**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačních technologií**

# Bakalářská práce

## Testování softwaru v agilním prostředí

**Jakub Jusko**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jakub Jusko

Systémové inženýrství a informatika
Informatika

Název práce

**Testování softwaru v agilním prostředí**

Název anglicky

**Software testing in agile software development**

---

**Cíle práce**

Cílem práce je na základě studia odborných a vědeckých zdrojů vytvořit podklady pro zpracování teoretické části práce testování v agilním prostředí v porovnání s klasickým vývojem softwaru. Hlavním cílem bude zhodnocení principů, které budou v praxi doporučeny týmům pracujícím v agilních metodikách ke zkvalitnění způsobu testování.

**Metodika**

Bakalářská práce bude vycházet ze studia odborných a vědeckých literárních zdrojů, které řeší problematiku testování softwaru.

Na základě studia literárních zdrojů budou stanoveny principy, které budou důležité pro porovnání testování v agilním prostředí. Navržené principy budou ověřeny provedením dotazníkového šetření a expertními rozhovory.

**Doporučený rozsah práce**

40 stran

**Klíčová slova**

testování softwaru, agilní metodika, zlepšení testování

---

**Doporučené zdroje informací**

BUREŠ, Miroslav, Miroslav RENDA, Michal DOLEŽEL, Peter SVOBODA, Zdeněk GRÖSSL, Martin KOMÁREK, Ondřej MACEK a Radoslav MLYNÁŘ. Efektivní testování softwaru: klíčové otázky pro efektivitu testovacího procesu. Praha: Grada, 2016. Profesionál. ISBN 978-80-247-5594-6.

CRISPIN, Lisa a Janet GREGORY. Agile testing: a practical guide for testers and agile teams. New Jersey: Addison-Wesley, c2009. The Addison-Wesley signature series. ISBN 9780321534460.

MYERS, Glenford J., Tom BADGETT a Corey SANDLER. The art of software testing. 3rd ed. Hoboken, New Jersey: Wiley, c2012. ISBN 1118031962.

---

**Předběžný termín obhajoby**

2021/22 LS – PEF

**Vedoucí práce**

doc. Ing. Edita Šilerová, Ph.D.

**Garantující pracoviště**

Katedra informačních technologií

Elektronicky schváleno dne 10. 8. 2021

**doc. Ing. Jiří Vaněk, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 19. 10. 2021

**Ing. Martin Pelikán, Ph.D.**

Děkan

V Praze dne 15. 03. 2022

**Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci "Testování softwaru v agilním prostředí" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2022 _____

**Poděkování**

Rád bych touto cestou poděkoval vedoucí práce doc. Ing. Editě Šilerové, Ph.D. za odborné vedení a také své rodině a přítelkyni za podporu.

# Testování softwaru v agilním prostředí


## Abstrakt

Tato bakalářská práce se zabývá tématem testování softwaru zejména v rámci agilních metodik, ale definuje i základy tradičního přístupu k testování softwaru, které slouží k následnému porovnání těchto přístupů. Jelikož principy všech agilních metodik vychází z Manifestu agilního vývoje softwaru, jsou nejprve vymezeny jeho myšlenky. Dále se práce věnuje popisu agilního a tradičního přístupu k vývoji softwaru, kde popisuje základní principy jednotlivých přístupů. Práce se věnuje oblasti testování, jeho důležitosti a typům a způsobům testování. Závěr teoretické práce je věnován hlavním rozdílům v testování z pohledu agilního a tradičního přístupu. V analytické části se nejprve práce věnuje navržení jednotlivých principů jakožto hlavnímu cíli této práce a následně se věnuje ověření navržených principů v praxi za pomoci dotazníkového šetření a expertních rozhovorů. V rámci obou provedených výzkumu byl, jako dílčí cíl, také hodnocen stav agilních týmů, ve kterých byly uvedené výzkumy prováděny.



**Klíčová slova:** testování softwaru, agilní metodika, tradiční způsob vývoje, zlepšení testování, principy pro zlepšení testování, porovnání metodik

# Software testing in agile software development

**Abstract**

This bachelor thesis deals with the topic of software testing, especially in the context of Agile methodologies, and defines the basics of the traditional approach to software testing, which is used for subsequent comparison of these approaches. Given the principles of all agile methodologies are based on the Agile Software Development Manifesto, its ideas are defined first. The work also describes the Agile and traditional approaches to software development and describes the basic principles of each approach. Furthermore, the thesis deals with the field of testing, its importance, and types of testing. The conclusion of the theoretical work is devoted to defining the main differences in testing in terms of Agile and traditional approaches. In the analytical part, the work firstly deals with defining individual principles as the main goal of this thesis secondly, attention will be paid to proven principles in practice with the help of a questionnaire survey and expert interviews. The research was carried out within agile teams, and in both surveys, the condition of these teams was also evaluated as a partial goal.

**Keywords**: software testing, Agile methodology, traditional software development, improvement of testing, principles for testing improvement, comparison of methodologies

# Obsah

# Seznam obrázků



# Seznam tabulek

# Seznam použitých zkratek

| | |
|---|---|
| BTT | Bug-Tracking Tool |
| CJ | Customer Journey expert |
| PO | Product Owner |
| SDLC | Software Development Life Cycle |
| FAA | Federal Aviation Administration |
| USD | United States Dollar |
| TDD | Test-Driven Development |
| ATDD | Acceptance Test-Driven Development |

# 1 Úvod

Během posledních let se testování softwaru stalo jednou z nejdůležitějších částí celého procesu vývoje softwaru, a to i díky častější implementaci agilních metodik, které nejsou již výhradou malých organizací, ale stále častěji je přejímají i velké korporátní společnosti. Průzkum *15th State of Agile Report* ukazuje, že: „došlo k významnému nárustu osvojení agilních metodik v rámci týmů pro vývoj softwaru, které se zvýšilo z 37 % z roku 2020 na 86 % v roce 2021“ (Digital, 2021, přeloženo autorem[1]).

S agilními metodikami se váže nejen upřednostnění testování, ale také změny ve struktuře týmů a požadavků na dovednosti jejich členů. Znalosti a jednotlivé role členů se navzájem prolínají, a proto za kvalitu dodávaného softwaru již nezodpovídají pouze testeři, ale celý vývojový tým. Také role zákazníka je zcela odlišná. Zákazníkova participace na vývoji softwaru nekončí pouhým předáním požadavků na samém začátku procesu, ale zákazník je součástí procesu po celou dobu. Díky krátkým iterací vývoje pravidelně konzultuje pokroky vývojového týmu a poskytuje mu zpětnou vazbu. Vzhledem k posunu autonomie na agilní týmy až jednotlivce dochází k narušení tradičních organizačních struktur, i v tomto tkví složitost implementace agilních metodik, jelikož ne každá organizace je na tento způsob struktury připravena. Práce se proto nezabývá pouze testováním jako takovým, ale i definicí jak agilního, tak tradičního přístupu k vývoji softwaru a vymezení jejich rozdílů.

Cílem této práce je tedy návrh principů pro zlepšení testování, objasnění celkového fungování agilních metodik a tradičního způsobu vývoje, vysvětlení rozdílů mezi těmito přístupy a detailní popis přístupu k testování v agilním prostředí a jeho porovnání s testováním v tradičních metodikách.

[1] Z originálu: „This year's findings indicate significant growth in Agile adoption within software development teams, increasing from 37 % in 2020 to 86 % in 2021.“

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Hlavním cílem teoretické části práce je vymezení základních pojmů v oblasti testování a agilních metodik, porovnání klasického způsobu vývoje softwaru s agilním způsobem a poukázání na hlavní rozdíly mezi těmito přístupy.

V praktické části práce je hlavním cílem navržení a zhodnocení principů pro zlepšení testování v agilním prostředí, dílčím cílem je hodnocení fungování týmů, které používají agilní metodiky, a ve kterých jsou výzkumy prováděny.

## 2.2 Metodika

V teoretické části práce se jedná o rešerši a analýzu odborných zdrojů zabývajících se problematikou testování softwaru a agilních metodik.

V praktické části je používán kvantitativní a kvalitativní výzkum cílený na členy agilních týmů zabývajících se testováním aplikací ve společnosti působící v České republice a dochází k analýze a porovnání výsledků těchto výzkumů.

# 3 Teoretická východiska

## 3.1 Agilní metodiky

Agilní přístup k vývoji softwaru je znám již od počátku 90. let, ale vždy byl využíván malými týmy a organizacemi a většina středních a velkých organizací k tomuto přístupu vývoje přecházela až během několika posledních let. Důvodů pro vznik nového pohledu na vývoj softwaru bylo hned několik – jedny z hlavních důvodů byly omezování kreativity vývojářů přemírou dokumentace a byrokratických procesů, nutnost lepší reakce na stále častěji se vyskytující změny požadavků a technologií anebo také upřednostnění koncového uživatele a jeho zapojení do procesu vývoje softwaru (Bureš a kol., 2016, s. 31).

### 3.1.1 Agilní manifest

Základní hodnoty agilních metodik vychází z Agilního manifestu, který byl vytvořen v roce 2001. Dnes existuji již velká řada agilních metodik a jejich variací, ale všechny vychází právě z Agilního manifestu a z jeho 4 hlavních hodnot (Agile manifesto, 2001):

1. **Jednotlivci a interakce** před procesy a nástroji
2. **Fungující software** před vyčerpávající dokumentací
3. **Spolupráce se zákazníkem** před vyjednáváním o smlouvě
4. **Reagování na změny** před dodržením plánu

*„Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více“* (Agilní manifesto, 2001). Agilní manifest dále rozšiřuje dvanáct principů agilního vývoje softwaru, které jsou uvedeny v Příloze 1.

### 3.1.2 Základní principy fungování

Použitím hodnot z Agilního manifestu vedeme tým k doručení malých částí produktu ve velmi krátkých intervalech (Crispin, Gregory, 2012, s. 3). Tyto krátké intervaly jsou zpravidla 1–4týdenní iterace (Crispin, Gregory, 2012, s. 8), kdy tým prochází všemi fázemi vývoje. Dochází také k zániku klasického „rozdělení“ pozic na testery, vývojáře a analytiky, nicméně v rámci této práce členy takto budeme rozdělovat, i když vhodnější definice by byla například „člen zajišťující testování“ = tester.

### Agilní tým a jeho struktura

Agilní tým, často také nazývaný squad, se skládá ze dvou základních komponentů, a to ze zákazníka a vývojového týmu. Mezi největší rozdíly mezi klasickým a agilním vývojem patří změny v přístupu a velikosti týmu. Agilní tým čítá mezi 3-10 členy a používáme zde tzv. „celotýmový přístup" ke kvalitě. Tým v agilních metodikách je samoorganizovaný nicméně nedochází zde k úplné absenci managementu, protože roli manažera zde zastupuje Product Owner[2]. Kvalita produktu již není zodpovědností pouze testerů, jak tomu bývá v tradičních metodách vývoje, ale celého týmu. Díky tomu lze dodávat vysoce kvalitní software za interval, který maximalizuje hodnotu daného produktu pro business (Crispin, Gregory, 2012, s. 15).

Na straně zákazníka jsou členové týmu, kteří definují business část produktu: Business Experti, Product Owner a Business Analytici (Crispin, Gregora, 2012, s. 7). Tato část týmu je zodpovědná za definici požadavků na začátku každé iterace a tyto definované požadavky následně předají vývojovému týmu, který je zodpovědný za jejich vývoj a doručení.

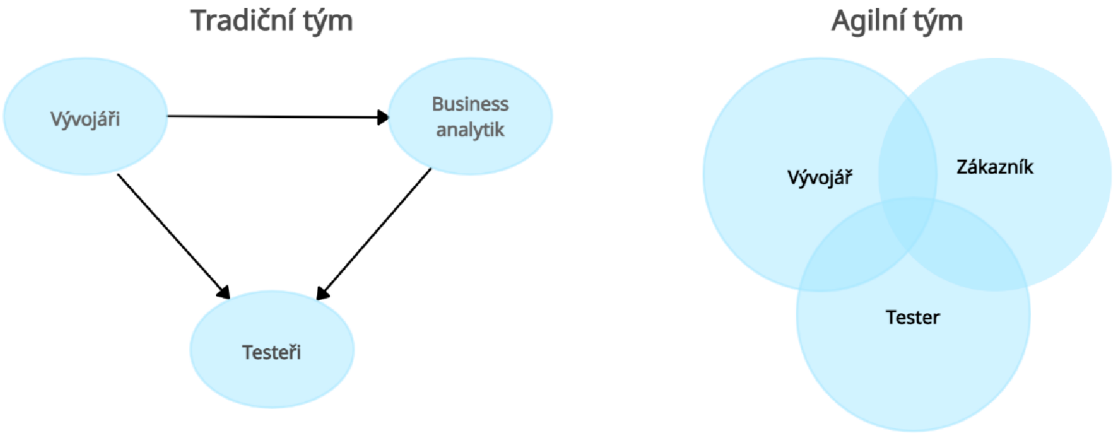
Vývojářský tým jsou všichni členové, kteří se podílejí na vývoji daného produktu. Agilní metodiky, oproti tradičnímu způsobu vývoje, podporují zastupitelnost a rozvoj dovedností všech členů v potřebných oblastech. Je však důležité, aby členové týmu byli na začátku experty v oblastech, který daný projekt potřebuje (Crispin, Gregory, 2012, s. 7) a až později získávaly ostatní potřebné dovednosti.

### Spolupráce agilního týmu

Jak zákazník, tak vývojový tým spolupracují velmi úzce po celou dobu iterace, v ideálním případě je možné je brát jako jediný tým se společným cílem – doručení přidané hodnoty společnosti, jejíž jsou součástí. Zákazník navrhne požadavky pro danou iteraci, vývojový tým k nim poskytne zpětnou vazbu ohledně časové náročnosti a proveditelnosti, zákazník rozhodne o prioritách požadavků a vývojový tým dané požadavky vyvine (Crispin, Gregory, 2012, s. 8). Narozdíl od týmu využívající tradiční způsob vývoje, v agilním týmu je spolupráce mnohem užší a často dochází k prolínání rolí (Obrázek 1).

[2] Role Product Ownera vychází z metodiky Scrum a je blíže popsána v kapitole 3.1.3.

Obrázek 1: Rozdíl mezi tradičním a agilním týmem




Zdroj: (Crispin, Gregory, 2012), vlastní zpracování

### 3.1.3 **Agilní metody**

Agilních přístupu existuje velké množství a každý z nich implementuje hodnoty Agilního manifestu různými způsoby. Tato práce se zaměřuje na popis třech základních agilních přístupů (ISTQB, 2014):

1. Extreme Programming,
2. kanban,
3. scrum.

**Extreme Programming (XP)**

*„Extreme programming je proces, který vývojářům pomáhá vytvářet velmi rychle kvalitní kód, kdy kvalitu kódu definujeme jako naplnění požadavků a očekávání zákazníka“* (Badgett, Mayers, Sandler, 2012 s. 180). Extreme programming se zaměřuje na (Badgett, Mayers, Sandler, 2012 s. 180, přeloženo autorem[3]):

- implementaci jednoduchých návrhů,
- komunikaci mezi vývojáři a zákazníkem,
- průběžné testování kódu,
- refaktoring,
- neustálou zpětnou vazbu.

[3] Z originálu: „XP is a software process that helps developers create highquality code, rapidly. Here, we define ‘‘quality’’ as a code base that meets the design specification and customer expectation.“

### Kanban

Kanban je systém pro vizualizaci a optimalizaci pracovního postupu v týmu za pomoci karet. Každá z karet představuje jeden úkol a jejich počet je omezený množstvím práce, na jaké je tým schopný souběžně pracovat. Po dokončení úkolu je tato karta vrácena zpět do oběhu a je možné k ní přiřadit další úkol (Anderson, 2010).

V případě vývoje softwaru se samostatný přístup Kanban využívá pouze v 6 % případů (Digital, 2021), častěji jej nalezneme v kombinaci s některým z dalších přístupů, například ScrumBan, který kombinuje metody Scrum a Kanban a podle průzkumu Digital (2021) ho využívá 9 % dotázaných.

### Scrum

Nejrozšířenější a nejpoužívanějším agilním přístupem celosvětově je, dle průzkumu Digital v roce 2021, Scrum, který využívá 66 % dotázaných. Scrum byl představen v roce 1995 (Schwaber, 1995) ve spolupráci Kena Schwabera a Jeffa Sutherlanda a definuje konkrétní nástroje a praktiky, které mají týmy používat. Přístup Scrum (Obrázek 2) vychází z předpokladu, že analýza, návrh a vývoj během Sprintu[4] jsou nepředvídatelné a zavádí kontrolní mechanismus, který právě tuto nepředvídatelnost a risk kontroluje (Schwaber, 1995).

