, Crystal Yellow, se hodí spíše pro týmy mezi deseti a dvaceti. Třetí, Crystal Orange, se hodí pro týmy mezi dvaceti a padesáti lidmi.

Metoda Crystal se zaměřuje na lidi zapojené do projektu, přičemž hlavní pozornost je věnována členům týmu, komunitě, dovednostem, způsobům komunikace, talentům a vzájemným vztahům.

Jednou z nejvyšších priorit společnosti Crystal je časté doručování: Vývojáři Crystalu musí vydávat iterace často a podle plánu. Harmonogram se u jednotlivých týmů liší, typický časový rámec je jeden týden mezi iteracemi (Clark, 2019).

Křišťálová metoda funguje také v rámci reflexní dílny. Workshop se zaměřuje na to, aby pomohl týmu upravit jeho jednání, nikoliv chování. Tyto workshopy se konají každých několik týdnů a tým během nich identifikuje, co nefunguje a jak problém odstranit. Dalším krokem po workshopu je osmotická komunikace. Osmotická je vědecký termín, který je definován jako proces vstřebávání nebo difúze naznačující tok osmotického působení zejména: často nevědomé osvojování bez vynaložené námahy (Cockburn, 2004). Stručně řečeno, osmotická komunikace je náhodné přeslechnutí zákulisních informací, které se později mohou ukázat jako důležité. Během osmotické komunikační schůzky je cílem, aby přesné informace rychle proudily do celé skupiny pomocí rychlých otázek a aktuálních informací o všech aspektech projektu.

3.3 Scrum

Scrum (v čestině se často označuje jako „Skrumáž“ nebo „Mlýn“) je nejznámější metoda agilního programování. Scrum pochází z ragby, kde značí soustředění hráčů na jednom místě tak, aby dosáhly svého cíle, tedy získali a udrželi míč. V ideálním případě by se také při vývoji softwaru pomocí metody Scrum měli soustředit všichni zúčastnění na jednom místě.

Hlavní výhodou metody Scrum je flexibilní reakce na měnící se požadavky zákazníka a rychlé získání zpětné vazby. Vývoj projektu probíhá v sprintech (iteracích) - obvykle

několika týdenních intervalech nazývaných sprint, kdy se nejprve stanoví cíl a po uplynutí intervalu dochází k jeho vyhodnocení (Šochová & Kunce, 2019).

Ken Schwaber tuto metodiku ve svém počátku popsal v roce 2004 značně zábavným způsobem: Osoby, které se účastní vývoje projektu můžeme rozdělit na tzv. Pigs a tzv. Chickens. Toto na první pohled nečekané rozdělení vzniklo z povídky o praseti a kuřeti. Kuře si chtělo otevřít restauraci „Ham n eggs“, ale prase to odmítlo s tím, že prase by bylo přímo součástí podnikání, kdežto kuřete by se pouze „týkalo“. V případě scrumu se mezi prasata řadí osoby vykonávající vývoj softwaru, tedy jsou jeho přímou součástí. Mezi kuřata pak řadíme osoby, které se přímo na vývoji nepodílí (Knesl, 2009).

Mezi prasata patří „Product owner“ a „Scrum master“ se svým týmem. Do kuřat patří „Stakeholders“ a „Managers“.

3.3.1 Pojmy v metodice Scrum

Scrum Master

V metodice Scrum se využívá několika klíčových označení rolí. Jednou z nich je tzv. Scrum master – osoba, která zodpovídá za řádný průběh celého procesu a úspěch projektu. Na tuto roli se nejvíce hodí projektový manažer nebo týmový koordinátor, a nejčastěji tuto roli také přebírá, ale kdybychom se striktně drželi agilního přístupu, tak by si tým sám demokraticky zvolil osobu, která pozici Scrum mastera bude zastávat. Úkolem osoby v této roli je programátory odstínit od okolního světa, řídit je a starat se o to, aby jim fungovaly počítače a technika, měli dostupný potřebný software a řešit případné spory a nedorozumění v týmu. Scrum master nesmí být zároveň i aktivním programátorem v projektu, v takovém případě by nebyl schopen programátory odfiltrovat od rušivých elementů, byl by sám v krizových situacích zranitelný a zaneprázdněný programováním, namísto hledání optimálního řešení dané situace. Scrum master by si tedy měl vždy udržovat jistý nadhled (Šochová & Kunce, 2019).

Product owner

Product Owner je zákazník, který rozhoduje o tom, jaké funkcionality má produkt mít, určuje jejich prioritu a sleduje business value a ROI produktu (Šochová & Kunce, 2019). Rozhoduje o obsahu jednotlivých vydání daného softwaru, datech, přijímá nebo odmítá

výslednou práci a pravidelně alteruje seznam vlastností výsledného produktu (Šochová & Kunce, 2019).

Stakeholders

Zainteresované strany – jednotlivci, týmy nebo organizace, které mají zájem na systému, nebo jsou ve vztahu k systému. V tomto případě jsou to především lidé ze strany zákazníka, např manažer podniku, uživatelé, ale také třeba testeři (Bruckner, 2012).

Managers

V počátcích vývoje metodiky Scrum byla role manažerů podle Kena Schwabera popisována jako klíčová pro úspěšné dokončení projektu. Jejich úkolem bylo vytvořit prostředí, které by podporovalo využití Scrumu, avšak nebyli součástí týmu a neměli specifické úkoly v rámci této metodiky (Schwaber, 2004).

Schwaber (2004) dále uvádí, že manažeři v rámci Scrumu nejsou součástí týmu a jejich hlavní rolí je podpora využití Scrumu a zajištění potřebných zdrojů pro projektové plnění. Manažeři nemají specifické role nebo zodpovědnosti v rámci Scrumu.

V dnešní době oficiální verze Scrumu, kterou publikují organizace Scrum.org a Scrum Alliance, již nezahrnuje roli manažerů. To znamená, že v současné verzi Scrumu nejsou manažeři oficiálně definovanou rolí a jejich úkoly jsou rozděleny mezi Product Ownera, Scrum Mastera a tým (Scrum.org, n.d.).

Sprint

Sprint je klíčovou časovou jednotkou v metodice Scrum a představuje krátký a časově omezený vývojový cyklus, který má pevně stanovenou délku trvání (typicky 1 až 4 týdny). Na konci každého sprintu se od týmu očekává hotová část softwaru, která by mohla být uvedena do ostrého provozu, pokud si to zákazník přeje (Schwaber & Sutherland, 2017). Doba trvání sprintu není pevně stanovena a může se lišit v závislosti na specifických potřebách týmu a projektu. Obecně se však doporučuje, aby trvání jednoho sprintu nepřesáhlo jeden měsíc. Pokud se vývoj některé části produktu odhaduje na delší dobu, je nezbytné ji rozdělit na menší dílčí úkoly, což pomáhá minimalizovat rizika a zlepšit přesnost odhadů (Doležal, 2016; Scrum.org, n.d.).

Je důležité dodržovat pravidelnost v délce trvání sprintů, protože to pomáhá týmu lépe plánovat a odhadovat svou práci. Tým, který pracuje pravidelně a v rytmu sprintů, může lépe určit, co je možné v rámci sprintu dokončit a co ne. To zlepšuje průběžné řízení projektu a umožňuje týmu přizpůsobovat se měnícím požadavkům a potřebám zákazníka (Schwaber & Sutherland, The Scrum Guide, 2017).

Product Backlog, Sprint Backlog

Podstatou úspěšného sprintu v metodice Scrum je jeho kvalitní naplánování. K tomu slouží tzv. Sprint Backlog, což je seznam úkolů a činností, které je třeba vykonat pro dokončení daného sprintu. Proces tvorby Sprint Backlogu začíná převedením všech požadavků na výsledný produkt a jejich prioritizací ze seznamu Product Backlogu. Tyto požadavky jsou poté pečlivě analyzovány a zahrnuty do Sprint Backlogu tak, aby byla zachována logická návaznost (Šochová & Kunce, 2019).

Sprint planning

Plánování sprintu je kolektivním úkolem celého Scrum týmu, který úzce spolupracuje a definuje práci, která bude vykonána v rámci následujícího sprintu a stanovuje cíl sprintu. Doba trvání této schůzky je pevně stanovena na maximálně 8 hodin, aby byla zajištěna efektivita a produktivita práce .

Na schůzce plánování sprintu vlastník produktu (Product Owner) představuje nejvyšší prioritou funkce produktového backlogu. Scrum tým následně diskutuje o možnostech, jak tyto funkce co nejlépe dosáhnout během nadcházejícího sprintu a rozděluje je na menší úkoly. Cílem tohoto procesu je vytvořit efektivní plán pro nadcházející sprint, který je dobře realizovatelný a současně plní požadavky a cíle projektu. (Schwaber & Sutherland, The Scrum Guide, 2017; Rubin, 2013)

Sprint goal

Cíl sprintu je jedním z nejdůležitějších prvků Scrumu a představuje malou vizi pro daný sprint. Během trvání sprintu by tento cíl neměl být nijak měněn, ale je možné upravit Sprint Backlog tak, aby bylo dosaženo cíle sprintu. Nicméně, jakékoli změny, které by mohly ohrozit cíl sprintu, by měly být zabráněny (Šochová & Kunce, 2019).

Sprint Goal slouží k tomu, aby tým měl jasnou představu o tom, na čem se bude během sprintu pracovat a jakým způsobem se bude postupovat. Cíl sprintu umožňuje týmu zaměřit se na klíčové úkoly a minimalizuje riziko zaměření na nesprávné priority.

