Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Ověření implementace metody Exploratory testování v podnikové praxi

Bc. Martina Bartošová

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Martina Bartošová

Systémové inženýrství a informatika
Informatika

Název práce

Ověření implementace metody Exploratory testování v podnikové praxi

Název anglicky

Verification of the implementation of the Exploratory Testing method in business practice

---

Cíle práce

Diplomová práce se zabývá použitím metody Exploratory testování v testování informačních systémů a aplikací ve středně velké firmě Volkswagen Financial Services CZ. Cílem diplomové práce je vytvoření metodiky pro Exploratory testování, která bude využita v podnikové praxi, a identifikace vhodných systémů či aplikací, kde lze metodu použít. Součástí bude následné ověření implementace v praxi. Dílčím cílem práce je analýza používaných metod testování v konkrétní firmě a porovnání se standardy.

Metodika

První část této práce bude založena na studiu odborných informačních zdrojů zabývajících se problematikou testování softwaru a porovnání jednotlivých metod. V praktické části bude zpracována metodika Exploratory testování a ověření implementace zvolené metody v praxi. Na základě získaných informací z implementace budou formulovány doporučení a závěry diplomové práce.

Doporučený rozsah práce

60 – 80 stran

Klíčová slova

exploratory testing, metody testování, manuální testování, testování softwaru

---

Doporučené zdroje informací

BLACK, Rex. Advanced Software Testing – Vol. 2, 2nd Edition: Guide to the ISTQB Advanced Certification As an Advanced Test Manager. Rocky Nook, 2014. ISBN 1937538508, 9781937538507.

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. Profes!onal. ISBN 978-80-247-5594-6.

KRUG, S. Nenuťte uživatele přemýšlet! : praktický průvodce testováním a opravou chyb použitelnosti webu. Brno: Computer Press, 2010. ISBN 978-80-251-2923-4.

PATTON, Ron. Testování softwaru. Přeložil David KRÁSENSKÝ. Praha: Computer Press, 2002. Programování. Pro každého uživatele. ISBN 80-7226-636-5.

ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. Řízení kvality softwaru: průvodce testováním. Brno: Computer Press, 2013. ISBN 978-80-251-3816-8.

WHITTAKER, James A. Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design. Pearson Education, 2009. ISBN 0321647858, 9780321647856.

Předběžný termín obhajoby

2022/23 ZS – PEF

Vedoucí práce

Ing. Petr Benda, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 9. 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 29. 11. 2022

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Ověření implementace metody Exploratory testování v podnikové praxi" jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

V Praze dne 29. listopadu 2022 _____

Poděkování

Ráda bych touto cestou poděkovala Ing. Petru Bendovi, Ph.D. za cenné podněty, ochotu a trpělivost v průběhu zpracování diplomové práce. Dále bych ráda poděkovala Mgr. Pavle Brunclíkové za veškerou pomoc a poskytnuté pracovní příležitosti při psaní této práce.

Ověření implementace metody Exploratory testování v podnikové praxi

Abstrakt

Exploratory testování je oblíbenou metodou mezi testery díky své proveditelnosti bez předem definovaných testovacích případů. Tato metoda je z části jednotlivými testery často používána, ale není nijak standardizována a obvykle je skryta za scénářové testování. Důvodem pro zavedení této metody ve zvolené společnosti je standardizace této metody a zefektivnění testovacího procesu se zachováním či zvýšením jeho kvality.

Pro pochopení dané problematiky je vypracována literární rešerše práce, která nejdříve představuje oblast zabývající se vývojem softwaru, následně popisuje komplexní oblast testování a na závěr vymezuje oblast Exploratory testování.

Vlastní práce se v první řadě zabývá analýzou testování ve zvolené společnosti. Jsou zde popsány používané aplikace a systémy společnosti, metody testování, role, nástroje a testovací prostředí. Na základě získaných poznatků je v rámci standardizace této metody vytvořena předpokládaná podoba metodiky Exploratory testování, která je následně prakticky ověřena na vybraných aplikacích či systémech. Toto ověření odpovídá na výzkumné otázky ohledně vhodnosti a efektivity Exploratory testování v praxi a ukazuje, zda je navržená podoba metodiky použitelná a vhodná.

Klíčová slova: exploratory testing, metody testování, manuální testování, testování softwaru

Verification of the implementation of the Exploratory Testing method in business practice

Abstract

Exploratory testing is a popular method among testers due to its feasibility without predefined test cases. This method is often used in part by individual testers, but it is not standardized in any way and is usually hidden behind 'scenario testing. The reason for introducing this method in the chosen company is it's means of standardizing and streamlining the testing process while maintaining or improving its quality.

In order to understand the issue at hand, a literature search of the thesis is done, which firstly introduces the area dealing with software development, then describes the complex area of testing, and finally defines the area of Exploratory testing.