*„Čím blíže vývojový tým pracuje k hraně chaosu, zatímco udržuje pořádek, tím více bude výsledný systém konkurenceschopný a užitečný“ (Schwaber, 1995, přeloženo autorem[5])*

Schwaber (1995) definuje tyto charakteristiky Scrum přístupu jako:

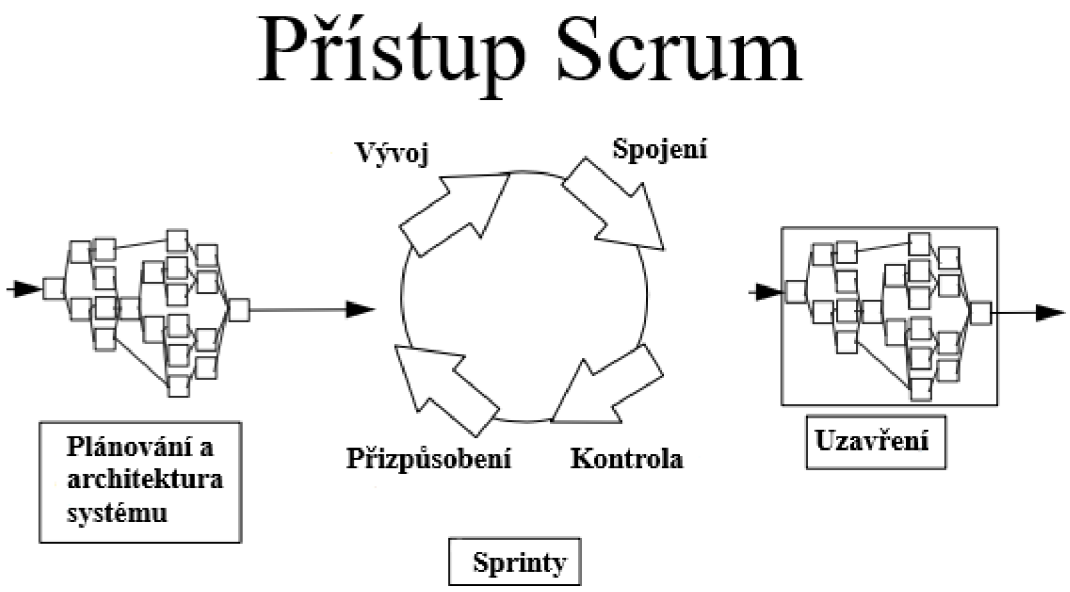
1. Veškeré procesy, vstupy a výstupy ve fázi plánování a uzavření[6] jsou jasně definovány a jak tyto procesy zpracovat je explicitní. Tok práce je lineární, s možnou iterací během fáze plánování.
2. Fáze Sprintu je postavena na empirických procesech a poznatcích, jelikož většina těchto procesů je neidentifikovatelných a nepředvídatelných, a je na ní nahlíženo metodou černé skříňky. Metody potřebné pro udržení kontroly a minimalizace risku

[4] Sprint je neměnící se krátký časový úsek, za který tým dodá naplánované množství práce.
[5] Z originálu: „The closer the development team operates to the edge of chaos, while still maintaining order, the more competitive and useful the resulting system will be.“
[6] Z angl. closure

jsou implementovány do každé iterace s cílem vyhnout se chaosu při maximalizaci flexibility.

3. Sprinty jsou nelineární a flexibilní. Kdykoli je to možné, využíváme explicitní znalost, jinak je možné využít metodu pokus-omyl, ze které následně získáváme znalosti explicitní. Sprinty jsou využívány na neustále zlepšování finálního produktu.
4. Projekt je otevřen externím činitelům[7] po celou dobu až do fáze uzavření. Doručitelnost produktu se může kdykoli během plánování i Spritu změnit.

Obrázek 2: Přístup Scrum




Zdroj: (Schwaber, 1995), vlastní zpracování

Scrum následně definuje tři role týmu (ISQTB, 2014):

- Scrum Master – zajišťuje dodržování a správné využívání metodiky Scrum. Nejedná se o vedoucího týmu, nýbrž o jakéhosi trenéra metodik.
- Product Owner – reprezentuje v týmu zákazníka, který vytváří, kontroluje a určuje priority jednotlivých požadavků. Stejně jako v případě Scrum Mastera se nejedná o vedoucího týmu.
- Vývojový tým – zajišťuje vývoj a testování produktu, tým je samoorganizující, veškerá rozhodnutí dělá celý vývojový tým a všichni členové by měli být zastupitelní.

[7] Z angl. environment

## 3.2 Tradiční způsob vývoje

V této kapitole se práce zabývá popisem tradičního přístupu k vývoji softwaru pro následné porovnání. Tradiční způsob vývoje vychází z jednotlivých fází životního cyklu informačních systémů[8] což je model pro návrh, vytvoření a udržování informačních systémů. Existuje celá řada těchto modelů, každá má své výhody a nevýhody a je doporučováno jejich použití pro různé druhy projektů. Jejich společným znakem je sekvenčnost – jedna fáze musí být zcela ukončena, aby mohla začít fáze další, a proto je velmi důležité na začátku každého projektu zvolit vhodný model, podle kterého bude software vyvíjený. Tato práce pracuje pouze se 3 základními modely.

### 3.2.1 Vodopádový model

Vodopádový model (Obrázek 3) je nejznámější a také nejstarší SDLC model a je stále velmi často využívaný ve velkých organizacích nebo státních projektech. Hlavním znakem tohoto modelu, jako celého tradičního způsobu vývoje, je jeho sekvenčnost – žádná z dalších fází nemůže začít, aniž by předchozí skončila. Specifická je také délka celého procesu, která se pohybuje v řádech měsíců či roků. Tyto fáze se dělí na (Alshamrani, Bahattah, 2015):

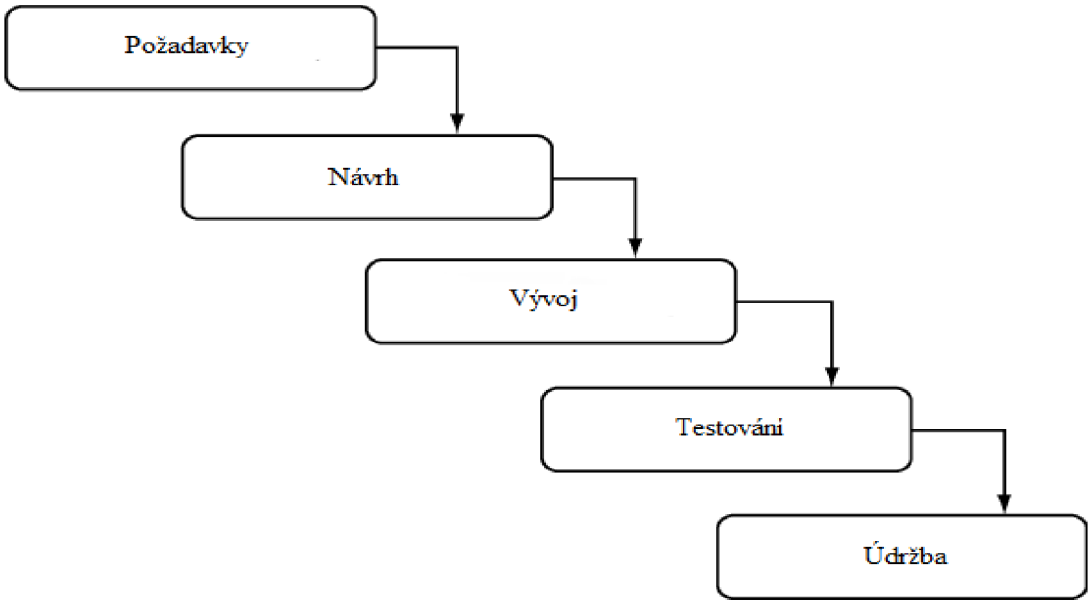
- požadavky,
- návrh,
- vývoj,
- testování,
- nasazení,
- údržba.

V první fázi dochází k sběru požadavků od klienta na specifikaci daného softwaru a jednotlivých funkcionalit. Následně dochází k analýze těchto požadavků a k vytvoření dokumentace, která následně slouží jako celková opora celého procesu ve všech jeho fázích. V případě právě vodopádového modelu je toto nejdůležitější fáze celého procesu, jelikož po jejím ukončení je velmi obtížná a nákladná jakákoli úprava specifikací, i z toho důvodu se doporučuje vodopádový model použít pouze pokud jsou požadavky velmi dobře známé a neměnné.

[8] Z angl. Systems Development Life Cycle - SDLC

Ve fázi návrhu se vymezené požadavky z minulé fáze zhodnotí a navrhne se konkrétní řešení, které bude použito –například architektura softwaru a datová struktura. Ve fázi vývoje dochází k vývoji softwaru podle dané dokumentace a v následné fázi testování dochází k porovnání výsledného softwaru s dokumentací, a k opravě veškerých nalezených chyb. Po nasazení softwaru je nutná průběžná údržba, a to ať se jedná o různé úpravy, opravy chyb nebo zabezpečení.

Obrázek 3: Vodopádový model vývoje softwaru:

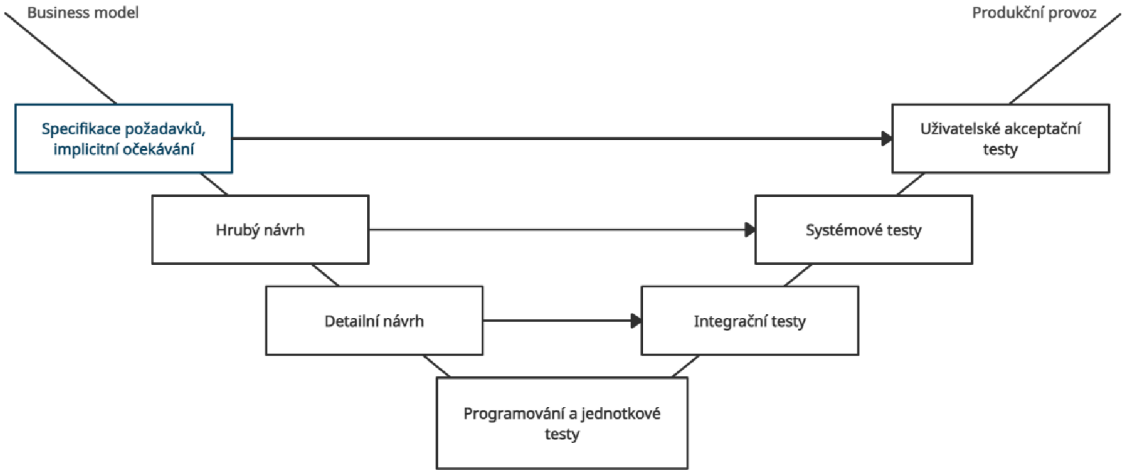



Zdroj: (Marsden, 2018), vlastní zpracování

### 3.2.2 **V-model**

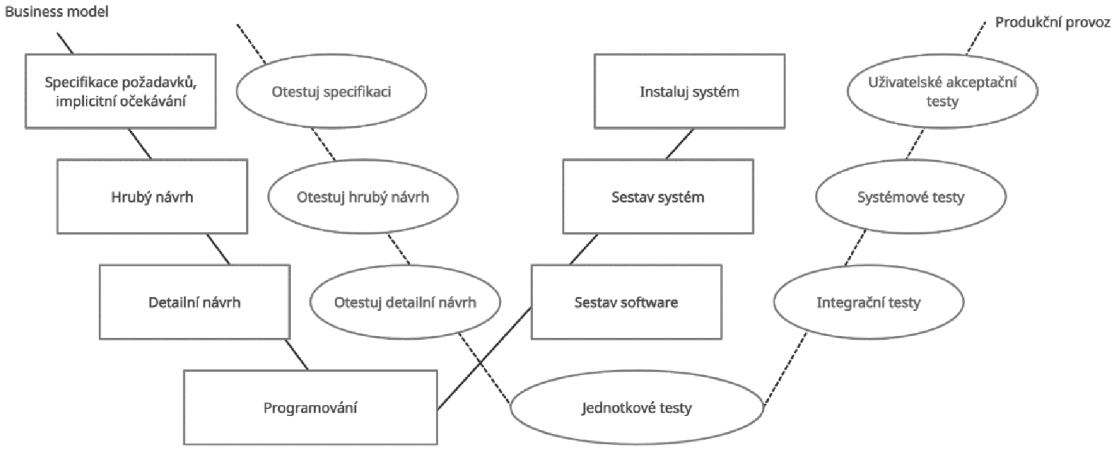
Podobně jako vodopádový model i V-model (Obrázek 4) je sekvenční, nicméně je zde kladen větší důraz na testování a testovací scénáře jsou vytvářeny v každé fázi ještě před začátkem samotného programování. Na levé straně se nachází aktivity spojené se specifikací a návrhem, na straně pravé jsou aktivity testování. Tím máme jasně danou spojitost mezi fází návrhu a potřebnými testy a po ukončení dané fáze vytvoříme testovací scénáře odpovídající dané úrovni. Testování tedy sice probíhá, stejně jako u vodopádového modelu, až v pozdějších fázích vývoje, nicméně jsou již dobře připravené testovací scénáře a testeři se s produktem nesetkávají poprvé až v této fázi. Tento nedostatek můžeme odstranit použitím W-modelu (Obrázek 5), který zapojuje testery již do počátečních fázích vývoje (levá část modelu) v tzv. statickém testování. (Bureš a kol., 2016, s. 28)

Obrázek 4: V-model




Zdroj: (Bureš a kol., 2016), vlastní zpracování

Obrázek 5: W-model




Zdroj: (Bureš a kol., 2016), vlastní zpracování

### 3.2.3 **Spirálový model**

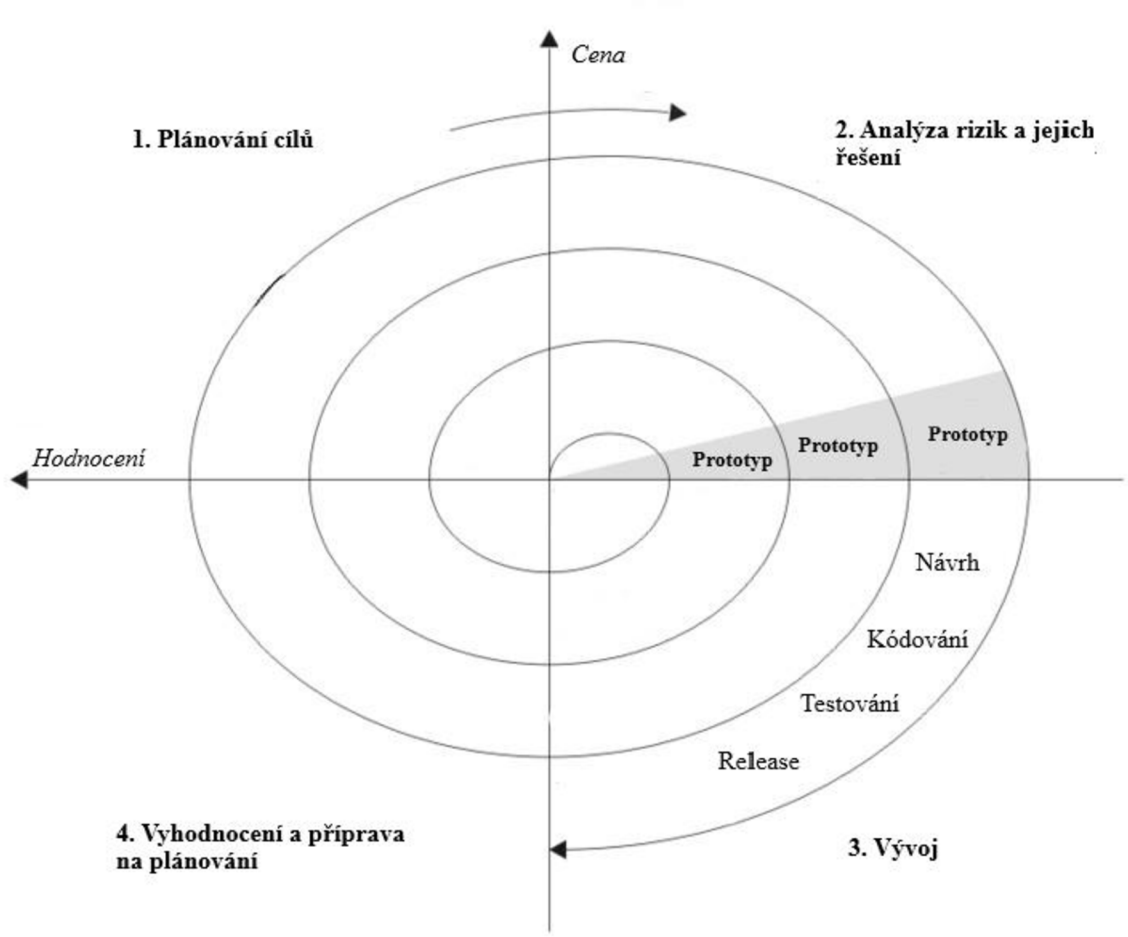
Spirálový model (Obrázek 6) je ve velké míře podobný agilnímu přístupu, jde o přístup založený na malých a iterativních přírůstcích, ale hlavním zaměřením tohoto modelu je analýza rizik. Tento model se skládá z těchto fází (Alshamrani, Bahattah, 2015):

- plánování,
- analýza rizik,
- vývoj,
- vyhodnocení.