Je důležité, aby Sprint Goal byl průběžně komunikován v rámci týmu a aby si každý člen týmu uvědomoval, jakým způsobem jeho práce přispívá k dosažení tohoto cíle. Zároveň je třeba, aby cíl sprintu byl realistický a dosažitelný, aby tým mohl být úspěšný při plnění svých úkolů a cílů (Scrum.org, n.d.)

Increment

Přírůstek je součet všech dokončených částí ze současného Sprintu a ze všech minulých Sprintů, které jsou zároveň i zapsány v Product Backlogu. Můžeme ho označit jako konečný produkt, který lze představit zákazníkovi (Šochová & Kunce, 2019).

Daily Stand-up

Také označován jako Scrum meeting je každodenní, zhruba deseti minutová schůzka (v ranních hodinách), při které se zjistí stav sprintu. Stand-upu se účastní projektový tým, Product Owner a Scrum master. Každý člen vývojového týmu popíše, jakou za předchozí den dokončil práci, jestli nastaly nějaké problémy a jakou práci dokončí v daný den (Doležal, 2016; Šochová & Kunce, 2019).

Sprint Review

Na konci každého vývojového cyklu, zvaného sprint, se uskutečňuje uzavírací schůze, označovaná jako Sprint Review, jejímž primárním cílem je získání zpětné vazby z právě ukončeného sprintu a identifikace zkušeností pro další iterace. Tato schůze je realizována za účelem ověření přírůstku (incrementu) a efektivního zpracování části produktu v souladu s produktovým backlogem.

Stěžejním prvkem Sprint Review je prezentace dokončeného meziproduktu zákazníkovi, během které je demonstrována dosažená funkcionalita. Product Owner má na starosti rozhodování o akceptaci jednotlivých položek, čímž tým získává cennou zpětnou vazbu pro další vývoj. Délka trvání této schůze by neměla přesáhnout 4 hodiny a jejími účastníky by měli být: Product Owner, projektový tým, Scrum Master a stakeholdeři (Doležal, 2016; Šochová & Kunce, 2019).

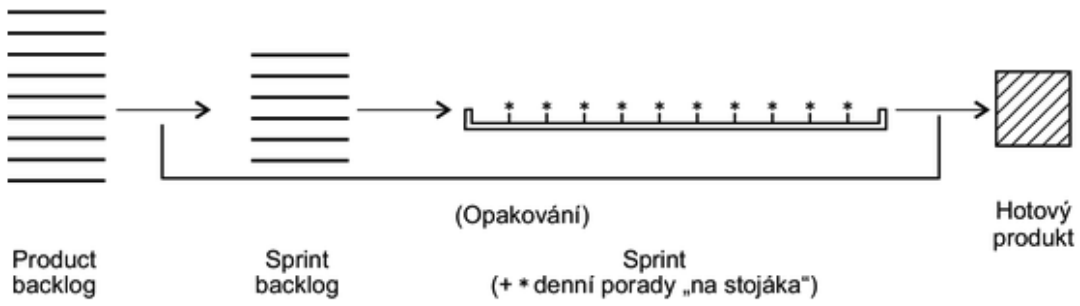
Sprint Retrospective

Po dokončení Sprint Review se celý Scrum tým účastní Retrospektivy sprintu, která slouží k reflektování práce vykonané během sprintu a identifikaci oblastí, které lze vylepšit. Cílem této akce je plánovat pozitivní změny, které mohou být implementovány v příštím sprintu. Retrospektiva se koná před dalším plánováním sprintu (Sprint planning) a může trvat maximálně tři hodiny.

Retrospektiva sprintu umožňuje týmu reflektovat na výkony během sprintu a identifikovat oblasti, které by mohly být zlepšeny. Tým spolupracuje na plánování pozitivních změn, které mohou být implementovány v dalším sprintu. Důležité je, aby byli všichni členové týmu aktivně zapojeni a podělili se o své názory, aby se dosáhlo co největšího úspěchu při vývoji produktu (Schwaber & Sutherland, The Scrum Guide, 2017; Rubin, 2013).

Obrázek 3

Schéma sprintu



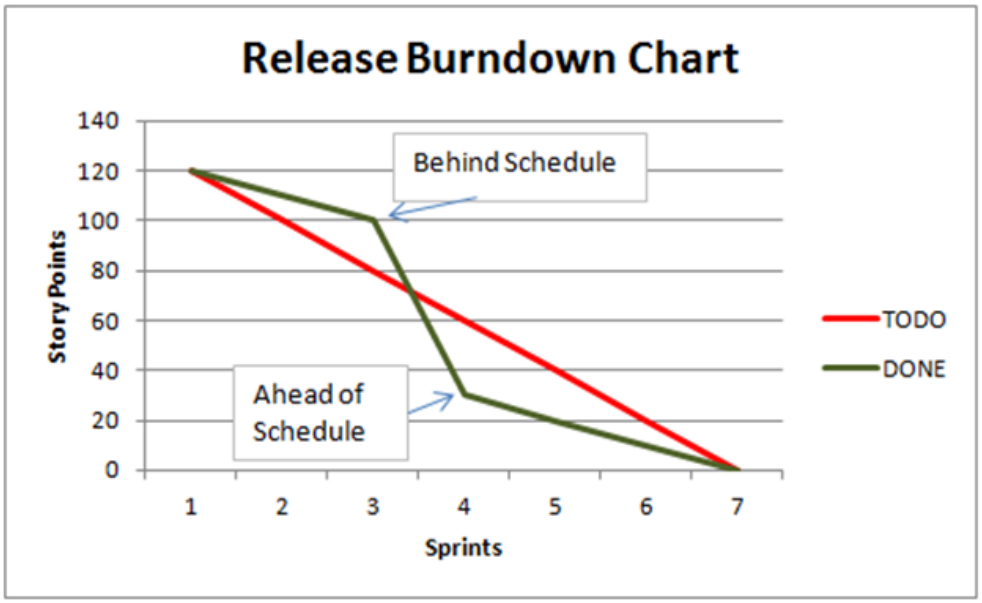
Zdroj: (Doležal, 2016)

Burndown Chart

Je graf používaný při agilním řízení projektu, který ukazuje množství zbývající práce pro daný sprint. Je to v podstatě součet času, který zbývá ze sprint backlogu po tom, co odečteme už hotovou práci. Je tím vizualizováno tempo práce týmu. Burndown graf by měl dostávat aktualizaci na denní bázi a na konci každého sprintu by měl dosahovat nuly (World of Agile, 2016).

Obrázek 4

Příklad Burndown Chart



Zdroj: (World of Agile, 2016)

User story

Agilní metody využívají k popisu budoucí funkcionality pojem User Story, základní podobu je: „Jako uživatel chci Funkcionalitu, abych dostal Business Value“ (Šochová & Kunce, 2019, str. 77). Jedná se o krátkou větu, která popisuje požadovanou funkcionalitu z hlediska uživatele, například: "Jako uživatel chci mít možnost změnit své heslo, abych si zlepšil bezpečnost svého účtu." Jak autoři dále uvádějí, každá User Story by měla dodržovat několik pravidel, které se označují zkratkou **INVEST:**

- **I**ndependent – nezávislá
- **N**egotiable – popsatelná – Každá User Story by měla být snadno pochopitelná a vysvětlitelná
- **V**aluable – hodnotná – Každá User Story by měla pro konečného uživatele přinášet hodnotu
- **E**stimable – zhodnotitelná – Členové týmu by si měli být schopni vytvořit o User Story obrázek a zhodnotit ji
- **S**mall – malá – User Story by měla být dostatečně malá na to, aby mohla být zhotovena v polovině každého sprintu, jinak musí být dělena na menší dílčí části

- **Testable –** testovatelná – Každá User Story musí mít jasně definovaná kritéria, dle kterých se vyhodnotí, jestli byla dokončená (Šochová & Kunce, 2019)

Scrum board

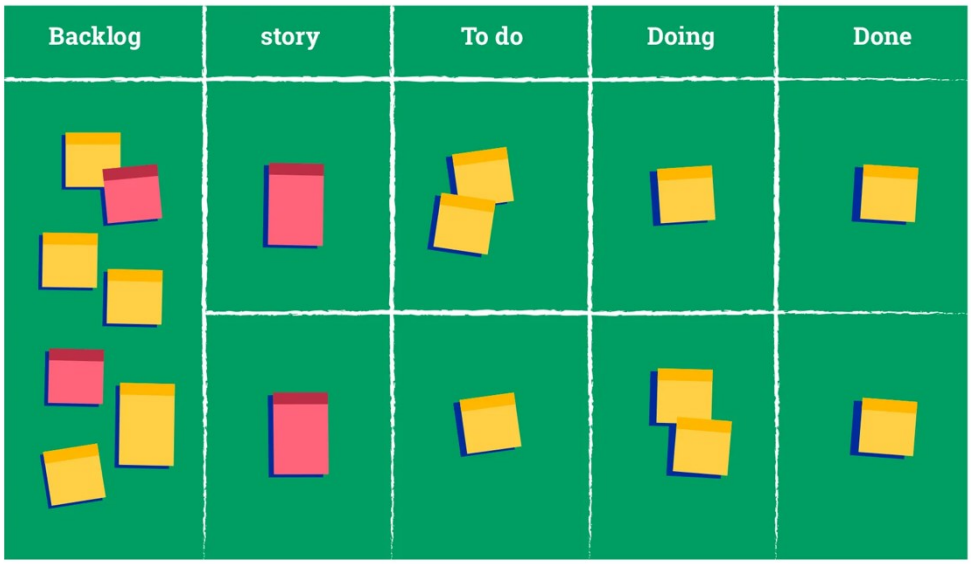
Jednotlivé uživatelské příběhy (user stories) se dále rozdělují do jednotlivých úkolů. Ty jsou ale příliš podrobné na to, aby se zapisovaly do Sprint/Product backlogu. Vývojový tým je naopak potřebuje mít neustále před sebou, aby měl možnost vizualizace průběhu prací. Nejjednodušší formu Scrum boardu znázorňuje Obrázek 5. Na jednotlivé user stories se vážou dílčí úkoly, kterých se obvykle některý člen týmu dobrovolně ujme na daily stand-upu, a osobně jej přesune do pole „Doing“. Jelikož se stand-upy konají každých dvacet čtyři hodin, měly by být dílčí úkoly navrženy tak, aby se stihly dokončit za jeden pracovní den.