Tyto čtyři fáze se opakují v jednotlivých iteracích a podobně jako v agilním přístupu by po každé iteraci měl být připravený produkt, které je možné vydat. Navíc po každé fázi plánovaní vznikne prototyp, který pomůže komunikaci se zákazníkem. Hlavním rozdílem oproti agilnímu přístupu je, že délky iterací jsou řádově delší a stejně jako v případě dvou předchozích modelů, k testování dochází až po dokončení fáze vývoje (Alshamrani, Bahattah, 2015).

Výhodami tohoto modelu je velká míra zaměření na analýzu rizik. Model je tedy vhodné využít z časového hlediska v případě velkých projektů a u systémů, které jsou kritické pro fungování organizace. Nevýhodou systému je jeho nákladnost a také to, že fáze analýzy rizik má velkou roly na úspěšnost projektu a je tedy nutné její bezchybné provedení.

Obrázek 6 : Spirálový model




Zdroj: (Boehm, 2000), vlastní zpracování

## 3.3 Testování

Testování softwaru je proces zkoumání kvality a naplnění požadovaného účelu daného produktu za použití různých metod. Nejedná se pouze o hledání chyb, jak se většina lidí milně domnívá, ale o komplexní přístup, který zajišťuje kvalitu dodávaného softwaru.

*„I když je testování softwaru úkol technický, je důležité si uvědomit, že zahrnuje stránku jak ekonomickou, tak stránku lidské psychologie. V ideálním případě bychom chtěli otestovat všechny možné scénáře, ale toho není možné dosáhnout, a proto je důležité, aby tester měl správný „vhled" do potřeb a chování uživatele, a ne pouze technické dovednosti."* (Mayers, Badgett, Sandler, 2012, s. 5, přeloženo autorem[9]). Definicí testování softwaru je nespočet a mohou se v pohledech lišit, nicméně základní myšlenka je stejná, a to zajištění dodání kvalitního a použitelného softwaru.

Principy jakéhokoli softwarového testování by se daly shrnout do základních deseti bodů (Mayers, Badgett, Sandler, 2012, s. 13):

1. Nezbytnou součástí testovacích scénářů je definice očekávaných výstupů nebo výsledků.
2. Programátor by se měl vyhnout pokusům o testování vlastních programů.
3. Skupina zodpovědná za vývoj by neměla testovat vlastní programy.
4. Jakýkoli proces testování by měl obsahovat důkladnou analýzu výsledků testů.
5. Testovací scénáře musí být psány jak pro validní a očekávané, tak nevalidní a neočekáváné vstupy.
6. Testovat program, zda nedělá to, co má dělat je pouze polovina práce. Ta druhá spočívá v kontrole, jestli program dělá to, co by neměl.
7. Je nutné otestovat i software u kterého je pravděpodobné, že nebude nasazen. Testovat nemusíme pouze software, u kterého je naprosto jisté, že nasazen nebude.
8. Neplánovat testovací scénáře pod domněnkou, že nebude nalezena žádná chyba.
9. Pravděpodobnost existence dalších chyb je úměrná počtu již nalezených chyb v dané části programu.

[9] Z originálu: „Software testing is a technical task, yes, but it also involves some important considerations of economics and human psychology. In an ideal world, we would want to test every possible permutation of a program. In most cases, however, this simply is not possible (…). The software tester needs the proper attitude (perhaps ''vision'' is a better word) to successfully test a software application. In some cases, the tester's attitude may be more important than the actual process itself"

10. Testování je velmi náročný úkol jak intelektuálně, tak i kreativně.

### 3.3.1 **Důležitost testování**

Je prakticky nemožné najít v dnešní společnosti oblast, kde by nebyla využívána zařízení s jakýmkoli druhem softwaru, ať už se jedná o výrobní procesy nebo státní infrastrukturu. Pro většinu lidí je zcela běžné, že pro většinu běžných úkonů využívají online systémy, ať už se jedná například o online bankovnictví, nakupování, nebo v poslední době práci z domova. Je proto nutné vytvořit systémy s robustním a bezpečným softwarem, který bude neustále dostupný. Důležitost testování ovšem netkví pouze v této rovině, ale v případě firem, které tyto zařízení dodávají, předchází ekonomickým ztrátám a ztrátě renomé v případě ať už nefunkčnosti zařízení, nebo dokonce prolomení jeho bezpečnosti. Důležitost testování tedy můžeme shrnout do těchto oblastí:

1. Ekonomická oblast
2. Bezpečnostní oblast
3. Kvalita produktu

**Ekonomická oblast**

Často je testování přehlíženou nebo podfinancovanou oblastí z důvodu jeho relativně vysoké ceny, která na výsledném produktu není na první pohled vidět. Můžeme zde uvést příklad (Bureš a kol., 2016, s. 15-20), kdy americká FAA zakázala provoz letadel Boeing 787, než bude opravena softwarová chyba, která způsobovala pád hlavního systému do nouzového režimu po nepřetržitém provozu 248 dnů. Přímé ekonomické náklady na pouhé restartování a oddálení chyby všech letadel stála 22 865 USD, nepřímé finanční dopady, jako poškození veřejného mínění či ušlý zisk se pohybuje až o dva řády výše. A nakonec, cena samotné opravy chyby, kdy bylo nutné analyzovat 110 milionů řádků kódu, se pochybuje v řádech několika milionů dolarů. Tomu všemu by bylo možné předejít investicí několika desítek tisíc USD do vhodného způsobu testování.

V Tabulce 1 můžeme vidět typické přímé náklady na opravu defektu v závislosti na fázi, kde byla nalezena (Bureš a kol., 2016).

Tabulka 1: Typické přímé náklady na opravu defektu

| | Analýza | Design | Vývoj | Vývojářské testy | Testování | Produkční provoz |
|---|---|---|---|---|---|---|
| Relativní počet zanesených defektů v dané fázi | 10 % | 40 % | 50 % | - | - | - |
| Relativní počet detekovaných defektů v dané fázi | 3 % | 5 % | 7 % | 25 % | 50 % | 10 % |
| Normalizované náklady na opravu defektů (EUR) | 250 | 250 | 250 | 1000 | 3000 | 12 500 |

Zdroj: (Bureš a kol., 2016, s. 16), vlastní zpracování

**Bezpečnostní oblast**

Bezpečnost softwarů je komplexní a kritickou oblastí. Je nutné chránit nejen samotná zařízení, ale především data a prostředky jejich uživatelů. Není proto výjimkou, že na všechna zařízení jsou vydávány neustále nové softwarové opravy. Každá firma si uvědomuje riziko, že případné narušení této bezpečnosti vede jak k poškození dat uživatelů, tak i ke ztrátě renomé dané firmy a s tím spojené ekonomické dopady.

**Kvalita produktu**

Kvalita produktu vždy byla a bude velmi důležitým měřítkem. Zákazníci budou spíše využívat produkty od společností, kterým můžou důvěřovat a které pravidelně dodávají produkty, které naplňují jejich potřeby. V dnešní době posun hardwarových možností není takový a zákazník volí tedy jemu příjemnější softwarové řešení.

### 3.3.2 Typy a způsoby testování

Testování, jak v tradičním, tak i v agilním přístupu k vývoji softwaru, se v zásadě neliší v použitých metodách jako takových. Způsoby a metody jsou podobné, nicméně v agilních

metodikách je potřeba zvolit například jiný přístup, anebo využít metody lépe uzpůsobené pro tento styl vývoje. V této kapitole se zaměříme na základní rozdělení testů a metody specifické pro testování v agilních metodikách rozvedeme až v kapitole navazující.

Testy můžeme rozdělit do jednotlivých úrovní podle toho, na jakou fázi SDLC navazují. Jednotlivé úrovně jsou (ISTQB, 2019):

- komponentové testování,
- integrační testování,
- systémové testování,
- akceptační testování.

Lepší vizualizaci jednotlivých úrovní a jejich vazbu na fáze SDLC můžeme vidět na Obrázku 4 a Obrázku 5.

**Komponentové testování**

Komponentové testování se zaměřuje na komponenty, které je možné otestovat samostatně a jeho cílem je především ověřit funkčnost (ISTQB, 2019):

- jednotek,
- modulů,
- datové struktury,
- tříd,
- databázových modulů.

Jedná se tedy o testování zdrojového kódu na té nejpodrobnější úrovni, z pravidla je prováděno vývojáři a jedná se o tzv. metodu bílé skříňky, kdy jsou testy psány na základě znalosti kódu. Typickými chybami, které je možné zjistit, jsou problémy s tokem dat nebo chyby v kódu. Pro tyto typy testů je také velmi běžná automatizace.

Výstupy komponentového testování slouží pro informovanost vývojového týmu o funkčnosti kódu a splnění zadaných požadavků od zákazníka. Interpretace těchto výsledků zákazníkům je vzhledem k požadovaným znalostem a složitosti skoro nemožná a nerelevantní.

**Integrační testování**

Integrační testování se zaměřuje na vzájemnou kompatibilitu a integraci komponent, které byly otestovány v rámci komponentového testování. Integrační testování se využívá ve dvou případech (ISTQB, 2019):

- Je potřeba otestovat, zda vyvinutá komponenta je kompatibilní se systémem, do kterého jí začleňujeme.
- Je potřeba otestovat interakci mezi jednotlivými systémy, balíčky a službami.

Jako příklad výstupních chyb z integračního testování můžeme uvést (ISTQB, 2019):

- Chybějící data nebo jejich špatná interpretace.
- Špatný návrh rozhraní.
- Chyba v komunikaci mezi jednotlivými komponenty.
- Špatné zabezpečení.

Stejně jako v případě komponentového testování i zde je velmi běžná automatizace testování a interpretace výsledků zákazníkům nerelevantní.

**Systémové testování**

Systémové testování se zaměřuje na chování systému jako celku a jeho schopnosti vykonat požadovaný úkol od začátku do konce[10]. Výstupy ze systémového testování jsou často prezentovány zákazníkům, kteří se na jejich základě rozhodují o vydání softwaru anebo se používají jako podklad pro zákonné a regulační požadavky a standardy (ISTQB, 2019).

Chyby, na které se tyto testy zaměřují jsou následující (ISTQB, 2019):

- Chybové nebo neočekáváné chování systému.
- Neschopnost provést zadaný úkol.
- Chybné datové toky v systému.
- Nefunkčnost systému v požadovaném prostředí.

Tyto testy jsou prováděny na základě tzv. specifikací – rozsáhlá dokumentace popisující podrobné chování systému. Jakákoli chyba ve specifikaci, například chybějící nebo špatně

[10] Z angl. end-to-end testing

formulovaný požadavek, mohou zapříčinit špatné vyhodnocení testu, což vede k zbytečné ztrátě času a zdrojů (ISTQB, 2019).

**Akceptační testování**

Akceptační testování je velmi podobné systémovému testování, ale z pravidla probíhá na straně zákazníka, dochází zde k validaci kvality a funkčnosti systému jako celku a testování je často prováděno na již reálném prostředí nebo jeho simulaci, aby se zákazník přesvědčil o naplnění jeho požadavků (ISTQB, 2019). Často zde bývají začleněné i testy výkonnosti a použitelnosti a testy používají metodu černé skříňky, kdy k testování dochází bez znalosti kódu.

### 3.3.3 Agilní testování

Testování v agilních metodikách se vyznačuje tím, že k testování dochází po každém malém přírůstku kódu[11] jakmile je dokončen. Tým se soustředí na vyvinutí jednoho z požadavků pro danou iterace a vzápětí ho ihned testuje, jelikož daný požadavek není možné označit jako dokončený[12] pouze jeho nakódováním, ale poté, až dojde i k jeho otestování. Po ověření jeho funkčnosti se daný požadavek označí za dokončený a tým se může přesunout k dalšímu požadavku. Z toho vyplývá, že vývojáři nemohou nikdy být „napřed" se svou prací před testery (Crispin, Gregory, 2009, s. 13).

**Test-First přístup**

Test-first přístup byl představen Kentem Beckem v roce 1999 v knize „*Extreme Programming (XP): Explained*" kde autor uvádí metodu psaní jednotkových testů ještě před samotným vývojem a jejich neustále retestování v průběhu kódování. V roce 2000 vydal knihu „*Test Driven Development by Example*" kde zavedl do Test-First přístupu velmi důležitý aspekt agilního přístupu, tzv. refaktoring (Brown, 2020) tedy proces neustálého vylepšování kódu. Definice refaktoringu je podle Badgett, Myers, Sandler (2012): „*Vyčištění a zjednodušení zdrojového kódu kdy pomocí jednotkových testů docílíte zachování funkcionalit v průběhu"* – přeloženo autorem[13].

[11] Z angl. increment
[12] Z angl. Done
[13] Z originálu: „Clean up and streamline your code base. Unit tests help ensure that you do not destroy the functionality in the process. You must rerun all unit tests after any refactoring"

Za test-first přístupy je možné považovat následující tři: Test-Driven Development, Acceptance Test-Driven Development a Behavioral-Driven Development a všechny zmíněné přístupy se vyznačují velkou mírou automatizace testů.

**Test-Driven Development**

Test-Driven Development je metoda, kdy testovací scénáře jsou nejen napsané před samotným vývojem kódu, ale samotné psaní kódu je přizpůsobeno právě těmto testovacím scénářům – kód je psaný tak, aby splnil testovací scénáře (Crispin, Gregory, 2009, s.109-112). TDD je využívaný pro vývoj řízený automatickými testy a definuje tyto kroky (ISTQB, 2014):

- napsání testu, který odráží pochopení požadavku vývojářem,
- spuštění testu, který by měl mít chybový výstup, jelikož daný kód ještě neexistuje,
- psát kód v iteracích a spouštět test do té doby, než test projde,
- refaktorovat napsaný kód, poté znovu test spustit a ujistit se, že refaktorovaný kód test stále splňuje,
- opakování tohoto procesu pro další požadavek a spuštění nejen aktuálního, ale i všech předchozích testů.

**Acceptance Test-Driven Development**

Tento přístup vychází z přístupu Test-Driven Development a jde o kolaborativní přístup, který umožňuje zákazníkům pochopit, jak mají jednotlivé vyvíjené komponenty fungovat a k vývoji jednotkových testů jsou využívány User stories (ISTQB, 2014), což jsou scénáře chování „reálného“ uživatele napsané zákazníkem. Návrh testů není tedy pouze v kompetenci vývojového týmu a díky výraznému snížení možné chybovosti v komunikaci při definici požadavku testy odpovídají více zákazníkovým představám

**Behavioral-Driven Development**

Tento přístup je kombinací Test-Driven Develoment a Acceptance Test-Driven Develoment, kde jsou testy psány na základě očekávaného chování systému. Stejně jako v případě ATDD, podaní výstupů z těchto testů je pro zákazníky srozumitelnější. Behavioral-Driven Development se vyznačuje použitím různých formátů pro definici akceptačních kritérií, jedním z nich je tzv. Given-When-Then (ISTQB, 2014):

- Given – jsou dané počáteční vstupy,
- When – a v případě nějakého jevu,
- Then – to má tyto výsledky.

**Průzkumné testování**

Průzkumné testování nereprezentuje testovací metodu jako takovou, jedná se spíše o přístup k testování a je typickým přístupem v agilních metodikách. Průzkumné testování je kombinací neustálého učení se o daném systému, návrhu testů a jejich provádění. Je prováděno manuálně, nicméně některé jeho části je možné automatizovat. Jde o pochopení produktu na velmi podrobné úrovni a testování již použitých story a scénářů, zda opravdu splňují všechny možné varianty či přístupy, které by mohli nabídnout. Průzkumné testování je nestrukturované a kompletně v kompetenci daného testera. Častými výstupy jsou další požadavky či úprava existujících požadavků a řešení. Schopnost provádět průzkumné testy s validními výstupy je u agilních testerů velmi oceňovanou dovedností (Crispin, Gregory. 2009, s.195-200).