Očekává se tedy, že další ráno člen týmu přesune své úkoly z pole „Doing“ do pole „Done“; pokud tak neudělá, dá se očekávat, že nastala nějaká překážka v práci a ta se následně řeší (Zoho.com, n.d.; Doležal, 2016).

K tvorbě scrum boardů se obvykle využívá dedikovaný software, ke kterému má přístup celý vývojový tým přímo z internetového prohlížeče. Mezi používané softwary patří např: Zoho Sprints, Jira, Wrike a Asana (York, 2023).

Obrázek 5

Příklad Scrum board



Zdroj: (Zoho.com, n.d.)

3.3.2 Scrum Postupy

Jak uvádějí Schwaber a Sutherland (2017), každý projekt by se dal v metodě Scrum stručně rozdělit do třech hlavních fází:

1. Předehra **(Pre-game):** Tato fáze se zabývá plánováním a architekturou projektu. Vytváří se Product Backlog a seznam úkolů pro rozdělení eposů do menších uživatelských příběhů (User Story). Uživatelské příběhy se přemění na jednotlivé úkoly ve fázi tvorby Sprint Backlogu. V této fázi se také provádí analýza rizik, volba nástrojů, doménová analýza a volba systémové architektury. Na konci první fáze získá tým představu o hrubém plánu projektu a může přiřadit jeho priority.
2. Hra **(Game):** Druhá fáze se soustředí na skutečný vývoj produktu, který probíhá v iterativních sprints. V průběhu každého sprintu se pracuje na položkách ze Sprint Backlogu, které mají nejvyšší prioritu. Během této fáze se zapisují nové položky do Product Backlogu a vytvářejí se burndown grafy, které ukazují pokrok týmu v průběhu sprintu.
3. Po hře **(Post-game):** Třetí fáze se zaměřuje na dokončení produktu a jeho připravenost k uvedení na trh. V této fázi se provádějí všechny potřebné kroky, jako je testování, integrace, školení a vytváření uživatelské dokumentace. Po dokončení všech těchto kroků je produkt připraven k vydání.

3.4 Kanban pro soběstačné týmy

Kanban je vedle SCRUM jednou z nejznámějších agilních metod. Pochází z Japonska a ve 40. letech 20. století ji vyvinul Taiichi Ohno. Byl navržen jako jednoduchý plánovací systém pro kontrolu a optimální řízení práce a zásob v každé fázi produkce. Slovo Kanban znamená v japonštině něco jako „karta, která je vidět.“ Zatímco Kanban vznikl ve výrobním průmyslu, David J. Anderson tento koncept poprvé aplikoval v roce 2004 na IT, vývoj softwaru a práci se znalostmi obecně (NimbleWork, n.d.).

Metoda Kanban se řídí souborem zásad a postupů pro řízení pracovního postupu. Dodržováním těchto postupů a zásad metoda Kanban úspěšně umožní maximalizovat přínosy pro podnikové procesy, zkrátit dobu cyklu a zvýšit hodnotu pro zákazníka díky větší předvídatelnosti (ProductPlan, 2022).

Existují čtyři základní principy, z nichž první je "Začněte s tím, co děláte teď" (ProductPlan, 2022). Kanban klade důraz na to, aby se stávající procesy neměnily hned. Obratnost v Kanbanu umožňuje týmu pomalu zavádět nové metody a postupy v závislosti na požadavcích projektu, aniž by došlo ke kulturnímu šoku z nových technologií nebo jiných procesů (Kanbanize, n.d.).

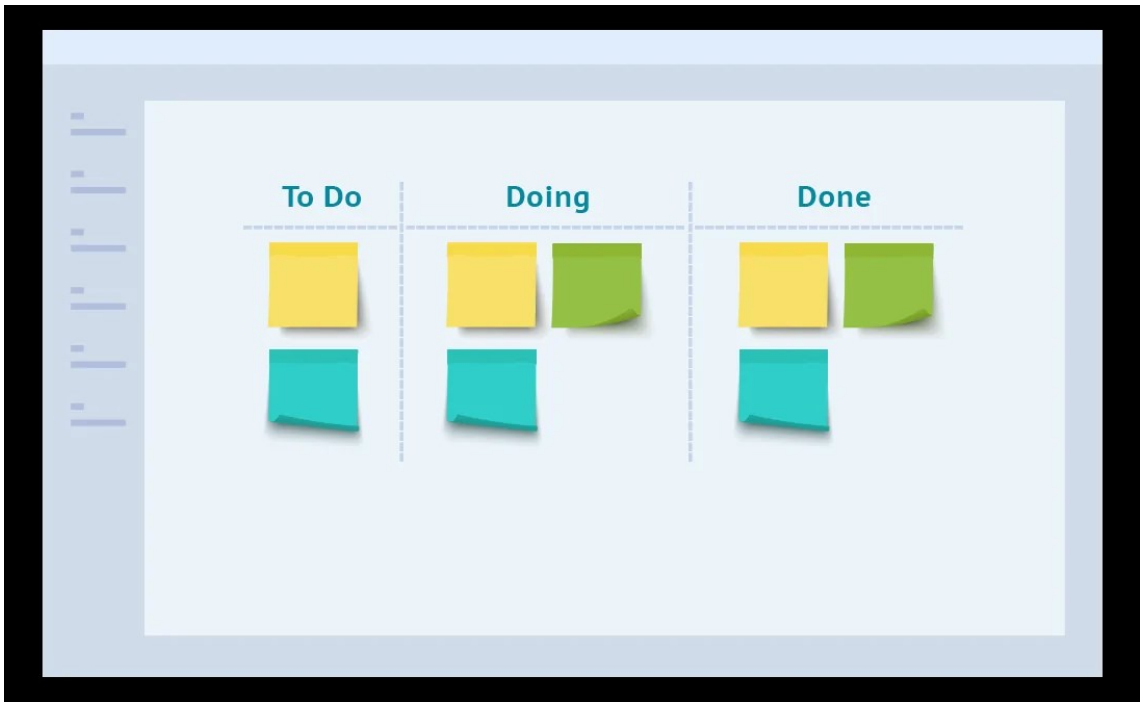
Druhým principem je "Souhlasit s postupnou evoluční změnou" (Kanbanize, n.d.). Tato zásada vybízí k provádění menších postupných změn, protože pokud by se místo toho prováděly změny radikální, setkaly by se v týmu a organizaci s odporem kvůli strachu nebo nejistotě (NimbleWork, n.d.). Třetím principem je "Zpočátku respektujte stávající role, odpovědnosti a pracovní tituly" (Kanbanize, n.d.). Kanban respektuje stávající hierarchii v organizaci. Pokud tedy existují role nebo funkce, které dobře fungují, není nutné je měnit. Tým bude muset prodiskutovat nutnost nových změn a v případě potřeby je pomalu zavádět. Tyto tři první principy byly navrženy tak, aby podporovaly a povzbuzovaly postupné, logické změny a nevyvolávaly v organizacích strach ze změn. Čtvrtým principem je "Podporujte vůdčí činy na všech úrovních" (Kanbanize, n.d.). Tato zásada vybízí každého zaměstnance na všech úrovních, aby poskytoval nápady a projevoval vůdčí schopnosti při zavádění změn, které povedou k neustálému zlepšování způsobu poskytování produktů a služeb.

Kanban má šest základních procesů nebo metod. Prvním z nich je "Vizualizace pracovního postupu", kterou lze vidět na Obrázku 5, tabuli Kanban. Některé týmy používají jako nástěnku Kanban aplikaci Trello. Tato nástěnka má pouze tři sloupce, kterými jsou: To Do, Doing a Done. Úkoly se zapisují na karty Kanban, přičemž každá karta představuje jeden úkol. Úkol by měl být jednoduchý, nesmí se vytvářet více úkolů pro jednu kartu. To znamená, že na kartách by měl být buď návrh, nebo dokumentace, nikoli "návrh a dokumentace". Při zahájení úkolu se přesune z požadovaného na probíhající, což usnadňuje sledování pracovního postupu a odhalování úzkých míst a pomalu postupujících úkolů (Kanbanize, n.d.).

Druhým procesem je "Limit rozpracované práce", zohledňuje, kolik úkolů může být "rozpracováno" najednou, obvykle by se jednalo o šest až sedm úkolů (Clark, 2019). Myšlenka za tím je, že hned vidíte, kterému úkolu je třeba věnovat okamžitou pozornost, protože brzdí rychlost vývoje.

Obrázek 6

Tabulka Kanban se třemi sloupci



Zdroj: (Kanbanize, n.d.)

"Řízení toku" je třetím procesem, po zavedení dalších dvou postupů je nutné řídit a zlepšovat tok práce. Řídit znamená řídit práci, ale ne lidi, a tok je pohyb pracovních položek v celém procesu. Jedním z důvodů pro zavedení systému Kanban je vytvoření plynulého a zdravého pracovního toku. Namísto mikromanagementu lidí je třeba se zaměřit na pracovní procesy a pochopení, jak práci urychlit. Tímto způsobem by Kanban rychleji vytvářel hodnoty (Ahmad et al., 2013).