**Kvadranty testování**

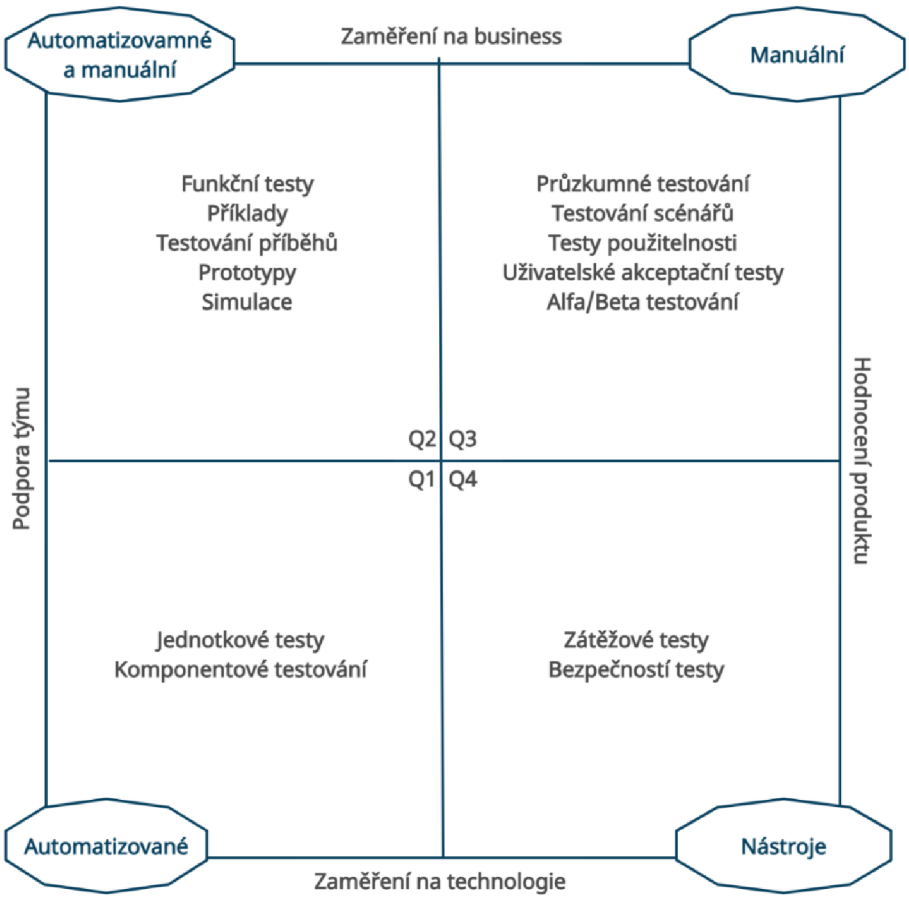
Crispin a Gregory (2009) rozdělují přístup k agilnímu testování do čtyř kvadrantů (Obrázek 7) a to podle různých důvodů, proč software testovat:

- Kvadrant 1 - otestování technologií daného produktu[14].
- Kvadrant 2 - podpora vývojové týmu[15].
- Kvadrant 3 - otestování konkurence schopnosti a využitelnosti v reálném provozu[16].
- Kvadrant 4 - hodnocení produktu jako celku[17].

Tyto kvadranty neudávají, v jakém pořadí máme testy vykonávat, ale poskytují celkový pohled na to, které úkony a procesy je žádoucí udělat. Vyhodnocení pořadí je specifické pro každý projekt a mělo by vyplývat z potřeb zákazníka a z možných rizik (Crispin, Gregory, 2009, s. 98).

[14] Z originálu: Technology-facing tests that support the team
[15] Z originálu: Business-facing tests that support the team
[16] Z originálu: Business-facing tests that critique the product
[17] Z originálu: Critiquing the product using technology-facing tests

Obrázek 7: Kvadranty agilního testování




Zdroj: (Gregory, Crispin, 2009), vlastní zpracování

**Kvadrant 1**

První kvadrant reprezentuje Test-Driven Development, což je velmi častý přístup k testování v agilních metodikách a který je považován za jeho klíčovou podstatu.

Pokud tým využívá přístup TDD, vývojový tým minimalizuje počet chyb, které je nutné řešit v pozdější fázi vývoje. Většině chyb ve zdrojovém kód je možné předejít právě psaním testů ještě před samotným kódem. Takovýto test je napsán tak, aby zachytil podstatu zákazníkova požadavku a kdykoli, v případě jakýchkoli nejasností nebo otázek, se vývojový tým může přímo zákazníka zeptat. Vhodné je také spárování vývojáře a testera, tzv. párové testování, které zaručí komplexní pohled na daný požadavek. Tímto přístupem se zajistí, že většina chyb ve zdrojovém kódu je odhalena v rané fázi vývoje a budou opraveny právě zde, než aby byly odhaleny až v pozdější fázi, zpozdili dodání a tím ovlivnili celkovou kvalitu

produktu (Crispin, Gregory, 2009, s. 111). Je také nutné dbát na dosažitelnou velikost jednotlivých story a tou by měla být právě jedna iterace. Občas je tedy nutné jednotlivé story rozdělit na několik menších dílů, nebo naopak několik jednodušších story spojit do jedné větší (Crispin, Gregory, 2009, s. 144).

**Kvadrant 2**

Výstupy z testů v druhém kvadrantu se zaměřují nejen na zhodnocení vývojovým týmem, ale i zákazníkem a demonstrují externí kvalitu a požadavky, které zákazník požaduje. Tyto testy jsou psány na základě User stories, které poskytne zákazník (Crispin, Gregory, 2009, s. 99) a jsou jednoduše interpretovatelné. Jako příklad zjednodušené použité User story může být: „Jako zákazník na internetovém obchodě chci dopravu zdarma, pokud součet cen mých objednávek překoná požadovanou hranici pro dopravu zdarma a tím využiji toho, že objednávám ve velkém“ (Crispin, Gregory, 2009, s. 130, přeloženo autorem[18]). Některé testy mohou být velmi podobné testům z prvního kvadrantu, nicméně hlavním cílem testů v druhém kvadrantu je demonstrace funkčnosti pro zákazníka, něčeho takového testy z prvního kvadrantu dosáhnout, ve většině případů, nemohou.

Stejně jako v případě testů prvního kvadrantu, i v druhém kvadrantu je do velké míry možné testy automatizovat. Tato automatizace nám pomůže například při následném přidávání funkcionalit, kdy můžeme spustit vytvořenou sadu automatických testů a během krátké doby vidíme, zda jsme tím nenarušili některou z již implementovaných funkcionalit. Tato automatizace nám také umožní věnovat se více testům z kvadrantu třetího a čtvrtého, které jsou ve většině případů manuální (Crispin, Gregory, 2009, s. 190).

**Kvadrant 3**

Testování v třetím kvadrantu se zaměřuje na hodnocení produktu a probíhá především manuálně. Je však možné automatizovat procesy jako například příprava dat potřebných k testování. Výhodou agilního přístupu je, že na hodnocení od zákazníka nemusíme čekat až do fáze vydání produktu, jako tomu je v tradičním přístupu, ale máme možnost i několikrát během jedné iterace konzultovat dosavadní výsledky se zákazníkem. Tím se zajistí, že zákazník může zhodnotit přínosnost produktu nejen před jeho samotným vydáním, ale

[18] Z originálu: „As an internet shopper (…) I want free shipping when my order exceeds the free shipping threshold, so that I can take advantage of ordering more at one time.“

zároveň po malých přírůstcích a tím zlepšit celkovou kvalitu dodávaného produktu (Crispin, Gregory, 2009, s. 190–192). V tomto kvadrantu Crispin a Gregory (2009) zmiňují důležitou součást agilního testování – průzkumné testování.

**Kvadrant 4**

Čtvrtý kvadrant se již nezaměřuje na tzv. funkční testy jako tři předchozí, ale více na nefunkční testy – sem mohou patřit například (Crispin, Gregory, 2009, s. 217):

- zátěžové testy,
- testy bezpečnosti,
- testy škálovatelnosti,
- testy spolehlivosti.

Často se stává, že díky zaměření zákazníka čistě na business a funkční stránku produktu, se opomíjí robustnost systému jako takového, nebo se předpokládá, že se o tuto stránku postará automaticky vývojový tým. Proto je důležité, aby vývojový tým zákazníkovi vysvětlit důležitost těchto testů a důsledky jejich přehlížení nebo upozadění. Problém je, že provádění testů z tohoto kvadrantu po každém přidaném požadavku je buď nemožné (chybějící architektura, systém není ještě nasazený a podobné) nebo velmi časové i finančně náročné. Vývojový tým by tedy měl se zákazníkem probrat případné upřednostnění těch požadavků, které vedou k vystavení takto otestovatelného systému v závislosti na hlavních požadavcích, které od systému má (Crispin, Gregory, 2009, s. 217-223). Pokud je daný produkt například klíčovou infrastrukturou pro internetové bankovnictví, je nutné upřednostnit bezpečnost, spolehlivost a zvládnutelnost zatížení.

## 3.4 Porovnání tradičního a agilního testování

Jedním z hlavních rozdílů agilního přístupu oproti tradičnímu je průběžné testování softwaru v případě agilních metodik. Sekvenční procházení jednotlivými fázemi v tradičních metodikách neumožňuje začít s testováním dříve, než je vyvinutý celý systém a velmi často se stává, že celé testování je uskutečněno až těsně před nasazením. Pokud fáze vývoje zabere větší množství času, než je očekáváno, může se stát, že na některé části testování vůbec nedojde (Bureš a kol., 2016, s. 29). V takovémto případě výrazně trpí kvalita dodávaného

systému, jelikož nedojde k odstranění chyb před nasazením a jejich oprava je v rámci nasazení výrazně komplikovanější a nákladnější.

Agilní přístup je mnohem flexibilnější a je zde možné testovat prakticky současně s vývojem kódu. Je zde důraz na tzv. „celo-týmový“ přístup a testeři již nejsou jediní, kteří jsou zodpovědní za kvalitu dodávaného produktu. Je zde také zcela jiné pojetí členů týmu, tradiční pojmenování tester, vývojář aj. se upozaďují, jelikož členové týmu by měli mít dovednosti ze všech odvětví, které daný tým pro vývoj potřebuje. Mění se také přístup k zákazníkovi, ten se dostává do klíčové pozice v rámci vývoje. V ideálním případě spolupracuje s týmem na denní bázi, hodnotí dosavadní výstupy a dodává podněty pro vývojový tým. Díky tomu je mnohem pravděpodobnější, že výsledný produkt bude podle zákazníkových představ.

Sběr požadavků v agilním testování je také velmi odlišný od tradičního přístupu. Týmy netráví měsíce jejich sběrem, analýzou a vyhodnocováním, jen aby později zjistili, že některé požadavky je potřeba změnit nebo použité technologie jsou již zastaralé. Místo toho zákazník požadavky formuluje v krátkých story a příkladech. Poté je na komunikaci vývojového týmu a zákazníka, aby dané požadavky formulovali co nejlépe. K této komunikaci jsou nejlépe přizpůsobeni právě testeři, jelikož mají jak potřebné technické znalosti, tak jsou schopni vnímat celkový obraz[19] produktu. Nástroji, které jsou ideální pro tuto komunikaci, jsou různé prototypy a simulace – ty umožní zákazníkovi vidět a pochopit požadavky nejlépe (Crispin, Gregory, 2009, s. 129–138).

[19] Z angl. big picture

# 4 Analytická část

Tato kapitola se zaměřuje na výzkum, který byl proveden za pomoci dotazníkového šetření a expertních rozhovorů. Toto šetření bylo provedeno ve vybrané bankovní společnosti působící na území České republiky a cílilo pouze na vybrané zaměstnance v IT odvětví. Dotazníkové šetření bylo využito jako kvantitativní sběr dat z důvodu většího počtu zaměstnanců, snazšího zapojení pro respondenty a zachování anonymity. Následné expertní rozhovory diskutují otázku agilních metodik v dané společnosti a navržené principy pro zlepšení testování.

## 4.1 Navržené principy umožňující zlepšení testování v agilních metodikách

Cílem této podkapitoly je navržení principů, které by měli mít pozitivní vliv na efektivitu spolupráce všech členů týmu, zvýšení produktivity a zlepšení způsobu dodání produktu a s tím související lepší zkušenost pro koncového zákazníka. Ačkoli se většina principů týká primárně testování softwaru, vzhledem ke struktuře agilních týmů je možné, principy považovat, do určité míry, za obecné. Návrh principů vychází z rešerše odborných publikací a následně jsou tyto principy konzultovány v rámci expertních rozhovorů s odborníky z praxe. Navrhnuté principy jsou následující:

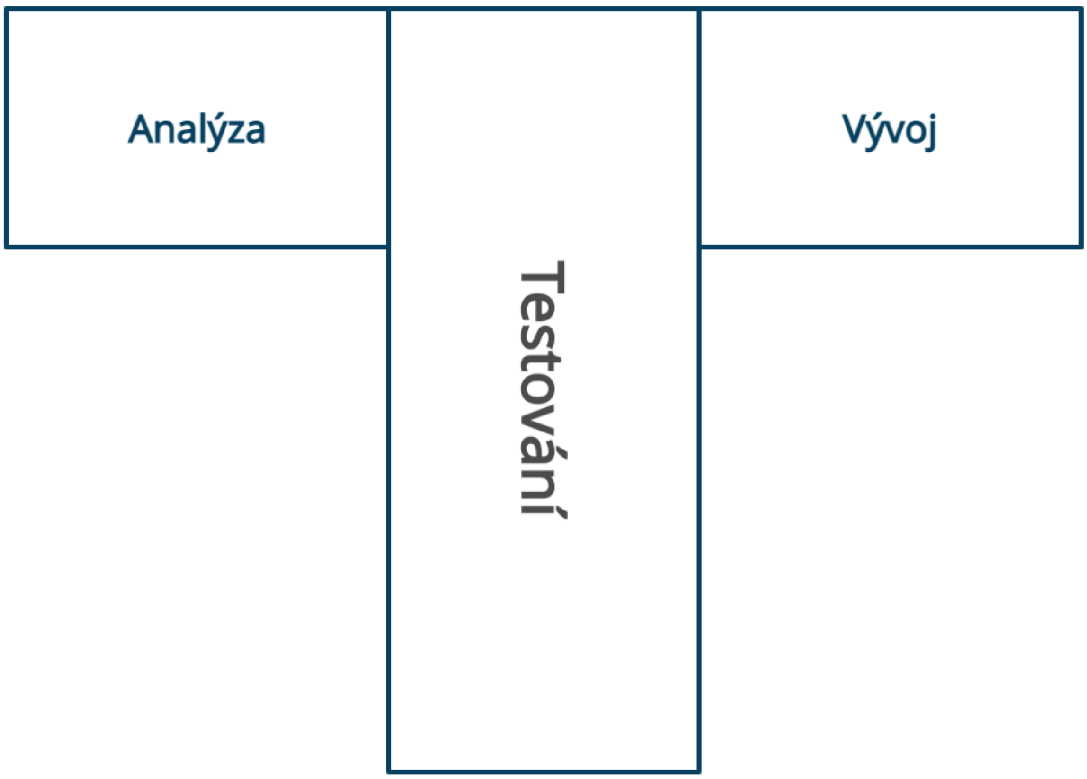
1. Definujte testování jako nedílnou a rovnocennou součást celého vývoje softwaru.
2. Aktivně zapojte testery do plánování sprintu a odhadněte náročnost úkolů.
3. Klaďte důraz na komunikaci v týmu.
4. Zapojte do testovacího procesu co nejvíce členů.
5. Zvolte vhodný a jednotný způsob pro záznam chyb.
6. Určete metriky úspěšnosti sprintu.

### 4.1.1 Definujte testování jako nedílnou a rovnocennou součást celého vývoje softwaru

V rámci agilních týmu je snaha o vyhnutí se klasické definici pozic jako tester, analytik, vývojář aj., jelikož každý člen týmu by měl mít, nad rámec expertízy ve svém odvětví, alespoň základní přehled o všech pozicích a být schopný tyto pozice zastoupit. K popisu rozdělní jednotlivých znalostí můžeme použít u testera například koncept T-dovedností,

který je na Obrázku 8. Vertikální část reprezentuje dominantní expertízu, horizontální všeobecnou znalost.

Obrázek 8: T-Shape dovednosti testera




Zdroj: Autor

Často se ovšem členové, zaměřující se na testing, setkávají s podhodnocováním jejich práce a obecně je testování přikládána menší důležitost. Tento jev je přejat z tradičních způsobů vývoje a je možné se s ním setkat i v týmech, které agilní metodiky přijali v nedávné době.

Je tedy nutné definovat testování jako nedílnou a rovnocennou součást celého vývoje softwaru a v rámci jednotlivých ceremonií[20] mu přikládat dostatečnou důležitost. Často je na testování nahlíženo jako pouhé spouštění a prověřování činností a chování systémů během jejich běhu – tzv. dynamické testování. Praxe nám však ukazuje, že testeři jsou velmi vhodní pro tzv. statické testování (otestování požadavků, hrubého/detailního návrhu) v úvodní fázi vývoje a měli by být v této činnosti podporováni. (Bureš a kol., 2016, s. 30)

[20] Ceremonie je název pro schůzky s předem stanovenou délkou, frekvencí a cíli.