Čtvrtý proces "Zpřístupnění procesních zásad" usiluje o vytvoření pracovních pokynů, které vytvoří společný základ pro všechny účastníky, aby pochopili, jak pracovat v systému Kanban (Ahmad et al., 2013). Není možné zlepšovat něco, čemu se nerozumí. Proto jsou směrnice zásadní a otevírají také prostor, kde mohou účastníci tyto směrnice nebo metody práce vylepšovat, jakmile se všichni seznámí se společným cílem v projektu (Kanbanize, n.d.).

Pro týmy a organizace, které chtějí být agilnější, je pátý proces "Implementace smyček zpětné vazby" povinný. Získávání zpětné vazby od zúčastněných stran, zákazníků a uživatelů v rámci tzv. v raných fázích projektu umožňuje týmům reagovat na případné změny dříve, než začnou vyvíjet software nesprávným směrem. To v konečném důsledku umožní dodat zákazníkovi správný produkt nebo službu v co nejkratším čase (Ahmad et al., 2013).

Posledním, šestým procesem je "Zlepšování ve spolupráci, experimentální vývoj (s využitím vědecké metody)". Týká se zlepšování metod Kanban tak, aby odpovídaly potřebám týmu. Týmy, které společně chápou cíle, pracovní postupy, procesy a rizika, budou s větší pravděpodobností spolupracovat na zlepšování (Ahmad et al., 2013; Kanbanize, n.d.).

Nejjednodušší způsob, jak pochopit Kanban, je přečíst si a pochopit čtyři základní principy a aplikovat je na každodenní práci týmu. Když je budete aplikovat na každodenní práci týmu a nějakou dobu pracovat se šesti postupy, tým si všimne, jak moc Kanban ovlivňuje jeho práci.

Využitím těchto čtyř základních principů a šesti procesů metoda Kanban přináší maximalizaci přínosů pro podnikové procesy, zkracuje dobu cyklu a zvyšuje hodnotu pro zákazníka díky větší předvídatelnosti. Hlavní zaznamenané výhody použití Kanban metody byly zlepšení času dodání softwaru, zlepšení kvality softwaru, zlepšení komunikace a koordinace, zvýšení konzistence doručování a snížení počtu chyb hlášených zákazníkem. Tímto způsobem se Kanban stává důležitým nástrojem pro efektivní řízení pracovního postupu a neustálé zlepšování (Ahmad et al., 2013).

V praxi týmy často kombinují metodu Kanban s dalšími agilními metodami, jako je Scrum nebo Lean, aby dosáhly ještě větší efektivity a úspěchu v projektech. Tato kombinace umožňuje týmům využít silných stránek jednotlivých metod a přizpůsobit je svým specifickým potřebám a situacím

Kanban je také široce používán mimo tradiční oblasti softwarového vývoje. Organizace z různých odvětví, jako je zdravotnictví, výroba, vzdělávání nebo marketing, začínají uznávat přínosy, které metoda Kanban přináší, a implementují ji ve svých pracovních procesech.

Důležitým aspektem úspěchu metodologie Kanban je kultura spolupráce, otevřenosti a důrazu na neustálé zlepšování. Týmy, které adoptují Kanban, by měly být připraveny sdílet své zkušenosti, učit se jeden od druhého a podporovat se navzájem v hledání nejlepších postupů a inovací.

3.5 Scrumban, hybridní agilní metoda

Scrumban, neboli Scrum plus Kanban je kombinovaná agilní metoda, která kombinuje prediktivnost Scrumu s flexibilitou a kontinuálním pracovním postupem Kanbanu. Týmy, které tuto metodu používají, si vybírají postupy jak ze Scrumu, tak z Kanbanu, aby naplnily nezbytné požadavky na agilitu projektů (ProductPlan, n.d.). Tato metoda se používá hlavně pro probíhající projekty, pro týmy, které považují Scrum za příliš přísnou metodu a potřebují flexibilitu Kanbanu (Keup, 2020). Část Kanban ve Scrumbanu slouží ke zlepšení procesů, vizualizaci pracovního postupu, omezení počtu rozpracovaných položek na Scrum nebo Kanban tabuli, podle toho, kterou se tým rozhodne používat, nebo zda chce tabule kombinovat. Tato metoda je stále značně nová a zatím nemá žádný konkrétní soubor postupů. Tým rozhoduje o postupech, které považuje za nezbytné pro udržení agilního týmu (NimbleWork, n.d.).

4 Použití Scrumbanu v probíhajícím projektu

V této kapitole se budu věnovat jednotlivým krokům první implementace agilní metodiky Scrumban u vybraného podnikatelského subjektu, který vyvíjel internetový obchod (e-shop) a na něj napojenou webovou aplikaci pro prodej dluhopisů. Cílem této části je ukázat, jak byly principy, které byly představené v teoretické části, použity při řízení vývoje softwaru.

4.1 Představení vybraného podnikatelského subjektu

S ohledem na majitele a zákazníka, kteří si přáli zůstat v anonymitě, obsahuje tato část pouze základní informace. Společnost se v době nasazování metody do praxe skládala z 5 lidí. Tento projekt byl první zakázkou tohoto nově vzniklého týmu a zároveň první zkušeností s agilním vývojem. Cílem bylo vytvořit funkční a uživatelsky přívětivý e-shop a webovou aplikaci pro prodej dluhopisů daného e-commerce projektu, které by vyhovovaly požadavkům zákazníků a poskytovaly jim příjemný a bezpečný nákupní zážitek.

4.2 Využité nástroje při vývoji a implementaci

Součástí cíle této práce bylo přezkoumat nástroje použité při vývoji software z pohledu projektového řízení, vyhodnotit různé nástroje pro budoucí projekty a najít nejlepší praktiky.

4.2.1 Nástroj pro řízení projektů Trello

Trello je nástroj pro správu projektů vytvořený společností Atlassian. Členové týmu byli zvyklí na využívání nástroje Trello pro správu projektů a úkolů v rámci jejich předešlých zkušeností. Nástroj umožňuje používat šablony pro agilní rámce vývoje softwaru a zadávání úkolů vývojových činností. Trello funguje na bázi karet, které reprezentují úkoly a mohou být přiřazeny libovolnému účastníkovi tabule (board). Úkoly lze komentovat a přesouvat mezi úkolovými seznamy Avšak Trello nepodporuje vzájemné vztahy nebo hierarchie mezi úkoly a kartami, ani výpočty pracovního úsilí, což omezuje jeho efektivitu pro sledování pracovní zátěže (Atlassian, n.d.-b). Tým využíval Trello v prvních dvou týdnech vývoje, ale kvůli nedostatečnému množství funkcí a efektivitě práce se rozhodl přejít na nástroj Jira.

4.2.2 Nástroj pro řízení projektů Jira

Jira je pokročilý nástroj pro správu projektů vytvořený společností Atlassian, který nabízí mnohem více funkcí než Trello. Oba nástroje lze integrovat, nicméně Jira podporuje širší sadu integrací, včetně přímé integrace s platformou pro správu verzí (GIT). Jira také obsahuje nativní reporty pro pracovní zátěž, postup práce na projektu, sprinty, uživatelské příběhy a hierarchie a jejich vzájemné vztahy. Jira je v odvětví široce využíván, ale jeho složitost vyžaduje speciální nastavení a noví vývojáři s ním zpravidla nejsou obeznámeni, neboť nenabízí bezplatnou licenci jako Trello (Atlassian, n.d.).

4.2.3 GIT a GitHub

Git je distribuovaný systém pro správu revizí, který je k dispozici na všech běžných vývojových platformách pod svobodnou licencí. Systém GIT se zaměřuje na revize softwaru jako hlavní prioritu, což je důležitý rozdíl oproti starším systémům. GIT umožňuje každému vývojáři mít kompletní soukromou kopii softwarového úložiště a nabízí řadu způsobů, jak spravovat revize v jeho rámci. Díky možnosti propojení místního úložiště s mnoha vzdálenými úložišti mohou vývojáři a jejich manažeři vytvářet řadu zajímavých distribuovaných pracovních postupů, které by nebyly možné v tradičním centralizovaném systému správy verzí. GIT je také rychlý, snadno se nastavuje a umožňuje práci i bez připojení k internetu. GitHub je poskytovatel hostingu repositáře GIT, který zjednodušuje mnoho úkolů správy repositáře prostřednictvím webového uživatelského rozhraní a zároveň podporuje spolupráci v open-source projektech (Spinellis, 2012).

4.2.4 Discord

Discord je komunitní/týmová vícekanálová komunikační platforma, která nabízí funkcemi, jako jsou hlasové hovory, videohovory a textové zprávy. Původně byl Discord navržen pro herní komunity, ale nyní se jeho užitečnost rozšiřuje i do oblasti vzdělávání a průmyslového sektoru. Discord je dostupný na většině platforem (Windows, macOS, Android, iOS, Linux a webové prohlížeče) (Filpal, 2022).

Tým využíval Discord jako hlavní nástroj pro komunikaci během vývoje projektu. Discord umožňoval rychlou a efektivní komunikaci mezi členy týmu, bez ohledu na jejich geografickou polohu. Tým vytvořil kanály pro různá témata, jako například

"otázky ohledně úkolů", "pull requests", který byl kanál, kde se automaticky sdílely informace z revizního systému Git, a "testování", což umožňovalo organizovanou diskuzi a snadný přístup k potřebným informacím. Využívali také možnosti sdílení obrazovky (screen sharing) pro prezentaci práce ostatním členům týmu. Discord tak přispěl k rychlejšímu a efektivnějšímu vývoji projektu. Díky možnosti hlasových a video kanálů zde také probíhaly denní stand-upy (Filpal, 2022).

4.3 Plánování a implementace Scrumbanu

Vývojový tým pracoval na svém prvním společném projektu – internetový obchod (e-shop) a na něj napojenou webovou aplikaci pro prodej dluhopisů. Tým vyhledal existující agilní metody a vybral jednu z nich k implementaci. Nejprve byla implementována metoda umožňující flexibilnější styl práce a rychlejší zahájení projektu, Kanban. Je důležité mít nějaké konkrétní pokyny/směrnice, které tým může použít na začátku projektu, a proto týmu přišla metoda Kanban jako ideální volba.

Dalším krokem bylo vytvoření produktového backlogu, úkolů a označení jednotlivých úkolů (například „Frontend“ nebo „Backend“) pro to, co je třeba udělat, aby se projekt rozběhl, a následné rozdělení jednotlivých úkolů každému členovi, aby na nich mohli začít pracovat. V případě, že by byl některý úkol příliš časově náročný, rozdělil by se na menší dílčí úkoly. Plánování zahrnovalo nové možné funkce, vylepšení stávajících funkcí a požadavky na vydání produktu.

Tým vytvořil digitalizovanou nástěnku v nástroji Jira pro produktový backlog, kde byly všechny nově vytvořené úkoly umístěny do sloupce s názvem „Product Backlog“. Brzy po přidání úkolů byly přidány další sloupce, a to „To Do“, „In Progress“, „Ready to Test“ , „Review“ a „Done“.

Ve sloupci "To Do" byly všechny úkoly, které byly prioritami na nějakou dobu dopředu, důvodem bylo, že tým nebyl schopen předvídat, kolik času jednotlivé úkoly zaberou. Ve sloupci „In Progress“ byly pouze úkoly, na kterých každý člen pracoval. Bylo dohodnuto, že na jednoho člověka připadne pouze jeden úkol, s výjimkou případů, kdy na sebe úkoly navazovaly nebo existovaly úkoly, které musely mít přednost. Dokončené funkce produktu podle uvážení člena, který na něm pracuje, by se přesunuly do sloupce „Ready to Test“. Tyto funkce bylo třeba otestovat před přidáním do produkční verze projektu.

Úkoly, které byly otestovány nebo které nebyly funkcemi produktu, bylo možné přesunout do sloupce „Done“, což znamenalo, že úkol byl splněn. Další úkol poté mohl být přesunut do pracovní fáze „In Progress.“

V době, kdy jsem se připojil do projektu s cílem napsání této práce, byl projekt už v pokročilé fázi vývoje. Při povšimnutí si, že má tým problém s udržováním backlogu jsem si vzpomněl, že jsem narazil během svojí rešerše k teoretické části bakalářské práce na metodu Scrumban a rozhodl se o ní shromáždit základní informace. Doporučil jsem tedy týmu zkusit tuto metodu implementovat a vzít si scrumové nástroje, které projektu dodají větší předvídatelnost. Těmito nástroji Scrumu byly týmové role, sprinty, denní scrums (daily stand-up), sprint review a retrospektiva.

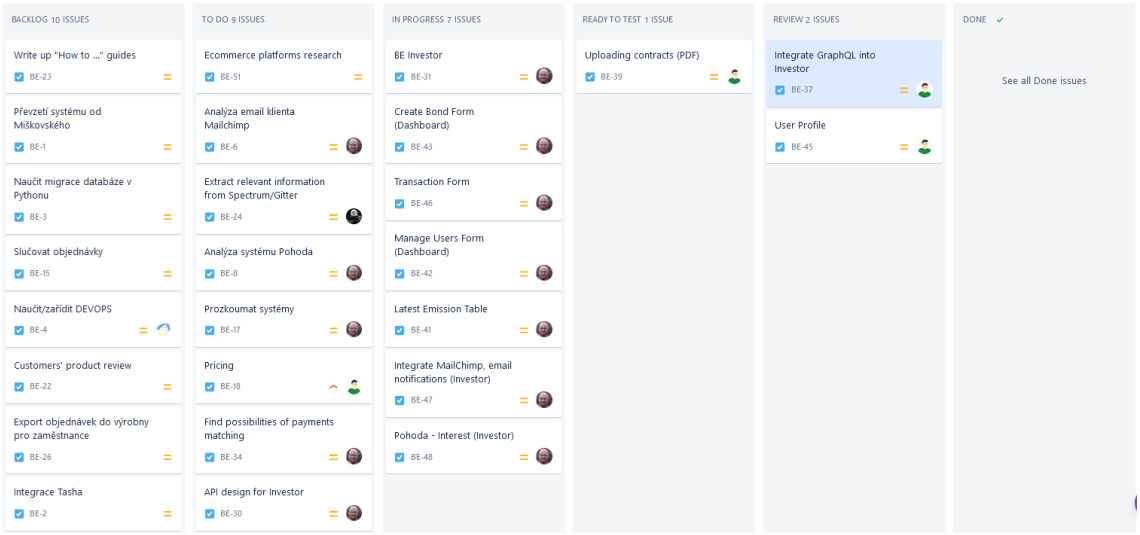
Na základě mého návrhu se tým rozhodl použít agilní metodiku Scrumban pro vývoj webové aplikace. Dalším krokem bylo určit, kdo bude zastávat role Scrum Mastera a Product Ownera. Po rychlém rozhodnutí se Product Ownerem stal manažer firmy, pro kterou byla aplikace vyvíjena, a role Scrum Mastera byla svěřena hlavnímu vývojáři. Pro každou roli bylo nutné vyjasnit úkoly. Vzhledem k tomu, že zvolený Scrum Master neměl dostatečné zkušenosti a znalosti v této roli, poskytoval jsem mu pomoc s teoretickými znalostmi, které jsem získal během zpracování teoretické části práce, při zavádění metody Scrumban do pracovního prostředí, udržování agilních nástrojů, hledání nových nástrojů pro zefektivnění týmu a práci na projektových úkolech. Product Owner měl konečné slovo při zadávání úkolů a určování jejich priorit, a zbytek týmu se stal členy vývojového týmu zodpovědnými za vývoj jednotlivých součástí webové aplikace.

Společně s vývojovým týmem jsme začali pracovat na existujícím backlogu, který byl původně vytvořen na nástroji Trello, a upravili jsme ho tak, aby odpovídal metodice Scrumban a používaným nástrojům. Tato úprava zahrnovala změnu všech sloupců s výjimkou sloupce product backlog. Každý sloupec byl rozšířen o prvek sprint, takže produktový backlog obsahoval sloupce to do, in progress, testing a done pro každý sprint. Bylo také nutné přidat sloupec "review" pro úkoly, které nebyly během sprintů dokončeny, jelikož se vyskytly úkoly, které nebyly během sprintů splněny. Tyto sloupce poskytly přehled o tom, jak se týmu dařilo během jednotlivých sprintů.

Obrázek 7 ukazuje výsledek produktového backlogu.

Obrázek 7

Jira board



Zdroj: Vlastní zpracování

Dalším krokem v našem projektu bylo uspořádání denních porad, tzv. daily stand-ups, které se konaly každé ráno a trvaly maximálně patnáct minut. Během těchto porad každý člen týmu informoval ostatní o tom, co dokončil od předchozího setkání, co má v plánu na následující období a zda narazil na nějaké problémy během práce na úkolech.

Vzhledem k tomu, že tyto schůzky byly velmi důležité pro koordinaci práce v týmu, bylo nutné zajistit, aby probíhaly efektivně a strukturovaně. Pokud se některý z členů týmu začal během porady vyjadřovat o něčem, co nesouviselo s daným tématem nebo se na řadu dostal v době, kdy mluvil někdo jiný, byl Scrum Master pověřený úkolem zasáhnout a vrátit poradu zpět na správnou kolej. Například v případě, že se konverzace rozptýlila, mohl Scrum Master navrhnout, aby si členové, kteří vedli rozhovor, dali chvilku pauzu a pokračovali po skončení denního stand-upu.

Dále jsem navrhl, aby si členové týmu dělali během daily stand-ups poznámky, což jim umožnilo udržet si přehled o postupu projektu a rychleji se vrátit k předchozím tématům, pokud bylo potřeba. Tato praxe se ukázala jako velmi užitečná, protože některé úkoly vyžadovaly více času a byly řešeny v průběhu několika setkání. Navíc

jsem doporučil Scrum Masterovi, aby po skončení stand-upu zorganizoval další setkání na řešení problému, pokud bylo třeba.

Pro koordinaci práce na úkolech bylo také nutné vytvářet nové úkoly během sprintu, pokud to bylo potřeba.

Přestože zavedení těchto čtyř nástrojů vypůjčených ze scrum metodiky, konkrétně daily stand-ups, produktového backlogu, sprintů a sprint review, poskytly vývojovému týmu základní strukturu a systém pro řízení projektu, bylo také důležité být kritický k jejich použití. Bylo nutné zajistit, aby byly tyto nástroje tým pochopil a naučil se jich využívat, a aby byly aplikovány správně a efektivně. V opačném případě by mohly být zdrojem zmatku a zpoždění v projektu.

4.4 Rozhovory s kvalifikovanými odborníky v oblasti agilních metod

V rámci implementace metody Scumbanu během vývoje jsem provedl několik rozhovorů s kvalifikovanými agilními odborníky, kteří představili své jedinečné zkušenosti a poznatky z uplatňování agilních metod ve svých projektech. Dva z nich byli certifikovaní Scrum Masters, zatímco třetí byl Agilní Kouč.