### 4.1.2 Aktivně zapojte testery do plánování sprintu a odhadněte náročnost úkolů

Aktivním zapojením testerů do plánovaní sprintu můžete již v úvodní fázi předejít problémům s dokončením funkcionalit během sprintu. Jedním z možných řešení je přidat testerské podúkoly k jednotlivým bodům plánování jako například přípravu testovacích dat, přípravu testovacích scénářů nebo opravy a úpravy automatizovaných testů. Nejen, že v této fázi můžeš dojít k nalezení nějakého problému, ale tester bude mít pří předání úkolu od vývojářů již předpřipravená data a testy a bude možné testování značně urychlit (Hák, 2020).

Dále je při plánování sprintu vhodné určovat náročnost úkolu, a to ať z pohledu pouze vývoje nebo testingu nebo obou částí zároveň. Pokud se tým soustředí na hodnocení náročnosti pouze z pohledu vývoje, může poté dojít k nedostatku času k provedení testování a obráceně. Ideální tedy je určovat náročnost z obou pohledů současně a úkoly s největší náročností odbavovat priorit. Časem může dojít k vytvoření vzorových Story (Hák, 2020) a tím nadále zjednodušit přístup nejen současným členům, ale případně i členům novým.

### 4.1.3 Klaďte důraz na komunikaci v týmu

Dobrá komunikace je klíč k úspěchu a agilní týmy jsou na ní postavené, ať už se jedná o komunikaci se zákazníkem nebo uvnitř týmu. V rámci komunikace je jedním z hlavních cílů testera porozumět dané problematice jak z business pohledu zákazníka, tak z technického úhlu vývojáře. Pokud se při testování objeví jakékoli pochybnosti ve splnění požadavku vývojáři, tester je právě tím členem týmu, který má potřebné znalosti v mediaci rozhovorů mezi vývojáři a zákazníkem, případně Product Ownerem (Crispin, Gregory, 2009, s. 23-24)

### 4.1.4 Zapojte do testovacího procesu co nejvíce členů

Je vhodné do procesu testování zapojit celý tým. K tomu není nutné znát veškeré možné varianty a scénáře tak podrobně, jako tester, ale díky znalostem a informacím z předchozích třech bodů těchto principů by každý člen týmu měl být schopný, v případě potřeby a do určitě míry, zastoupit role testera.

Větší zapojení například vývojářů do testování bývá často opomíjena z nedostatečného zájmu a informací na straně vývojářů, ale především v případě automatizovaných testů je jejich zapojení prospěšné pro celý tým. Vývojář může díky znalostem psát základní testy pro sebou vyvíjený kód a díky následné validaci ze strany testera se vyhneme porušení principu 2 z kapitoly 3.1. této práce. Vývojář tímto zjistí případné problémy s jeho kódem již v rané fázi vývoje a tester se zdokonalí ve své znalosti kódu, což vede z dalšímu rozvoji jednotlivých členů týmu a k přidané hodnotě výsledného produktu (Hák, 2020).

### 4.1.5 **Zvolte vhodný a jednotný způsob pro záznam chyb**

Je nutné zvážit veškerá pro a proti možných přístupů k záznamu chyb v agilním týmu a najít takový způsob, který bude pro daný tým nejefektivnější. Většina testerů byla z tradičního způsobu vývoje zvyklá na využívání BTT a dle průzkumu (Digital, 2021) 62 % respondentů v agilních týmech stále BTT využívá. Je ovšem nutné nespoléhat se pouze na zvolený nástroj, lidská interakce by měla být vždy upřednostňována.

### 4.1.6 **Určete metriky úspěšnosti sprintu**

Zvolení správných metrik a jejich správná interpretace je klíčová k úspěchu agilního týmu. Vzhledem k razantnímu zkrácení dodání produktu oproti klasickému přístupu je nutné hodnotit postup týmu na dennodenní bázi a získaná data správně interpretovat nejen pro daný tým, ale i pro zákazníka.

Základní doporučení korelují s třemi nejoblíbenějšími metrikami dle průzkumu (VersionOne, 2015):

1. Rychlost vývojového týmu za danou iteraci.
2. Práce odvedená během jedné iterace.
3. Práce odvedená za celý release[21].

[21] Vydání hotového produktu

## 4.2 Dotazníkové šetření

### 4.2.1 Prostředí firmy

Společnost, ve které je prováděn výzkum, se v nedávné době transformovala na agilní metodiky a je organizačně dělena podle Spotify[22] modelu. Zaměstnanci jsou rozděleni do Squadů, které dále spadají pod jednotlivé Triby dle svého zaměření. Tato společnost byla vybrána nejen na základě nedávné transformace, ale i díky velkému počtu pracovníků, kteří v rámci jednotlivých Squadů mají na starost vývoj celé aplikace nebo její části. Aktivity jednotlivých Squadů se pohybují od vývoje aplikací, bankovního API, testování, Bussiness Intelligence až po správu a provoz aplikací na testovacím, vývojovém nebo produkčním prostředí. Výjimkou není ani správa vlastních serverů.

Z důvodu velké provázanosti jednotlivých aplikací je nutné, aby všechny aplikace dokázali společně komunikovat a předávat si data, ale také aby měli jednotný design pro jednodušší orientaci klientů. Ke komunikaci s klientem banka využívá všechny běžně používané způsoby jako webové stránky, internetové a mobilní bankovnictví a další způsoby.

### 4.2.2 Sestavení dotazníku

Dotazník vznikl na základě rešerše odborných publikací a studiem dané problematiky. Jednotlivé otázky a možnosti odpovědí byly také konzultovány s odborníkem z praxe, aby bylo zajištěno, že je znění pro respondenty jasně srozumitelné a docházelo tak k co nejmenšímu zkreslení výstupních dat. K analýze byl použit nástroj MS Excel a MS Forms.

### 4.2.3 Struktura dotazníku

Dotazník byl složen z několika částí a bylo použito větvení navázané na odpovědi. První část dotazníku se zaměřuje na demografické údaje a charakteristiku respondentů. Následuje část zabývající se používanými agilními metodami a jejich hodnocení. V poslední části dochází k ohodnocení jednotlivých principů a respondent zde má také možnost případně vyjádřit své další myšlenky, které nebyly součástí definovaných otázek.

[22] Spotify model je přístup ke struktuře organizace umožňující agilitu (zdroj: https://www.atlassian.com/agile/agile-at-scale/spotify)

Formy použitých otázek byly:

- otevřené otázky,
- uzavřené výběrové otázky s jednou odpovědí,
- uzavřené výběrové otázky s více odpověďmi,
- otázky s hodnotící škálou (tzv. Likertova škála).

Dotazník byl vytvořen v prostředí MS Forms, které nabízí firma Microsoft v rámci balíčku Office 365.

**Část 1 - charakteristika respondenta a demografické údaje**

První část dotazníku se zaměřuje na charakteristiku respondenta a získává informace obecného typu jako zastávaná role v týmu, délku zaměstnání, dosažené vzdělání apod. Všechny tyto otázky nám umožní sestavit profil respondenta a mohou být využity k další analýze dat. Pokud daný respondent byl zaměstnancem i před transformací banky na agilní metodiky, dotazník se dále zaměřuje na obecný přístup před a po transformaci a také na celkové vnímání transformace. V Příloze 2 je kompletní sada všech otázek pro tento dotazník.

**Část 2 - používané agilní metody**

Druhá část dotazníku se zabývá jednotlivými metodami agilních metodik a jejich využití v týmech. Část těchto otázek byla inspirována reportem *15th Annual State Of Agile Report* firmy Digital (2021), která uvádí procentuální zastoupení různých metodik v rámci respondentů. Autoři tohoto dotazníkového šetření byli schopni zanalyzovat 1382 odpovědí z celkových 4182, které zahrnovaly respondenty z více jak 100 zemí a široké škály velkostí organizací (Digital, 2021). V druhé části dotazníku se dále objevují otázky s hodnotící škálou pro obecný popis aktuálního fungování týmu. Otázky se týkají oblastí komunikace a spolupráce jednotlivých členů týmu, rychlosti reakce na změny v požadavcích nebo opravy nalezených chyb, celkovou kvalitu dodávaného softwaru a schopnost včasného dodání.

**Část 3 – hodnocení definovaných principů**

Otázky v třetí části dotazníkového šetření se týkaly principů, které definuje tato práce. Pomocí hodnotící škály měli respondenti určit důležitost jednotlivých tvrzení. V závěrečné části byla položena otevřená otázka v případě potřeby respondenta doplnit další informace k dotazníku.

### 4.2.4 Distribuce dotazníku

Vzhledem k zaměření dotazníkového šetření na již zmíněnou bankovní společnost byl výběr respondentů jasný. Přestože k výzkumu došlo pouze v rámci jedné společnosti, bylo nutné zařídit relevantní šíření dotazníkového šetření, a to pouze na ty týmy, které se zaměřují na vývoj a dodávku softwaru. K šíření byla využita primárné pomoc jednotlivých Chapter[23] Leadů za IT, kteří dotazníkové šetření šířili do tvých týmů, sekundárně brigádnické IT oddělení. Pro zajištění relevantního šíření dotazníku přes jednotlivé Chapter Leady bylo použito metody pohodlné vzorkování[24], v kombinaci s metodou sněhové koule[25]. Součástí šíření dotazníku byla také informace, aby se šetření zúčastnili pouze respondenti zabývající se vývojem a dodáním softwaru a nedošlo ke zkreslením výsledků zapojením kolegů z ostatních oddělení.

**Metoda Pohodlného vzorkování**

Jedná se o metodu, která vychází z oslovení respondentů, kteří jsou ochotní se šetření zúčastnit a je jednoduché je kontaktovat. Tato metoda má relativně rychlé a jednoduše dosažitelné výsledky, ale hlavním problémem této metody je nerovnoměrnost rozložení respondentů, a tudíž limituje reprezentativnost získaných výsledků (Lopez, Whitehead, 2013, s. 124).

**Metoda sněhové koule**

Tato metoda spoléhá na oslovení několika jedinců (respondentů), kteří následně oslovují další respondenty a díky tomuto doporučení jsou také ochotni se šetření také zúčastnit a výsledný efekt je přirovnáván ke sněhové kouli. Tato metoda zaručuje velké množství oslovených a relevantních respondentů. Jedná se o podpůrnou metodu, a proto byla využita v kombinaci právě s pohodlným vzorkováním (Lopez, Whitehead, 2013, s. 125).

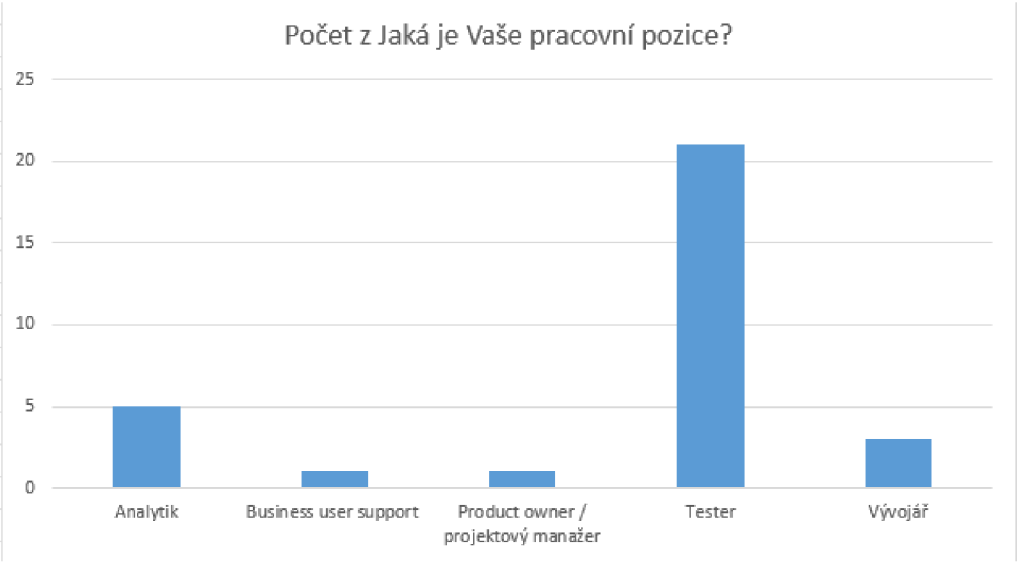
[23] Chapter je skupina sdružující členy jednotlivých Squadů se stejnými zájmy / profesemi.
[24] Z angl. Convinience sampling
[25] Z angl. Snowball sampling

## 4.3 Struktura výsledného souboru

Sběr výzkumných dat probíhal v prostředí organizace, která je blíže představena v podkapitole 4.2.1 Prostředí firmy. Průzkum probíhal 2 týdny a přihlásilo se celkem 31 respondentů z různých agilních týmů napříč celou společností zabývajících se vývojem softwaru. Celkový počet oslovených respondentů nelze určit vzhledem k šíření za pomoci metody sněhové koule.

Prvním demografickým rozdělením dat je rozložení respondentů podle jednotlivých rolí, které zastávají v agilním týmu. Toto rozdělení je zobrazeno na Obrázku 9.

Obrázek 9: Rozdělení jednotlivých rolí respondentů




Zdroj: Autor

Největší zastoupení měla role testera (21 respondentů, 68 %), analytika (5 respondentů, 16 %) a vývojáře (3 respondenti, 10 %). Naopak nejmenší zastoupení měla role Product Ownera (1 respondent), takovýto výsledek se, vzhledem ke stavbě agilních týmů, dal očekávat. Dále je zde jeden respondent zastupující roli Business User Support, což v kontextu organizace je pozice blízko Product Ownerovi. Jako další demografické rozdělení můžeme použít rozdělení jednotlivých rolí ve vztahu k nejvyššímu dosaženému vzdělání. Středoškolské vzdělání má 15 respondentů a vysokoškolské bakalářské má 16 respondentů. Toto rozdělení je vidět v Tabulce 2.

Tabulka 2: Role respondentů ve vztahu k nejvyššímu dosaženému vzdělání

| Vzdělání / Role | Tester | Analytik | Vývojář | Product Owner | Jiné |
|---|---|---|---|---|---|
| Základní | 0 | - | - | - | - |
| Středoškolské | 14 | 1 | - | - | - |
| Vysokoškolské – bakalářské | 7 | 4 | 3 | 1 | 1 |
| Vysokoškolské – magisterské | - | - | - | - | - |
| Vysokoškolské – doktorské | - | - | - | - | - |
| Celkem | 21 | 5 | 3 | 1 | 1 |

Zdroj: Autor

Z tabulky je jasně viditelné, že nejnižší dosažené vzdělání mají respondenti s rolí testera. To je dáno tím, že tato role je v mnoha organizacích často považována jako vstupní do oblasti IT a je tedy obsazována brigádně vysokoškolskými studenty.

Dalším demografickým porovnáním je vztah mezi jednotlivými rolemi a délkou zaměstnání v dané organizaci. Toto rozdělení je reprezentováno v Tabulce 3.

Tabulka 3: Role respondentů ve vztahu k délce zaměstnání u organizace

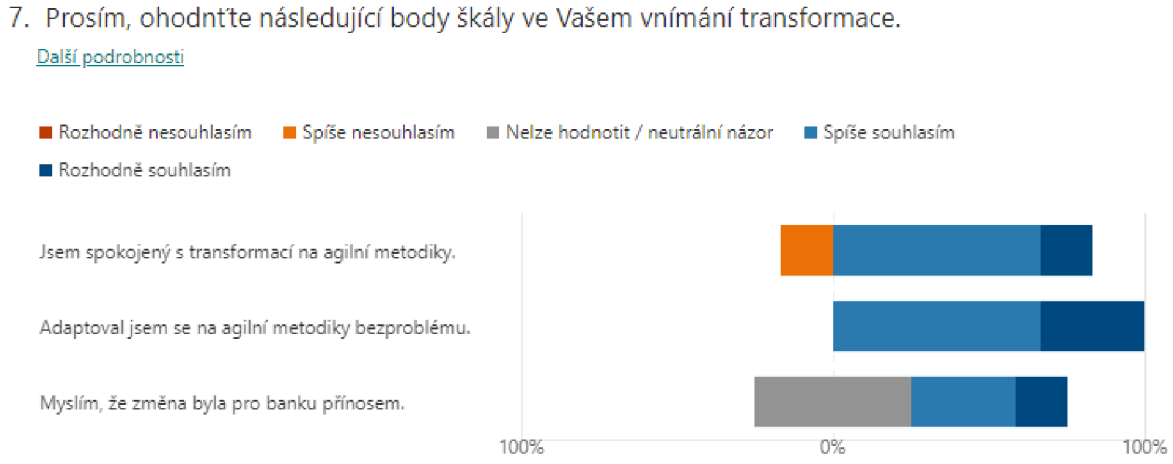
| Délka zaměstnání / Role | Tester | Analytik | Vývojář | Product Owner | Jiné |
|---|---|---|---|---|---|
| Méně jak jeden rok | 13 | 3 | - | - | - |
| 1–2 roky | 6 | 1 | 1 | - | - |
| 2–4 let | 2 | 1 | 1 | 1 | - |
| Více jak čtyři roky | - | - | 1 | - | 1 |
| Celkem | 21 | 5 | 3 | 1 | 1 |

Zdroj: Autor

Z tohoto rozdělení je patrné, že nejvíce respondentů je zaměstnáno kratší dobu než jeden rok, což koresponduje s tím, že nejvíce testerů v organizaci působí na brigádnických pozicích během svých vysokoškolských studií. Toto tvrzení dále potvrzuje i výzkumná otázka, zda je aktuální zaměstnání respondentovo první v oblasti agilních metodik. Z celkových 31 respondentů odpovědělo 30 na tuto otázku „Ano".

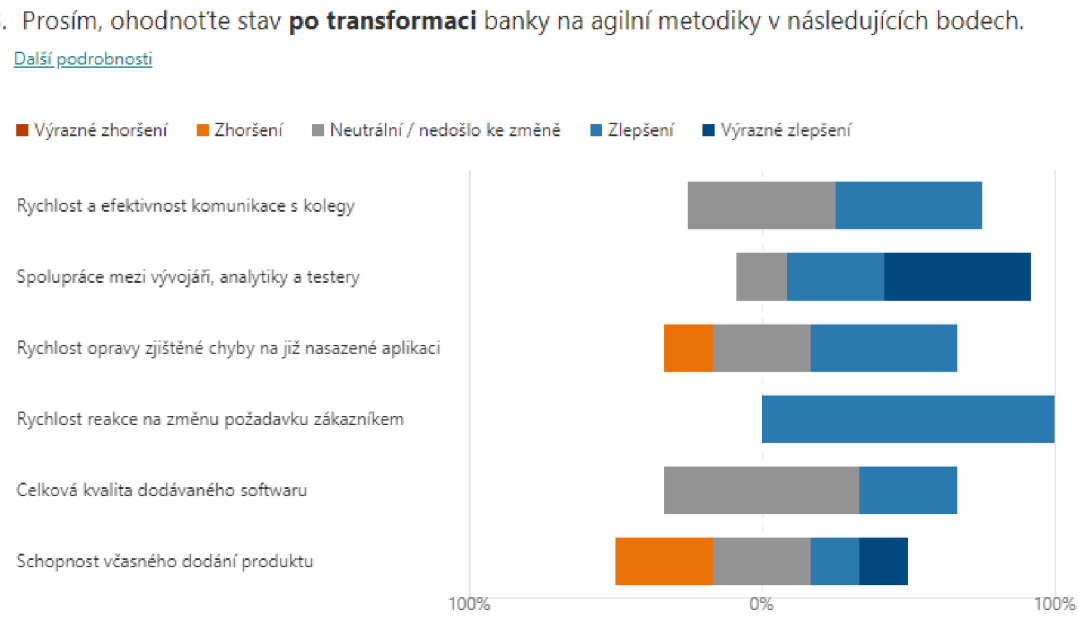
Dále je pro výzkumný soubor důležité, zda respondent byl zaměstnancem již před transformací na agilní metodiky. Zde z 31 respondentů pouze 6 odpovědělo, že byly zaměstnanci před transformací. Obrázek 10 reprezentuje rozdělení hodnotící škály otázek v závislosti na vnímání transformace a Obrázek 11 reprezentuje fungování týmu v porovnání před a po transformaci. Pozitivně hodnotí transformaci 83,4 % respondentů a u všech respondentů došlo k víceméně bezproblémovému přizpůsobení se agilním metodikám. V případě však celkového přínosu pro organizaci hodnotí pouze 50 % respondentů transformaci pozitivně.

Obrázek 10:Vnímání transformace respondenty




Zdroj: Autor

Obrázek 11: Fungování týmu pro transformaci organizace




Zdroj: Autor

Po transformaci hodnotí 50 % respondentů, že došlo ke zlepšení komunikace s kolegy, 50 % uvádí neutrální odpověď. V případě spolupráce mezi jednotlivými členy uvádí zlepšení 83,3 % respondentů, 16,7 % uvádí neutrální odpověď. U otázky rychlosti opravy zjištěné chyby na již nasazené aplikaci uvádí 50 % respondentů zlepšení, 33,3 % respondentů uvádí neutrální hodnocení a 16,7 % respondentů uvádí, že došlo ke zhoršení. Rychlost reakce na změnu požadavku zákazníkem hodnotí 100 % respondentů pozitivně, tj. došlo k jasnému zlepšení. V případě celkové kvality dodávaného softwaru došlo dle 33,3 % respondentů ke zlepšení, 66,7 % uvádí neutrální odpověď. Nejvíce se výsledky liší u otázky ohledně schopnosti včasného dodání produktu, kde zlepšení uvádí pouze 33,4 % respondentů, 33,3 % uvádí neutrální odpověď a 33,3 % respondentů uvádí zhoršení. Tento názor má i Expert 1 v rámci expertních rozhovorů, který tento problém vysvětluje jako stále probíhající adaptace týmů na krátké sprinty, kdy je problematické odhadnout náročnost úkolů, a to vede k občasnému pozdržení dodávky.

## 4.4 Expertní rozhovory s odborníky z praxe

Pro ověření navržených principů byly uskutečněny čtyři kvalitativní rozhovory odborníků z praxe, kteří souhlasili s poskytnutím rozhovoru. Každý z rozhovorů trval 45-75 minut a otázky byly účastníkům poskytnuty dopředu. Základní struktura otázek je uvedena v následujícím výčtu, celé znění otázek je uvedeno v Příloze 3:

1. Průběh sprintu, informace o týmu.
2. Zkušenost před/po transformaci organizace na agilní metodiky.
3. Používané agilní metody.
4. Přístup k testování v týmu.
5. Dovednosti agilního testera.
6. Komunikační kanály týmu.
7. Podpůrné nástroje/činnosti k práci týmu.
8. Zavedené metriky pro hodnocení sprintu.
9. Diskuze k principům.

### 4.4.1 Představení expertů

Z důvodu nutné anonymizace není možné uvádět jména expertů nebo jejich vazby na společnost, a proto jsou v této práci uvedeni pod označením „Expert“.

**Expert 1**

Expert 1 má také inženýrské vzdělání v oboru informatika na České zemědělské univerzitě a v oblasti testingu se pohybuje již více jak 6 let, v posledních 3 letech jako senior testing expert a v posledních 2 letech také jako Chapter Lead a vede jedenácti členný tým.

**Expert 2**

Expert 2 má inženýrské vzdělání z Karlovy Univerzity v oboru IT a v této oblasti se pohybuje již více jak 12 let. V organizaci, kde je aktuálně zaměstnán, byl nejprve v roli projektového manažera a seniorního vývojáře, v posledních 3 letech je na pozici Area Lead pro vývoj.

**Expert 3**

Expert 3 se pohybuje v oblasti IT již 12 let a celý čas se věnuje testingu. Začínal nejprve jako tester, poté jako senior tester a test manažer. Po transformaci organizace na agilní metodiky je v posledních 3 letech na pozici Chapter Lead.

**Expert 4**

Expert 4 se pohybuje v oblasti IT necelé 3 roky a začínal brigádně jako tester-junior při studiu vysoké školy. Poslední rok pracuje jako test manažer a zároveň z brigádnické pozice přešel na plný úvazek. Do organizace, kde je zaměstnán, nastupoval krátce po transformaci na agilní přístup.

### 4.4.2 Struktura rozhovoru a provedení

Sběr dat byl proveden za pomoci polo-strukturovaných rozhovorů a účastníci obdrželi sadu otázek (Příloha 3) v předstihu k nastudování. Rozhovory probíhali osobně, případně online na platformě MS Teams.

Začátek každého provedeného rozhovoru se zabýval obecným seznámením experta se zkoumaným tématem a případnými otázkami ze strany experta. Následně byli experti požádání o udělení souhlasu k pořízení audio nebo audio-vizuálního záznamu, které sloužili k dodatečnému studiu probraných témat a k výstupům pro tuto práci. Z důvodu nutné anonymity jednotlivých expertů nebo organizací nejsou kompletní přepisy těchto rozhovorů součástí příloh této práce.

### 4.4.3 Výsledky kvalitativních rozhovorů

V této podkapitole práce shrnuje výstupy jednotlivých expertů v otázkách 2,4,5 a 8 (Příloha 3). Tyto otázky byly vybrány z důvodů největší relevantnosti pro tuto práci.

**Otázka 2 – Jaká je Vaše zkušenost před a po transformaci Vaší organizace na agilní metodiky? Hodnotíte tuto změnu kladně nebo záporně?**

*„Předtím* (před transformací, pozn.aut.) *to byl úplný chaos, byly hrozně velké releasy a hlavně v jiné části organizace, na kterou navazujeme, byl agile od začátku, zatímco tady byl vodopádový způsob, tedy jednou za 3 měsíce velké releasy ve velkých balících které na sebe divně navazovali. Teď* (po transformaci, pozn.aut.) *se zlepšila efektivita práce, protože ta práce přichází postupně a není to tak jako v těch nárazových balících. Je i větší transparentnost o tom, co kdo dělá nebo nedělá a ta práce se rozdělí, aby všichni měli stejně.* (…) *Zajímavý je i pohled na počet defektů, teď* (po transformaci, pozn.aut.) *je jich dokonce víc, protože se to dodává po částech, ale zase jsou rychleji opravované. Protože předtím tam třeba byl jeden nebo dva defekty, že to nefunguje vůbec a pak se 14 dní čekalo, než se to opraví. Ale když teď vystavím defekt, že něco nefunguje, tak to je opravený prakticky hned"* (Expert 1, 2022).

*„Transformace, i když proběhla před více jak dvěma roky, stále běží, nemá jasný začátek a konec a nelze říct „Tak a teď jsme agilní". Celý proces transformace vidím v přivlastnění myšlenek vycházejících z agilního manifestu a jejich aplikace do všech úrovní organizace, ne snaha aplikace známé metodiky – ta ve většině případů vede k neúspěšnému pokusu o transformaci.* (…) *Za první největší změnu po transformaci považuji vnímání uvnitř organizace a realizace, že kdykoli se mohou změnit jak vnitřní, tak vnější podmínky, a to nás nutí k mnohem flexibilnější revalidaci cílů, potřeb a strategie. Jako druhou považuji posun autonomie na jednotlivé týmy až jednotlivce a tím zanikající mikromanagement, který se transformuje na leadership"* (Expert 2, 2022).

*„Před transformací byly centrálně řízení testeři a hlavní focus byl na to, aby fungovalo testování end-to-end napříč celou organizací. Byli jasně nastavené procesy, které byly nastavený tomu, že se releasovalo čtyřikrát do roka. (…) Po transformaci, co bych viděl jako největší výhodu, že došlo ke spojení businessu a IT – máme CJ, PO, vývojáře a analytiky. Máš tu dodávku díky tomuhle víc pod kontrolou, není to jiný tým – jiný šéf, ale všichni sedí u sebe a můžou se o tom bavit na denní bázi. To je největší posun, že všichni, kteří k tomu mají co říct, se koncentrují do nějaké jednotky"* (Expert 3, 2022).

**Otázka 4 – Jak se přistupuje k testování ve Vašem týmu?**

*„My to máme hodně rozdělené po funkčnostech, co jsou u nás v produktu. Takže to první rozdělení, když přijde balík na otestování, proběhne automaticky podle toho, na co ty lidi mají expertízu. Náš produkt je docela velký, takže úplně neumí všichni všechno. Sice to zvládnou otestovat, ale ne do takového detailu, jako ten člověk, co se na to zaměřuje. A pokud tam je nějaký problém, že by ten člověk nestíhal, buď ty nové věci, co přijdou nebo regresní test toho starého, tak potom se pobavíme o tom, kdo by na to měl čas“* (Expert 1, 2022).

*„Testování napříč celou organizací se posouvá z časově náročného čistě manuálního end-to-end testování*[26] *proti dokumentaci k využití automatizace komponentového testování. Takové testování nám umožňuje, pokud dodržujeme nastavené standardy, mnohem rychlejší výstupy a znovupoužitelné testy se stejnou vypovídající hodnotou jako v případě end-to-end testování. Tohoto stavu ale ještě není možné zcela dosáhnout, a tak se přistupuje alespoň k automatizaci end-to-end testování proti strojově čitelné dokumentaci s využitím nástrojů jako Robot Framework. Nenahraditelné je využití expertízy testerů pro testování bezpečnosti (např. penetrační testování) a průzkumného testování“* (Expert 2, 2022).

*„Tester je nedílnou součástí týmu (…) a účastní se všech ceremonií a všechny jejich úkony jsou zaznamenané v Jire*[27]*, takže když něco dělají, tak tam mají task (…), vlastně všechny aktivity jsou zhmotněný v tomhle ticketovacím nástroji“* (Expert 3, 2022).

*„U nás to funguje tak, že vždycky během sprintu jsou daní dva lidé, které určíme na Retrospektivě*[28] *a ti jsou zodpovědní za napsání systémových testů k jednotlivým Story a navzájem si testy validovali (…). A pak to máme nastavené tak, že k psaní jednotlivých Test casů dochází v jakékoli fázi požadavku, ale validace se dělá až se dodá technická specifikace. Dřív se validace dělala podle slovní specifikace, ještě, než to bylo vyvinuté, ale zjistili jsme, že nám to tak nevyhovuje, protože máme více jak 20 produktů a je těžké bez technické specifikace na něco nezapomenout“* (Expert 4, 2022).

[26] V této práci také jako systémové testování.
[27] Jira je nástroj pro evidenci chyb vyvinutý firmou Atlassian.
[28] Retrospektiva je schůzka celého týmu na konci Sprintu, kde se probírá celý jeho průběh a hodnotí se.

**Otázka 5 – Jaké dovednosti by podle Vás měl mít dobrý tester v agilním týmu?**

*„Ty technické znalosti se určitě hodí, ale nejdůležitější je podle mě mít trochu analytické myšlení, aby byla možnost se pobavit hlavně s tím analytikem, jak to má fungovat, a hlavně třeba mu dávat i zpětnou vazbu, že to takhle nepůjde. Stává se často, že lidi něco navrhnou, ty lidi to prostě udělají, prostě nad tím nepřemýšlí a potom, když ten tester k tomu jde a zjistí, že to takhle bude dělat ten klient, tak se z toho zblázní (…) takže schopnost dát tuhle zpětnou vazbu je podle mě důležitá a umět říct, jak by se to dalo zlepšit a předělat. Takže pokud se nebavíme o těch hard skillech jako umět SQL a rozumět architektuře, tak ta komunikace, a to analytický myšlení je důležitý“* (Expert 1, 2022).

*„Myslím si, že nejdůležitější jsou v oblasti soft skills tři roviny, zaprvé tester by měl mít velmi dobrou znalost domény* (myšleno doménou produktu, pozn.aut.)*, ve které se pohybuje. Zadruhé interakci zákazníka s produktem, jak zákazník produkt využívá a jak se mu s ním pracuje a za třetí dobré analytické myšlení, pečlivost a být šťoura – neustále myslet na hraniční scénáře a možnosti a prozkoumávat je. V případě hard skills jde hlavně o posun k vývoji, kde by měl být schopný alespoň číst v kódu kvůli spolupráci s vývojáři a ideálně umět kódovat kvůli vývoji automatizovaných testů“* (Expert 2, 2022).

*„Ideálním testerem v agilním týmu je někdo, kdo dokáže kombinovat skilly Test manažera, (tahle pozice u nás už oficiálně není), testera a zvládá test automatizaci. (…) Takže musí být schopný definovat nějakou test strategii – musí dobře odhadnout pracnost, musí zajistit dostupnost testovacího prostředku a když je to změna, která má dopady i do jiných aplikací a ten squad iniciuje tu změnu, tak je vhodný, aby zajistil i end-to-end testy, popřípadě UAT*[29] *testy. (…) A pak samozřejmě i test automatizace, která je asi poslední dva roky nedílnou součástí práce testera“* (Expert 3, 2022).

*„Úplně nejzákladnější věc, kterou musí mít je komunikace. SQL, UML a další hard skills jsou určitě potřeba, ale ta komunikace je nedůležitější, aby když něčemu nerozumí se mohl zeptat a vykomunikovat to s analytikem nebo vývojářem. Dále také čtení specifikace, protože tomu musí rozumět a umět číst UML diagramy a jednotlivé aktivity co se tam děje […]. Důležitá je také administrativa, nesmí být líný evidovat, co dělá, jakmile to nebude evidované, tak se v tom už nikdo další nevyzná. A také pomáhá analytické myšlení, je dobré*

[29] User Acceptance Testing

*si umět přečíst a představit si, co všechno ta aplikace má dělat, hodně to usnadní práci"* (Expert 4, 2022).)