První certifikovaný Scrum Master, Novotný (2022), se zaměřil na efektivní řešení konfliktů mezi zainteresovanými stranami. Popisoval svou zkušenost s vedením workshopu, který pomohl zainteresovaným stranám porozumět potřebě kompromisu a spolupráce při stanovování priorit v rámci produktového backlogu. Během workshopu se zainteresované strany naučily, jak identifikovat své priority a potřeby a jak je sdílet s týmem. Díky této iniciativě došlo k lepšímu porozumění a koordinaci mezi zainteresovanými stranami a týmem. Druhý ScrumMaster (2022) představil svou zkušenost s adaptací Scrumu na vzdálené týmy, které pracovaly na jednom projektu ze různých časových pásem. Popsal výzvy spojené s koordinací a komunikací mezi členy týmu a jak bylo důležité přizpůsobit Scrum ceremoniály tak, aby byly přístupné pro všechny členy týmu. Použití komunikačních nástrojů, jako jsou Teams nebo Slack, a vytvoření pravidelných časových slotů pro setkání vývojového týmu pomohlo překonat tyto výzvy a zajistit úspěšné dokončení projektu. Novotný (2022) také představil zajímavý koncept tzv. "mob programming", což je přístup, kdy celý vývojový tým pracuje společně na jednom úkolu současně. Tento přístup umožnil týmu rychlejší řešení

problémů, sdílení znalostí a zlepšení komunikace mezi členy týmu. Tento koncept byl úspěšně aplikován v několika projektech, které Novotný (2022) řídil.

Novotný (2022) také přinesl příklad zavedení techniky "Definition of Done" pro úkoly v rámci projektu. Tato technika zahrnuje seznam položek, které musí být splněny pro dokončení projektu (Huether, n.d.). Strukturovaný seznam úkolů členům týmu pomohl lépe pochopit, jaké kroky je třeba splnit pro úspěšné dokončení úkolu, což vedlo k vyšší kvalitě práce a snížení chyb. Navíc to umožnilo lepší koordinaci mezi členy týmu a zajistilo, že všichni mají jednotný pohled na to, co znamená "dokončený" úkol. ScrumMaster (2022) sdílel příběh, kdy se jeho tým potýkal s neustále rostoucím technickým dluhem. Pro řešení tohoto problému navrhl, aby se každý třetí sprint zaměřil výhradně na refaktoring kódu a odstraňování technického dluhu. Tento přístup pomohl týmu snížit technický dluh a udržet kvalitu kódu na vysoké úrovni. Náš vývojový tým tak mohl zvážit zavedení podobného pravidla pro udržení kvality kódu v průběhu projektu.

Další zmíněnou technikou ze scrum metodiky byly tzv. Story points, které jsou měřítkem odhadovaného úsilí pro danou práci. Story points tedy slouží k odhadu při plánovaní příchozího sprintu. Hlavní výhodou používání Story points místo pouze časového odhadu je že hodnotí náročnost práce více komplexně a zahrnují i risky a nečekané události. Tím pádem, využívání Story points může vést k usnadnění plánování a větší efektivitě práce. Navíc, o hodnotě Story points rozhoduje každý člen týmu, a tím pádem má každý jednotlivec šanci vyjádřit svojí vlastní perspektivu ohledně náročnosti práce (Radigan, n.d.). Tým se společně může rozhodnout, že jeden Story point je nejjednodušší úkol, a podle toho odstupňovat Story points pro ostatní úkoly (ScrumMaster, 2022), nebo individuální práce pro každou osobu pracující na úkolu (Novotný, 2022), nebo může být Story points jednotkou času. Existuje nástroj, který lze použít k tomu, aby tým společně měřil hodnotu úkolu, nazývá se Scrum Poker (Anonymní, 2022). Scrum poker je technika v agilním softwarovém vývoji, kdy tým odhaduje náročnost úkolů pomocí karet s čísly a toto odhadování probíhá ve formě diskuse mezi členy týmu, kteří navzájem vysvětlují, proč se jim daný úkol zdá jednoduchý či složitý. Cílem je dosáhnout společného rozhodnutí o odhadované náročnosti úkolu a zlepšit týmovou komunikaci (Weagileyou, n.d.). Sprinty byly dvoutýdenní, přičemž každý sprint se zaměřoval pouze na jednu funkcionalitu. Každý úkol musel být dokončen a důkladně otestován (Anonymní, 2022; Novotný, 2022; ScrumMaster, 2022). Plánování bylo první a poslední věcí každého sprintu, kdy byl kompletně naplánován sprint backlog (Novotný, 2022). Pokud by bylo

nutné přidat nějaké nové úkoly, znamenalo by to, že sprint "přeteče" a bude mít příliš mnoho úkolů, což by mohlo vést k zastavení sprintu a nutnosti naplánovat sprint nový (Novotný, 2022). K těmto situacím by docházelo pouze v extrémních výjimkách (ScrumMaster, 2022). Retrospektiva, plánování a sprint review nebo sprint demo byly kombinovanou akcí, která mohla trvat až celý pracovní den (Anonymní, 2022; Novotný, 2022). Tyto nástroje zahrnovaly přezkoumání produktu se zúčastněnými stranami a to, že tým prošel poslední sprint aby zjistil, jestli bylo třeba něco zlepšit, a plánování sprintu dalšího.

Druhá část rozhovoru se týkala Kanbanu, jak funguje a jak se liší od Scrumu. Kanban je metodika řízení projektů, která využívá vizuální úkoly pro správu pracovních postupů, zatímco Scrum je metodika řízení projektů, která pomáhá týmům strukturovat a spravovat svou práci prostřednictvím souboru hodnot, principů a praktik. Přestože jsou praktiky Scrumu a Kanbanu odlišné, jejich základní principy jsou velmi podobné (Rehkopf, n.d.). Kanban je metoda, která je flexibilnější a nemá jiné striktní termíny než datum vydání produktu. Je to metoda, která vyžaduje hodně komunikace mezi týmem, nepracuje se na mnoha úkolech najednou a product owner je stejně jako ve Scrumu zodpovědný za priority úkolů. Místo sprintů se však konají týdenní schůzky s product owenerem a případně se zainteresovanými stranami, kde tým předvádí dokončenou práci. Agilní Kouč (Anonymní, 2022), který má dlouholeté zkušenosti s kanbanem se zaměřil na důležitost zavádění limitů prací v průběhu (WIP limitů) pro zajištění efektivního průtoku úkolů a prevenci zacyklení. Tyto limity omezují počet úkolů, které mohou být v každé fázi procesu současně zpracovávány (Anonymní, 2022).

Rozhovory s odborníky v oblasti agilních metod přinesly cenné poznatky a nápady pro implementaci Scrumbanu, například zmiňované metodiky jako "Definition of Done" a koncept tzv. Story points. Nicméně, bylo důležité přistupovat k těmto zkušenostem a poznatkům kriticky a zvážit jejich aplikaci v kontextu konkrétního projektu. Například "mob programming" by mohl být pro takto malý tým časově neefektivní a méně produktivní, protože by je mohlo zdržovat od jiných úkolů.

4.5 Aktualizování, údržba a vylepšování Scumbanu

Na začátku se vyskytly problémy s udržováním produktového backlogu, typické bylo zapomínání na přidávání úkolů, přesouvání úkolů z jedné fáze do druhé nebo zapomínání na jejich existenci. Denní Stand-ups probíhaly vždy standartně a po nich se konala krátká

porada: zda má někdo něco k projektu, nové nápady, další priority, které je třeba uskutečnit, a problémy s úkoly.

Poté byly implementovány nástroje Scrum pro retrospektivu a kontrolu sprintů s cílem zlepšit každý sprint. Uspořádání prvního sezení s oběma nástroji, kterého se museli zúčastnit všichni členové týmu a hovořit o projektu, v podstatě přimělo tým, aby společně usiloval o stejný cíl. Retrospektiva poskytla týmu způsob, jak volně komunikovat o vzestupech, pádech a nápadech na zlepšení sprintu, stala se také plánováním sprintu, kde si všichni naplánovali počet úkolů, které by podle nich mohli zvládnout, jaké jsou priority pro tento sprint a zda má některý z vynechaných sprintových úkolů stále vysokou prioritu, zda by na něm členové mohli začít pracovat dále, nebo by měl být přesunut zpět do kolonky product backlog.

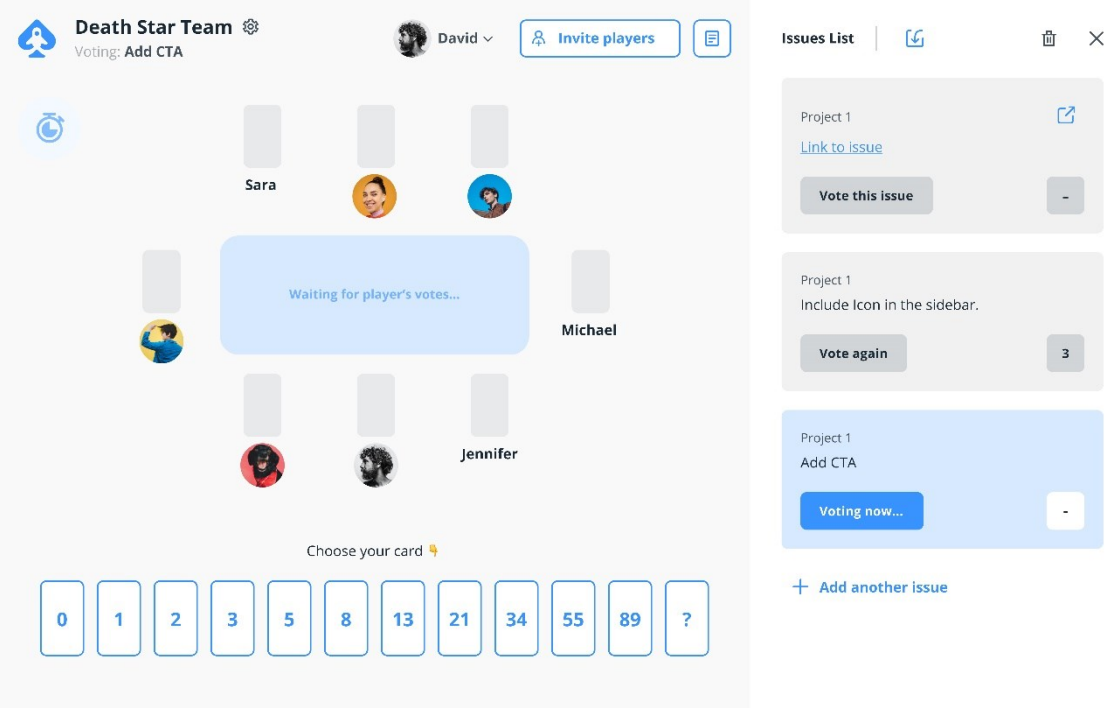
Příliš mnoho úkolů na produktovém backlogu během sprintu byl často vyskytující se problém, a tak tým potřeboval způsob, jak to udělat předvídatelnější. Byl vytvořen odhad, který byl označen jako Story Points (SP), což je měřítkem náročnosti daného úkolu.

Story Points byly zmíněny všemi třemi dotazovanými odborníky v předchozí kapitole. Rozhodl jsem se tedy Scrum Masterovi poradit, aby s týmem implementoval Scrum Poker k odhadování náročnosti jednotlivých úkolech. Při další Retrospektivě se tým společně rozhodl tyto metody otestovat.

Jakmile bylo rozhodnuto o bodech příběhu, bylo dalším krokem naplánovat, jakou hodnotu budou mít jednotlivé úkoly. Scrum tým (Product Owner, Scrum Master a vývojový tým) se rozhodl vyzkoušet scrum poker a hlasovat o příběhových bodech (SP) potřebných pro jednotlivé úkoly v příštím sprintu. Na obrázku 8 je ukázka, jak takový scrum poker vypadá. Při prvním pokusu hlasoval pouze Scrum Master a vývojový tým. Bylo to takto provedeno, protože Product Owner si nebyl jistý, jak úkoly hodnotit. Díky tomu bylo pro tým snazší diskutovat o jednotlivých úkolech a každý člen byl vyslyšen.

Obrázek 8

Scrum poker (planning poker)

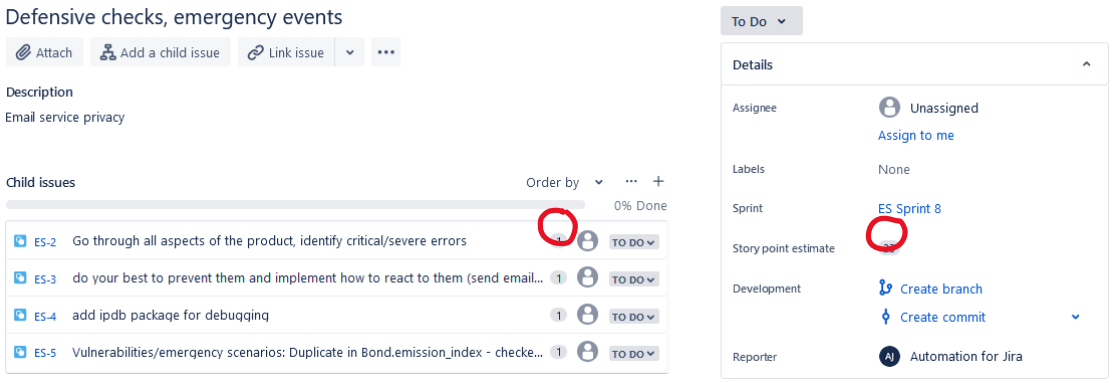


Zdroj: (Weagileyou, n.d.)

Po jednom pokusu se plánovací poker stal součástí každé sprint review, čím více jej vývojový tým používal, tím méně úkolů se zmeškalo. Bylo mnohem snazší plánovat každý sprint s vědomím přibližného množství práce, kterou je možné udělat. Dalším zjištěním týkajícím se úkolů byla jejich velikost, pokud se jednalo o větší úkoly, například něco, co tým během hlasování odhadl množstvím příběhových bodů (SP) vysoce převyšujícím běžné úkoly, musely být rozděleny na menší úkoly. Poté, co jsme se s tím setkali během sprint retrospective, padlo doporučení zachovat velké úkoly, ale vytvořit malé úkoly, které pak byly propojeny s úkolem, příklad na obrázku 9.

Obrázek 9

Větší úkol s fázemi danými SP pro jednotlivé menší úkoly



Zdroj: Vlastní zpracování

Scrumban v této podobě poskytl týmu rámec, který umožnil i týmu bez předchozích zkušeností s agilními metodikami pracovat efektivněji a lépe zvládat výzvy spojené s organizací práce na projektu.

5 Zhodnocení a diskuse

Agilní metody poskytují týmu vývojářů softwaru všechny nástroje pro práci v prostředí, které je pro ně výzvou a poskytuje týmu flexibilitu, aby mohl uspokojit zákazníky. Agilní rámce však vyžadují, aby týmy byly soběstačné a dokázaly projekt řídit. Agilní metody dávají návod při implementaci metod, nikoliv návod, jak se to dělá, což je důvod, proč firmy a jiné týmy používají metody poněkud odlišně. Existuje mnoho způsobů, jak se agilní metody používají, a vše záleží na tom, jak se je vývojové týmy naučí používat. Protože existuje mnoho způsobů, jak nástroje jednotlivých metod používat, jediným způsobem, jak skutečně zjistit, zda je nástroj pro daný tým užitečný, je vyzkoušet ho.

Scrumban je metoda, ve které se pro tým spojily nejlepší vlastnosti Kanbanu a Scrumu. Zavedení Scrumbanu dalo skupině vývojářů možnost vyzkoušet si Kanban i Scrum. Použitím tolika metod Scrumu poskytlo pohled na to, jak se tyto metody mohou používat v pracovním prostředí a co může každý očekávat, když začne pracovat pro různé, větší společnosti. Při zavádění metody Scrumban se však objevilo poměrně hodně smíšených pocitů. Hlavním problémem z mého pohledu bylo, že roli Scrum Mastera zároveň zastával hlavní vývojář projektu, a tak byl často zaneprázdněný programováním a chyběl mu tím často nadhled v krizových situacích. Z tohoto důvodu také dlouho trvalo, než tým pochopil dané scrum principy. Nástroje, které dávaly smysl od samého počátku, byly denní scrum stand-ups a produktový backlog. Tyto nástroje se po uvedení do provozu staly bez problémů součástí projektového řízení. Přidání sprintů do metody mi přišlo jako správná věc, která všem pomohla ujasnit si, kolik práce lze udělat během krátkého časového úseku. Retrospektivní část scrumu byla pro každý sprint velmi důležitá, protože vyjadřovala obavy uvnitř týmu. Každý člen mohl mluvit o problémech, které měl se sprinty, a ty se pak mohly společně řešit, a teprve pokud s tím všichni členové souhlasili, řešení se zavedlo nebo alespoň vyzkoušelo.

I přes úspěch scrumban metody v tomto projektu došlo k několika problémům, se kterými nebyl zákazník spokojen a které způsobily zdržení a prodražení projektu. Jedním z problémů byla nedostatečná analýza požadavků zákazníka na začátku projektu, což vedlo k nedorozuměním a změnám během vývoje. Tým se také potýkal s technickými výzvami, jako bylo nedostatečné zvládnutí některých nástrojů a technologií, což způsobilo další zpoždění a následné prodražení projektu. Při pohledu zpět by se dalo udělat několik věcí lépe. Například bych doporučil implementování **burndown charts**, ta mohla týmu

poskytnout lepší povědomí o tom, jakým tempem pracují a zda je třeba zvýšit úsilí, aby splnili své cíle včas.

I když scrumban byl pro mladý tým bez zkušeností s agilním řízením po nějaké době funkční, retrospektivně by se dalo zvážit použití jiné metody nebo kombinace metod, které by lépe vyhovovaly potřebám a dovednostem týmu. Při zvažování použití jiné metody je důležité vzít v úvahu, že mladý tým potřebuje čas na to, aby se naučil efektivně používat agilní nástroje a postupy. Proto by bylo vhodné provést více školení ve formě workshopů, nebo mentorování týmu Agilním koučem, aby se zlepšily jejich dovednosti a zvyšovala se úroveň zkušeností, což by mohlo vést k úspěšnějšímu průběhu příštího projektu. V ideálním případě bych doporučil do týmu najmout **certifikovaného Scrum Mastera**, který by týmu pomohl implementovat všechny části tohoto agilního rámce, a zároveň by umožnila hlavnímu vývojáři neztrácet čas řízením projektu. Díky tomu by se mohl plně věnovat vývoji softwaru v příštím projektu.

Metoda Scrumban poskytla týmu příležitost vyzkoušet si nové způsoby práce. Pouze metodou pokusů a omylů a experimentováním s různými agilními nástroji mohl tým najít ty správné nástroje pro práci

Nakonec je důležité zdůraznit, že neexistuje jedno správné řešení pro každý tým nebo projekt. Agilní metody by měly být považovány za nástroje, které týmům umožňují pružně reagovat na změny a kontinuálně se zlepšovat. Klíčem k úspěchu je nalezení vhodné kombinace metod a nástrojů, které vyhovují specifickým potřebám týmu a projektu, a pravidelné hodnocení a úprava těchto postupů v průběhu času.

Závěr

Stanoveným cílem práce bylo vymezit základní metodiky vhodné pro řízení vývoje softwaru, představit vybraný podnikatelský subjekt vyvíjející softwarový projekt a následně popsat nasazení vybrané metodiky do praxe a vyhodnotit celý postup.