**Otázka 8 – Máte zavedené nějaké metriky pro hodnocení úspěšnosti sprintu/releasu? Případně jak takovéto hodnocení děláte?**

*„My úplně vyhodnocení toho sprintu nemáme, nám se to povede, když to stihneme a nepovede, když to nestihneme. Když to nestihneme, tak vyhodnocujeme, proč jsme to nestihli, co se stalo. Ale pokud to stihneme, tak už na tom nějak víc nepracujeme, maximálně během toho sprintu na stand-upech, pokud jsou tam problémy nebo něco trvá fakt dlouho, tak se na to pak zaměřujeme, co se tam děje, proč to trvá tak dlouho a jestli by nestálo za to, to nějak poměnit. Ale vyloženě vyhodnocení sprintu neděláme"* (Expert 1, 2022).

*„My úspěšnost sprintu hodnotíme podle toho, jestli se nám podařilo dokončit to, co jsme si v rámci sprintu naplánovali. Výstupy prezentujeme pravidelně na review*[30]*, kde jsou stake holders (…) a říkáme, co jsme si naplánovali a co jsme zvládli. Ukazujeme tam i živé ukázky."* (Expert 3, 2022).

*„Fungujeme tak, že během Groomingu*[31] *a plánování ohodnocujeme jednotlivé Story a víme, že maximálně stihneme Story ohodnocenou 21, pokud Story ohodnotíme víc, tak to raději rozdělíme do dvou sprintů, jinak to nejsme schopni dodat. Ten agilní způsob by měl být právě o tom, že co naplánuješ, to dodáš – neměl bys nic přidávat ani přetahovat. Zatím se nám to tolik nedaří, takže máme cíl, že nám může zbýt maximálně 20 % a to se nám poslední 2 sprinty povedlo, takže se zlepšujeme"* (Expert 4, 2022).

[30] Ceremonie velmi podobná Retrospektivě.
[31] Proces, při kterém Product Owner zajišťuje aktuálnost požadavků pro vývojový tým.

# 5 Výsledky a diskuze

## 5.1 Vyhodnocení výzkumných otázek

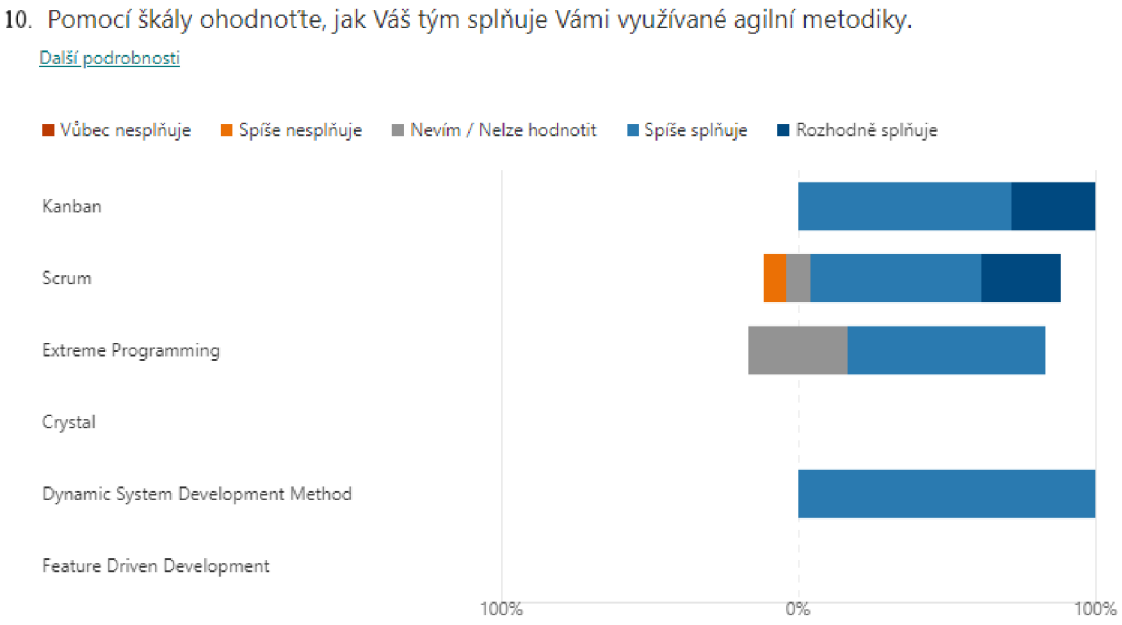
V rámci této podkapitoly jsou vyhodnocovány výzkumné otázky 9-14 uvedené v Příloze 2.

**Používané agilní metodiky**

Tato otázka se zaměřuje na agilní metodiky, které daný tým využívá. Výsledky výzkumu v této práci korelují s výzkumem společnosti Digital (2021) *15th Annual State of Agile Report,* který uvádí, že nejčastější agilní metodikou je metoda Scrum, druhou nejpoužívanější metodikou je Kanban. Metodu Scrum využívá 26 respondentů, metodu Kanban 14 respondentů a dále 6 respondentů využívá metodu Extreme Programming a pouze 1 respondent uvedl i metodu Dynamic Systém Development Method. Zbylé dvě metody – Crystal a Feature Driven Development, nikdo z respondentů neuvedl.

V návaznosti na používané metody měli respondenti ohodnotit, do jaké míry týmem zvolené metody splňují. Reprezentace těchto dat je zobrazena na Obrázku 12. Z výstupu lze vidět, že se týmům daří naplňovat jimi zvolené metody, jelikož u metody Kanban uvedlo 71,4 % respondentů odpověď spíše splňuje a 28,6 % odpověď rozhodně splňuje. V případě metody Scrum uvedlo 57,7 % respondentů odpověď spíše splňuje, 26,9 % rozhodně splňuje, 7,7 % neutrální odpověď a 7,7 % spíše nesplňuje. U metody Extreme Programming uvedlo 66,7 % respondentů spíše splňuje a 33,3 % neutrální odpověď. U metody Dynamic Systém Development Method uvedlo 100 % respondentů spíše splňuje.

Obrázek 12: Splnění zvolených metodik týmy




Zdroj: Autor

V případě hodnocení spíše nesplňuje u metody Scrum respondent v rámci otázky 11 (Příloha 2) uvedl, že v jeho týmu vznikla vlastní metoda inspirována metodou Scrum a tudíž se nepovedlo její přijetí jako takové.

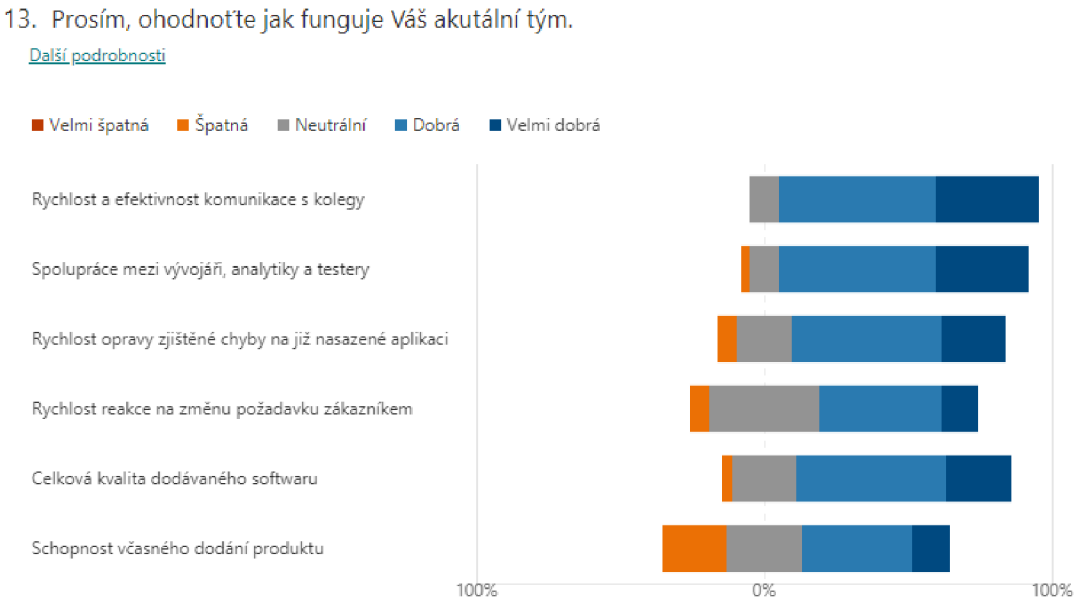
Z výsledků odpovědí na Obrázku 12 vidíme, že nejproblematičtější metodou je metoda Extreme Programming, kde 33,3 % respondentů nedokázalo odpovědět, zda i přes vybrání této metody týmem se daří její implementace.

**Fungování týmu v agilních metodikách**

Obrázek 13 reprezentuje data obecného charakteru fungování týmů, kterých jsou respondenti součástí. Reprezentovaná data ukazují, že týmy dotázaných respondentů fungují velmi dobře, 90,3 % respondentů uvedlo komunikaci s kolegy jako dobrou či velmi dobrou, 87,1 % respondentů uvedlo spolupráci v týmu jako dobrou nebo velmi dobrou a pouze 3,2 % respondentů označilo spolupráci v týmu jako špatnou. Dále 74,2 % respondentů uvedlo, že rychlost opravy zjištěné chyby na nasazeném produktu je dobrá nebo velmi dobrá, 19,4 % respondentů uvedlo neutrální odpověď a 6,5 % respondentů uvedlo rychlost opravy jako špatnou. V případě rychlosti reakce na změnu požadavku od zákazníka uvedlo 54,8 % respondentů dobré nebo velmi dobré hodnocení a pouze 6,5 % respondentů uvedlo špatné hodnocení. Neutrální hodnocení uvedlo 38,7 % respondentů. Celkovou kvalitu dodávaného

softwaru hodnotí 74,2 % respondentů jako dobrou nebo velmi dobrou, 22,6 % respondentů uvedlo neutrální odpověď a pouze 3,2 % respondentů uvedlo celkovou kvalitu jako špatnou. Největší rozdíl byl u otázky „Schopnost včasného dodání produktu“, kdy uvedlo dobrou či velmi dobrou zkušenost pouze 51,6 % respondentů, neutrální odpověď uvedlo 25,8 % respondentů a špatnou zkušenost 22,6 % respondentů. Tuto odchylku vysvětluje Expert 1 v expertním rozhovoru a je rozepsána v podkapitole 4.4.3 Výsledky kvalitativních rozhovorů.

Obrázek 13: Celkové fungování týmu z pohledu respondentů

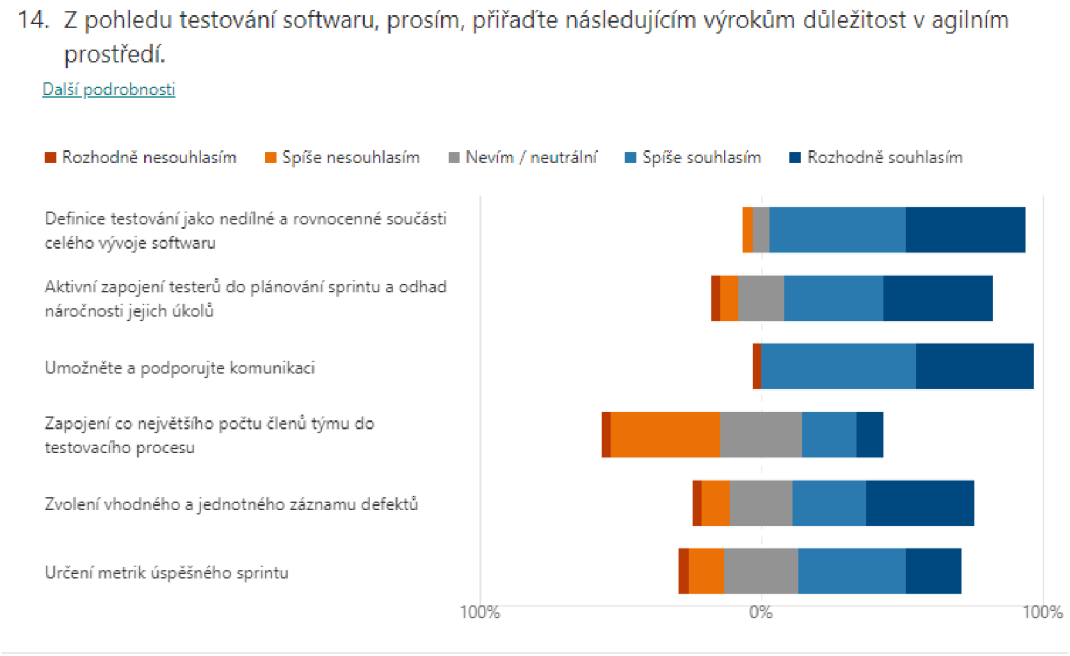



Zdroj: Autor

## 5.2 Vyhodnocení výzkumné otázky – definované principy

Poslední výzkumná otázka se zaměřila na hodnocení definovaných principů touto prací. Hodnocení respondenti provedli pomocí hodnotící škály a reprezentace výsledků je na Obrázku 14.

Obrázek 14: Hodnocení definovaných principů respondenty




Zdroj: Autor

Data korelují s výsledky expertních rozhovorů v kapitole 4.4 Expertní rozhovory s odborníky z praxe a jako nejdůležitější se jeví 1., 2. a 3. princip.
U principu 1 uvedlo 90,3 % respondentů spíše souhlasím nebo rozhodně souhlasím, 6,5 % respondentů uvedlo neutrální odpověď a pouze 3,2 % respondentů uvedlo spíše nesouhlasím.

U principu 2 uvedlo 74,2 % respondentů spíše souhlasím nebo rozhodně souhlasím, 16,1 % respondentů uvedlo neutrální odpověď, 6,5 % respondentů uvedlo odpověď spíše nesouhlasím a 3,2 % respondentů uvedlo rozhodně nesouhlasím. Jednalo se o respondenta v roli testera a s délkou zaměstnání pod jeden rok.

U principu 3 uvedlo 96,7 % respondentů hodnocení spíše souhlasím nebo rozhodně souhlasím. Respondent, který uvedl hodnocení rozhodně nesouhlasím, byl v roli testera a

délka zaměstnání byla kratší než jeden rok, jedná se však o jiného respondenta než v případě principu 2. Až na tuto anomálii můžeme tedy potvrdit, že komunikace je v agilních týmech velmi důležitá. Toto potvrzují i expertní rozhovory v kapitole 4.4 Expertní rozhovory s odborníky z praxe.

Stejně jako v případě expertních rozhovorů, nejkontroverznější je princip 4, kde 38,7 % respondentů uvedlo odpověď spíše nesouhlasím a 3,2 % respondentů rozhodně nesouhlasím. Neutrální odpověď uvedlo 29 % respondentů a hodnocení spíše souhlasím nebo rozhodně souhlasím uvedlo pouze 29,1 % respondentů. Experti v rámci rozhovorů to vysvětlují tím, že je nutné zapojit pouze členy, kteří o to mají zájem a také upozorňují, že zapojení většího počtu lidé do procesu může vést k většímu chaosu v týmu až případné paralyzace týmu.

V případě principu 5 uvedlo 64,5 % respondentů odpověď spíše souhlasím a rozhodně souhlasím, 22,6 % respondentů uvedlo neutrální odpověď, 9,7 % respondentů uvedlo spíše nesouhlasím a 3,2 % respondentů uvedlo rozhodně nesouhlasím.

U principu 6 uvádí 58,1 % respondentů odpověď spíše souhlasím a rozhodně souhlasím. 25,8 % uvedlo neutrální odpověď, 12,9 % respondentů uvedlo spíše nesouhlasím a 3,2 % respondentů uvedlo rozhodně nesouhlasím.

## 5.3 Výsledky kvalitativních rozhovorů – hodnocení jednotlivých principů

Expert 1 jako nejdůležitější uvedl princip 1 a 3. Naopak u principu 4 zmiňuje, že nemá dobré zkušenosti se zapojením celého týmu, hlavně u vývojářů: *„Můj názor je, že vývojáři jsou špatní testeři, protože buď to po sobě neotestují pořádně, protože si hodně věří, anebo naopak si myslí, že dokážou po někom kód přečíst tak dobře, že samotné testování provádí jen letmo. Dobré spojení je dvojice analytik a tester a tam je ta komunikace rozhodně důležitá“* (Expert 1, 2022).

Expert 2 jako nejdůležitější vyhodnotil principy 1, 2, 3. U principu 1 zdůrazňuje, že je důležité nejen testování brát jako nedílnou a rovnocennou součást, ale také zajistit kontinuální testování v průběhu celého sprintu. U principu 2 zdůrazňuje, že je zapojení testerů důležité zejména v definici akceptačních kritérií a také komplexního pohledu na

náročnost jednotlivých úkolů. Není možné náročnost určovat pouze z pohledu testingu nebo vývoje (nebo z pohledu jiných jednotlivých profesí) a za ideální považuje stav, kdy tým, díky předchozím zkušenostem, vytvoří modelové případy, podle kterých je určena náročnost. U principu 3 zdůrazňuje, že komunikace je základ úspěšného fungování v agilním týmu, ale je nutné neaplikovat přístup „*všichni budou u všeho*“, tj. ne všichni členové musí být na všech schůzkách, jelikož to může vést k chaosu a paralyzaci týmu. Dále můžeme zmínit princip 5, kde Expert 2 zastává názor, že je nutné, aby se zvolený způsob záznamu chyb neopíral pouze o BTT, ale apeluje na mezilidskou interakci, jelikož BTT svádí k izolaci testerů. Odkazuje se na jednu z hlavních myšlenek Agilního manifestu „*Jednotlivci a interakce před procesy a nástroji*“.

Expert 3 označil jako nejdůležitější principy 1 a 2. Uvádí, že bez těchto principů nelze agilní testování efektivně provádět, ale i přesto se v organizaci, ve které působí, setkává s názory, že testeři nejsou potřební a jejich roli zastanou analytici. *„Poprvé, když jsem si to přečetl (Princip 1, pozn. aut.), jsem si říkal, že je to přece jasný, že je potřeba mít testing jako nedílnou součást životního cyklu softwaru, ale pak mi zpětně došlo, že někdo to prostě nepovažoval za prioritu (…) a během transformace jsem se setkával s názory, že jejich tým testery nepotřebuje. To pro mě bylo překvapení“* (Expert 3, 2022). Naopak za méně důležitější považuje princip 4, jelikož je zastáncem toho, že by mohl ve většině případů být kontraproduktivní. Začlenil by pouze členy, kteří o to mají evidentní zájem, a hlavně jsou přínosem pro testovací proces.

Expert 4 uvedl jako nejdůležitější principy 1, 3, 5. *„Určitě za nejdůležitější považuji komunikaci v týmu a testování jako nedílnou součást celého vývoje softwaru. Testování je opravdu velice důležité a stejně tak QA*[32] *je základní věc. Pokud to pořádně neotestuješ, budeš mít do budoucna mnohem více práce, protože je lepší tyhle chyby a nedodělávky vychytat v procesu vývoje než potom na produkci. Je jednodušší případně dodávku i zrušit než to neotestovat, nasadit, a klient pak zjistí, že to vůbec nefunguje“* (Expert 4, 2022).

[32] Quality Assurance

# 6 Závěr

Celý agilní přístup se opírá o členy agilních týmů a jejich schopnost komunikace, nejedná se pouze o přístup k vývoji softwaru, ale o celkový mindset daného týmu. Jakmile jeden z členů týmu nezvládne transformaci na agilní metodiky nebo nepochopí jejich základní principy, může to vést ke zhroucení celého procesu vývoje. I proto byly agilní metodiky tak dlouho doménou pouze malých organizací nebo jednotlivých týmů. Avšak aktuální doba, neustále se měnící potřeby společnosti a technologický pokrok nutí i velké organizace změnit jejich přístup k vývoji softwaru a implementovat agilní metodiky. Pokud tak neučiní, je možné, že by postupem času ztráceli na konkurenceschopnosti a atraktivitě pro zákazníky.

Agilní metodiky s sebou nesou zcela nový pohled na potřebné znalosti a dovednosti jednotlivých členů týmu. Klade se důraz na komplexnost znalostí všech potřebných oblastí, které tým ke svému fungování potřebuje. A také představuje zcela nový pohled na vnímání role zákazníka, a to v celém procesu vývoje. V agilních metodikách se zákazník dostává do středu vývojového procesu a je nezbytnou součástí pro jeho úspěšnost. Je proto nutné tuto roli zákazníkovi dostatečně vysvětlit, aby chápal její důležitost.

Práce čtenáři nejprve představuje Agilní manifest, ze kterého všechny agilní metody vychází. V následující kapitole jsou popsány základní principy fungování agilních metodik a základní metody agilního vývoje. Poté se čtenář seznámí s přístupem tradičního způsobu vývoje softwaru, který mu poskytne komplexní pohled na rozdíly mezi těmito přístupy. Další část práce vymezuje pojem testování softwaru jako takového, důležitost testování a obecné způsoby a přístupy k testování softwaru. Následně práce představuje přístupy k testování, které jsou specifické pro agilní metodiky a v závěru teoretické části práce shrnuje rozdíly přístupů k testování v agilním a tradičním vývoji softwaru.

Praktická část práce měla za cíl navrhnutí principů, které mohou v praxi pomoci ke zlepšení testování softwaru v agilních metodikách, ať už pro týmy, které agilní metodiky využívají, nebo pro týmy, které o jejich zavedení uvažují. Tyto principy byly následně ověřeny v praxi za pomoci expertních rozhovorů a dotazníkového šetření. Jako dílčí cíl bylo, v rámci obou výzkumných metod, provedeno hodnocení fungování agilních týmů v organizaci, ve které v nedávné době došlo k transformaci na agilní metodiky.

Celkovým přínosem této práce je vytvoření uceleného pohledu na agilní a tradiční vývoj softwaru a poukázání na rozdíly mezi nimi, které mohou sloužit testerům i ostatním členům týmu pro jejich pochopení a snazší transformaci na agilní metodiky. Principy definované v této práci následně mohou zajistit nejen lepší proces transformace, ale i následnou spolupráci všech členů týmu.

Možnosti rozšíření tématu této bakalářské práce mohou být například dlouhodobým pozorováním týmů, které již agilní metodiky využívají, nebo skrze hlubší analýzu uskupení, které se na transformaci připravují nebo jí v nedávné době prošli.

# 7 Seznam použitých zdrojů


BADGETT, Tom, MYERS, Glenford J., SANDLER Corey. The art of software testing. 3rd ed. Hoboken, New Jersey: Wiley, 2012. 256 s. ISBN 1118031962.

ALSHAMRANI, Adel, BAHATTAB, Abdullah. A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model IJCSI International Journal of Computer Science Issues [online]. January 2015, vol 12, issue 1. ISSN 1694-0784. Dostupné z https://www.ijcsi.org/papers/IJCSI-12-1-1-106-111.pdf

BUREŠ, Miroslav, Michal DOLEŽEL, Miroslav RENDA, a kolektiv. Efektivní testování softwaru: klíčové otázky pro efektivitu testovacího procesu. Praha: Grada, 2016. 232 s. ISBN 978-80-247-5594-6.

CRISPIN, Lisa, GREGORY Janet. Agile testing: a practical guide for testers and agile teams. New Jersey: Addison-Wesley, 2009. 576 s. ISBN 978-0-321-53446-0.

MUNASSAR, N.M.A., Govardhan, A. A Comparison Between Five Models Of Software Engineering IJCSI International Journal of Computer Science Issues [online]. September 2010, vol 7, issue 5. ISSN 1694-0814. Dostupné na WWW: https://www.ijcsi.org/papers/7-5-94-101.pdf

MARSDEN, Gary, MAUNDER, Andrew, PARKER, Munie. People are people, but technology is not technology. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences [online]. 2008, vol. 366, issue 1881. ISSN 1364-503X. Dostupné na WWW: https://www.researchgate.net/publication/23142609_People_are_people_but_technology

VersionOne. 9th annual State of Agile Survey [online]. 2015. Dostupné na WWW: https://kipdf.com/state-of-agile-survey-9-th-annual_5afcc7c18ead0eb9108b4692.html

Digital. 15th Annual State of Agile Report [online]. 2021. Dostupné na WWW: https://info.digital.ai/rs/981-LQX-968/images/RE-SA-15th-Annual-State-Of-Agile-Report.pdf

HÁK, Tomáš. Testing v agilním prostředí [online]. 2020. Dostupné na WWW: https://www.tesena.com/novinky/testing-v-agilnim-prostredi

Agile manifesto. Manifest Agilního vývoje software [online]. 2001. Dostupné na WWW: https://agilemanifesto.org/iso/cs/manifesto.html

ISTQB. Foundation Level Extension Syllabus Agile Tester [online], 2014. Dostupné na WWW: https://www.istqb.org/downloads/send/5-agile-tester-extension-documents/41-agile-tester-extension-syllabus.html

ANDERSON, David, J. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010. 280 s. ISBN 0984521402.

SCHWABER, Ken, SUTHERLAND, Jeff. The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game [online]. 2020. Dostupné na WWW: https://billlewistraining.com/wp-content/uploads/2017/02/PMP-Agile-Study-Materials.pdf

SCHWABER. Ken. SCRUM Development Process [online]. 1995. Dostupné na WWW: http://damiantgordon.com/Methodologies/Papers/Business%20Object%20Design%20and%20Implementation.pdf

ISTQB. Foundation Level Syllabus [online]. 2019. Dostupné na WWW: https://www.istqb.org/downloads/send/2-foundation-level-documents/281-istqb-ctfl-syllabus-2018-v3-1.html

BROWN, Alexander. Test First approach sounds simple enough, right? [online]. 2020. Dostupné na WWW: https://www.scrum.org/resources/blog/test-first-approach-sounds-simple-enough-right

BOEHM, Barry. Spiral Development: Experience, Principles,and Refinements [online]. 2000. Dostupné na WWW: https://resources.sei.cmu.edu/asset_files/SpecialReport/2000_003_001_13655.pdf

LOPEZ, Violeta, WHITEHEAD, Dean. Nursing and midwifery research: methods and appraisal for evidence-based practice. Chatswood, N:S:W.: Elsevier Austria, ISBN 978-0-7295-4137-4.

CRUTH, Mark. Discover the Spotify model [online]. Nedatováno. Dostupné na WWW: https://www.atlassian.com/agile/agile-at-scale/spotify

EXPERT 1, Chapter Lead. Expertní rozhovor k závěrečné práci. 2022.

EXPERT 2, Area Lead. Expertní rozhovor k závěrečné práci. 2022.

EXPERT 3, Chapter Lead. Expertní rozhovor k závěrečné práci. 2022.

EXPERT 4, Test manažer. Expertní rozhovor k závěrečné práci. 2022.

# Přílohy

## 7.1 Příloha 1: 12 principů Agilního manifestu

1. Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
2. Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
3. Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
4. Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.
5. Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
6. Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
7. Hlavním měřítkem pokroku je fungující software.
8. Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
9. Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
10. Jednoduchost--umění maximalizovat množství nevykonané práce--je klíčová.
11. Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.
12. Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti.

## 7.2 Příloha 2: Otázky dotazníkového šetření

Tabulka 4: Otázky dotazníkového šetření

| Sekce | Typ otázky | Znění otázky | Možnosti odpovědi |
|---|---|---|---|
| Charakteristika respondenta | Uzavřená výběrová | Jaká je vaše pracovní pozice? | Tester; Analytik; Vývojář; Product Owner / projektový manažer; Jiné |
| Charakteristika respondenta | Uzavřená výběrová | Jaké je vaše nejvyšší dosažené vzdělání? | Základní; Středoškolské; Vysokoškolské – bakalářské; Vysokoškolské – magisterské; Vysokoškolské – doktorské |
| Charakteristika respondenta | Uzavřená výběrová | Jak dlouho jste zaměstnání v organizaci? | Méně jak jeden rok; 1-2 roky; 2-4 let; Více jak čtyři roky |
| Charakteristika respondenta | Uzavřená výběrová | Je vaše aktuální pozice první v zaměstnání v oblasti IT za použití agilních metodik? | Ano; Ne |
| Charakteristika respondenta | Otevřená dlouhá | Prosím popište rozdíl v přístupu a používaných metodik v předchozím a aktuálním zaměstnání [33]. | |
| Charakteristika respondenta | Uzavřená výběrová | Byl/a jste zaměstnancem před transformací na agilní metodiky? | Ano; Ne |
| Charakteristika respondenta | Hodnotící škála | Prosím, ohodnoťte následující body škály ve Vašem vnímání transformace:<br><br>Jsem spokojený s transformací na agilní metodiky;<br>Adaptoval jsem se na agilní metodiky bezproblému;<br>Myslím, že změna byla pro banku přínosem. [34] | Rozhodně nesouhlasím; Spíše nesouhlasím; Nelze hodnotit / neutrální názor; Spíše souhlasím; Rozhodně souhlasím |
| Charakteristika respondenta | Hodnotící škála | Prosím, ohodnoťte stav po transformaci organizace na agilní metodiky v následujících bodech:<br><br>Rychlost a efektivnost komunikace s kolegy;<br>Spolupráce mezi vývojáři, analytiky a testery;<br>Rychlost opravy zjištěné chyby na již nasazené aplikaci;<br>Rychlost reakce na změnu požadavku zákazníkem; | Rozhodně nesouhlasím; Spíše nesouhlasím; Nelze hodnotit / neutrální názor; Spíše souhlasím; Rozhodně souhlasím |

[33] Tato otázka byla položena pouze pokud respondent uvedl v předchozí otázce odpověď „Ne“.
[34] Tyto otázky byly položeny pouze pokud v otázce „Byl/a jste zaměstnancem před transformací na agilní metodiky?“ respondent odpověděl „Ano“.

| | | | |
|---|---|---|---|
| | | Celková kvalita dodávaného softwaru;<br>Schopnost včasného dodání produktu [35]. | |
| Používané agilní metodiky | Uzavřená vícevýběrová | Prosím, vyjmenujte, které agilní metodiky váš tým používá. | Kanban; Scrum; Extreme Programming (XP); Crystal; Dynamic Systém Development Method (DSDM); Feature Driven Development (FDD) |
| Používané agilní metodiky | Hodnotící škála | Pomocí škály ohodnoťte, jak váš tým splňuje vámi využívané agilní metodiky. [36] | Vůbec nesplňuje; Spíše nesplňuje; Nevím / nelze hodnotit; Spíše splňuje; Rozhodně splňuje |
| Používané agilní metodiky | Otevřená dlouhá | Pokud jste v otázce „Prosím vyjmenujte, které agilní metodiky Váš tým používá.“ Nenašli Vámi využívanou metodiku, prosím o její zapsání zde a slovní ohodnocení jako v otázce „Pomocí škály ohodnoťte, jak Váš tým splňuje Vámi využívané agilní metodiky.“ | |
| Používané agilní metodiky | Otevřená dlouhá | Pokud jste v předchozích dvou otázkách uvedli „Vůbec nesplňuje“, prosím o doplnění, proč se Vašemu tým danou metodiku nedaří naplnit. | |
| Používané agilní metodiky | Hodnotící škála | Prosím, ohodnoťte, jak funguje váš aktuální tým.<br><br>Rychlost a efektivnost komunikace s kolegy;<br>Spolupráce mezi vývojáři, analytiky a testery;<br>Rychlost opravy zjištěné chyby na již nasazené aplikaci;<br>Rychlost reakce na změnu požadavku zákazníkem;<br>Celková kvalita dodávaného softwaru;<br>Schopnost včasného dodání produktu | Velmi špatná; Špatná; Neutrální; Dobrá; Velmi dobrá |
| Hodnocení principů | Hodnotící škála | Principy byly formulovány tak, jako v kapitole 4.1. | Rozhodně nesouhlasím; Spíše nesouhlasím; Nevím / neutrální; Spíše souhlasím; Rozhodně souhlasím |
| Závěrečné otázky | Otevřená dlouhá | Chybělo v dotazníku něco, co byste rád/a doplnil/a? | |

[35] Tato otázka byla položena pouze pokud respondent uvedl v otázce „Byl/a jste zaměstnancem před transformací na agilní metodiky?“ odpověď „Ano“.
[36] Respondenti hodnotili každou zvolenou metodu jednotlivě.

## 7.3 Příloha 3: Otázky k expertním rozhovorům

1. Jak u vás v týmu probíhá sprint, kolik členů tým má a rozdělení jednotlivých rolí.
2. Jaká je vaše zkušenost před a po transformaci banky na agilní metodiky? Hodnotíte tuto změnu kladně/záporně a proč?
3. Jaké agilní metody váš tým využívá? Máte případně zkušenosti i s jinými metodami?
4. Jak se přistupuje k testování ve vašem týmu? Zapojují se testeři aktivně do plánování/diskuze v rámci schůzek?
5. Jaké dovednosti by podle vás měl mít dobrý tester v agilním týmu?
6. Jaké komunikační kanály používáte v týmu?
    a. Jak reportujete jednotlivé úkoly pro testery, výstupy testů apod.
7. Jaké nástroje k testování používáte? Do jaké míry vedete dokumentaci k vašemu produktu/funkcionalitě?
8. Máte zavedené nějaké metriky pro hodnocení úspěšnosti sprintu/releasu? Případně jak takovéto hodnocení děláte?
9. Diskuze k principům:
    a. Který z těchto principů vám přijde nedůležitější?
    b. Může týmům k transformaci na agilní metodiky pomoci tato/obecně sada principů?
    c. Je zde nějaký princip, který vám na základě praxe přijde zbytečný?
    d. Naopak, chybí zde nějaký důležitý princip?