Práce obsahuje část teoretickou a část praktickou. V teoretické části byly popsány oblasti projektové řízení a vymezeny základní pojmy jako je projekt, software a metoda. Dále byly zmíněny používané metody řízení vývoje softwaru, jejich obsah a organizace. Byly zde také zmíněny rozdíly mezi tradičním a agilním přístupem k řízení vývoje softwaru. V neposlední řadě zde byly podrobně popsány agilní přístupy k řízení projektů, především Scrum, Kanban, byly představeny jejich základní koncepty a uplatnění, ale také jejich možné sloučení do hybridní metody označované „Scrumban“, která hrála svojí roli v praktické části práce.

Praktická část bakalářské práce byla věnovaná implementaci procesů metodiky scrum do projektu, který v té době používal agilní praktiky převzaté z metodiky Kanban. V úvodu této části byly popsány základní charakteristiky zvoleného podnikatelského subjektu.

Dále v této části byl popsán proces plánování, nástroje použité k implementaci a kroky nezbytné k udržení a vylepšení nově zavedených Scrum principů. Tato část také obsahovala rozhovory s odborníky v oblasti agilních metod, kteří sdíleli své zkušenosti a rady z praxe. Zmíněné údaje v této části byly získány z interních materiálů a na základě osobní komunikace s členy týmu.

V poslední řadě bylo provedeno vyhodnocení procesu implementace metodiky Scrumban do vybraného projektu a byly zde zmíněny doporučení pro zavedení nových procesů pro lepší agilní fungování týmu. Tato doporučení budou předána ke zvážení hlavnímu vývojáři, který celý projekt řídil.

Seznam použité literatury

Agile Aliance. (2001). The Agile Manifesto. Načteno z agilealliance.org: https://www.agilealliance.org/agile101/the-agile-manifesto/

Ahmad, M., Markkula, J., & Ovio, M. (2013). Kanban in software development: A systematic literature review. In Software Engineering and Advanced Applications. https://doi.org/10.1109/seaa.2013.28

Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. Načteno z International Journal of Computer Science Issues (IJCSI), 12(1), 106-111.: https://www.proquest.com/docview/1660801422

Anonymní, A. K. (2022, Květen). Agilní Kouč.

Atlassian. (n.d.-b). Trello Guide. Načteno z Attlassian Trello: https://trello.com/guide

Atlassian. (n.d.). Jira Software. Načteno z https://www.atlassian.com/software/jira

Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change. Addison-Wesley.

Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21, 61-72.

Bruckner, T. (2012). Tvorba informačních systémů: principy, metodiky, architektury. Praha: Grada.

Clark, W. (2019). Agile Methodology: A Beginner's Guide to Agile Method and Principles. Nezávisle vydáno.

Cockburn, A. (2006). Agile Software Development The Cooperative Game. Addison-Wesley Professional; 2nd edition.

Cockburn, A. (2004). Crystal Clear: A Human-Powered Methodology for Small Teams: A Human-Powered Methodology for Small Teams. Addison-Wesley Professional. Načteno z https://www.informit.com/articles/article.aspx?p=345009&seqNum=3.

Dočkal, J. (2015, Únor). Metody řízení projektů. Načteno z flek.cz: https://flek.cz/clanky/dalsi-tipy-a-informace/agilni-a-waterfall-metody-rizeni-projektu

Doležal, J. (2016). Projektový management: komplexně, prakticky a podle světových standardů. Praha: Grada Publishing.

Filpal. (2022, Únor). Using Discord in Agile Practices. Načteno z filpal.wordpress.com: https://filpal.wordpress.com/2022/02/13/using-discord-in-agile-practices/

Huether, D. (n.d.). Definition of Done. Načteno z LeadingAgile: https://www.leadingagile.com/2017/02/definition-of-done/

JavaTpoint. (n.d.). www.javatpoint.com. Načteno z Spiral Model (Software Engineering) – javatpoint: https://www.javatpoint.com/software-engineering-spiral-model

Kanbanize. (n.d.). Kanban Explained for Beginners / The Complete Guide. Načteno z https://kanbanize.com/kanban-resources/getting-started/what-is-kanban

Keup, M. (2020). projectmanager.com. Načteno z What is Scrumban? How it Differs from Scrum and kanban.: https://www.projectmanager.com/blog/what-is-scrumban

Knesl, J. (2009). Agilní vývoj: Scrum. Načteno ze zdroják.cz: https://zdrojak.cz/clanky/agilni-vyvoj-scrum/

Lizner, J. (2018). easyproject. Načteno z Waterfall vs. Agile: kterou metodiku zvolit pro vaše projekty?: https://www.easyproject.cz/kontakt/rizeni-projektu-jednoduse-blog-tipy-zdroje/waterfall-vs-agile-kterou-metodiku-zvolit-pro-vase-projekty

Myslín, J. (2016). Scrum: průvodce agilním vývojem softwaru. Brno: Computer Press.

NimbleWork. (n.d.). What is Kanban? Načteno z nimblework.com: https://www.nimblework.com/kanban/what-is-kanban/

Novotný, L. (2022, Květen). Aktivní Scrum Master.

Ochrana, F. (2010). Metodologie vědy: úvod do problému. Praha: Karolinum.

PM Consulting. (n.d.). Trojimperativ projektu a jeho význam pro praxi - PM Consulting. Načteno z PM Consulting: https://www.pmconsulting.cz/pm-wiki/trojimperativ-projektu/

ProductPlan. (n.d.). Načteno z Scrumban: https://www.productplan.com/glossary/scrumban/

Radigan, D. (n.d.). Story points and estimation. Načteno z Atlassian: https://www.atlassian.com/agile/project-management/estimation

Rehkopf, M. (n.d.). Kanban vs. scrum: which agile are you? Načteno z Atlassian: https://www.atlassian.com/agile/kanban/kanban-vs-scrum#:~:text=Summary%3A%20Kanban%20is%20a%20project,values%2C%20principles%2C%20and%20practices.

Rubin, K. S. (2013). Essential scrum: A practical guide to the most popular agile process.Addison-Wesley.

Scrum.org. (n.d.). Scrum.org. Načteno z What is Scrum?: https://www.scrum.org/learning-series/what-is-scrum/what-is-scrum

ScrumMaster, A. (2022, Květen). Certifikovaný Scrum Master.

Schwaber, K. (2004). Agile Project Management with Scrum. Microsoft Press.

Schwaber, K., & Sutherland, J. (2017). The Scrum Guide. Načteno z scrumguides.org: https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

Singh, R. (2023). Waterfall Methodology. Načteno z Institute of Project Management: https://www.projectmanagement.ie/blog/waterfall-methodology/

Sommerville, I. (2013). Softwarové inženýrství. Brno: Computer Press.

Spinellis, D. (2012). GIT. IEEE Software.

StarAgile. (2020). Overview of Scrum Phases. Načteno z StarAgile: https://staragile.com/blog/scrum-phases

Svozilová, A. (2016). Projektový management: systémový přístup k řízení projektů. Praha: Grada.

Šochová, Z., & Kunce, E. (2019). Agilní metody řízení projektů. Brno: Computer Press.

Testování Softwaru. (n.d.). Vodopádový model. Načteno z Testování softwaru: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/vodopadovy-model/

Vijayasarathy, L. R., & Butler, C. W. (2016). Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? IEEE Software(5), stránky 86-94. doi:10.1109/MS.2015.26

Weagileyou. (n.d.). Planning Poker. Načteno z Planning Poker Online: https://planningpokeronline.com/

World of Agile. (2016). Burn Down Chart. Načteno z World of Agile: https://worldofagile.com/blog/burn-down-chart/

York, A. (2023). 15 Best Scrum Tools for 2023. Načteno z clickup.com: https://clickup.com/blog/scrum-tools/

Zoho.com. (n.d.). What is a scrum board. Načteno z zoho.com:

https://www.zoho.com/sprints/what-is-a-scrum-board.html

Seznam obrázků

Abstrakt

Vyrut, M. (2023). *Metody řízení vývoje softwaru ve vybrané organizaci* [Bakalářská práce, Západočeská univerzita v Plzni].

Klíčová slova: řízení vývoje softwaru, metodika projektového řízení, tradiční metody vývoje softwaru, agilní metodiky, Scrum, Kanban

Bakalářská práce se zaměřuje na vymezení základních metodik pro řízení vývoje softwaru a jejich aplikaci v konkrétním softwarovém projektu vybraného podnikatelského subjektu. Práce pokrývá teoretické základy projektového řízení, tradiční metody vývoje softwaru, agilní metodiky jako je Scrum a Kanban a jejich implementaci do praxe. V praktické části je popsána aplikace hybridní metodiky Scrumban v probíhajícím projektu, včetně nástrojů, plánování, udržení a vylepšení metodiky, a rozhovory s odborníky. Práce nabízí vyhodnocení procesu implementace a doporučení pro zavedení nových procesů pro lepší agilní fungování týmu.

Abstract

Vyrut, M. (2023). *Management methods of software development in a selected organization* [Bachelors Thesis, The University of West Bohemia, Czech Republic].

Keywords: software development management, project management methodology, traditional software development methods, agile methodologies, Scrum, Kanban

The bachelor's thesis focuses on defining the basic methodologies for software development management and their application in a specific software project of a selected business entity. The work covers the theoretical foundations of project management, traditional software development methods, agile methodologies such as Scrum and Kanban, and their implementation in practice. In the practical part, the application of the hybrid Scrumban methodology in an ongoing project is described, including tools, planning, maintenance and improvement of the methodology, and interviews with experts. The work offers an evaluation of the implementation process and recommendations for the introduction of new processes for better agile team functioning.