The thesis itself is primarily concerned with the analysis of testing in the chosen company. The applications and systems used by the company, testing methods, roles, tools, and testing environment are described. On the basis of the obtained knowledge, an assumed form of Exploratory testing methodology is created within the framework of standardization of this method. This methodology is then practically verified on selected applications or systems. Based on the verified findings, we can answer the research questions regarding the suitability and effectiveness of Exploratory testing in practice and shows whether the proposed methodology is applicable and suitable.

Keywords: exploratory testing, testing methods, manual testing, software testing

Obsah

1 Úvod

V 19. století se začala, s prvním vytvořeným mechanickým počítačem vynálezcem Charlesem Babbagem, psát historie systémových poruch. Termín bug neboli chyba však použil poprvé Thomas Edison již v roce 1878. Začátkem 20. století v důsledku 2. světové války počítačová technologie pokročila a byly poprvé v historii testování softwaru použity termíny počítačová chyba (computer bug) a vychytávání chyb (debugging). Avšak za otce testování je považován Joseph Juran, který poprvé v roce 1951 ve své knize označil důležitost zajištění kvality softwaru. Co se týče moderní historie testování softwaru, v roce 1957 Charles L. Baker konečně odlišil výše zmíněné termíny a vytvořil první testovací tým pro projekt Mercury, jenž byl programem pro pilotované kosmické lety v USA (Prytulenets, 2022).

Během nadcházejících let se testování stalo hlavním nástrojem pro poskytování vysoce kvalitních softwarových řešení a stalo se neoddělitelnou součástí celého procesu vývoje softwaru. Software je všude kolem nás, v oblastech obchodu, lékařství, vzdělávání, vojenství, v telekomunikacích, a proto je důležité, aby byl vyvíjen ve vysoké kvalitě bez závad. Lidé bez zkušeností v oblasti testování tuto činnost vnímají jako pouhé klikání, jedná se však o inženýrskou disciplínu, která je formována a silně ovlivňována projektem a organizací, ve které je vykonávána.

Testování softwaru není izolovaná aktivita na konci procesu vývoje softwaru, jak bylo chápáno v minulosti, ale činnost, která je provázána mnoha dalšími aktivitami prováděnými týmy, jenž mají na starost vývoj softwaru (Bureš, a další, 2016). Proto je důležité na začátek této práce zmínit základní metodiky vývoje softwaru, kterých je testování součástí a dle kterých se mohou metodiky testování v organizaci lišit.

Pro pochopení celé problematiky testování softwaru je důležité popsat, co to je chyba, defekt a co se může stát, pokud nejsou v dostatečném čase odhalené. V diplomové práci budou popsány důležité termíny, role a odpovědnosti potřebné ke správě testování, dále pak dělení testů, úrovně testování a celý management testování až po samotný cíl celé práce, který se týká metody Exploratory testování a její zavedení ve zvolené společnosti.

Ne vždy jsou všechny defekty odhaleny pomocí formálního postupu, jehož součástí jsou testovací scénáře. Velice často je prováděno neformální testování bez scénářů, pouze na základě zkušeností se softwarem. Toto testování se nazývá Exploratory neboli průzkumné testování. Postup Exploratory testování je však správný, jen je potřeba ho jistým způsobem

zformalizovat. Poté se může stát efektivním pomocníkem ke scénářovému testování, jak tomu bude uvedeno v teoretické části práce. V praktické části práce bude na základě získaných znalostí vytvořena metodika Exploratory testování, jenž bude ověřena pomocí několika provedených testování, na jejichž základě budou ověřeny a zhodnoceny jeho možné přínosy do standardního testování zvolené organizace.

2 Cíl práce a metodika

2.1 Cíl práce

Diplomová práce se zabývá použitím metody Exploratory testování v testování informačních systémů a aplikací ve středně velké firmě Volkswagen Financial Services CZ. Cílem diplomové práce je vytvoření metodiky pro Exploratory testování, která bude využita v podnikové praxi. Součástí cíle je identifikace vhodných systémů či aplikací, kde lze metodu použít a ověření její implementace v praxi. Dílčím cílem práce je analýza používaných metod testování ve zvolené společnosti a porovnání se standardy.

2.2 Metodika

První část této práce bude založena na studiu odborných informačních zdrojů zabývajících se problematikou testování softwaru a porovnání jednotlivých metod testování. V praktické části bude nejprve provedena analýza fungování IT, konkrétně pak testování ve zvolené společnosti. Na základě získaných znalostí bude zpracována metodika Exploratory testování, budou vytipovány vhodné aplikace či systémy a bude provedeno ověření implementace zvolené metody v praxi. Na základě získaných informací z implementace budou formulovány doporučení a závěry diplomové práce.

3 Teoretická východiska

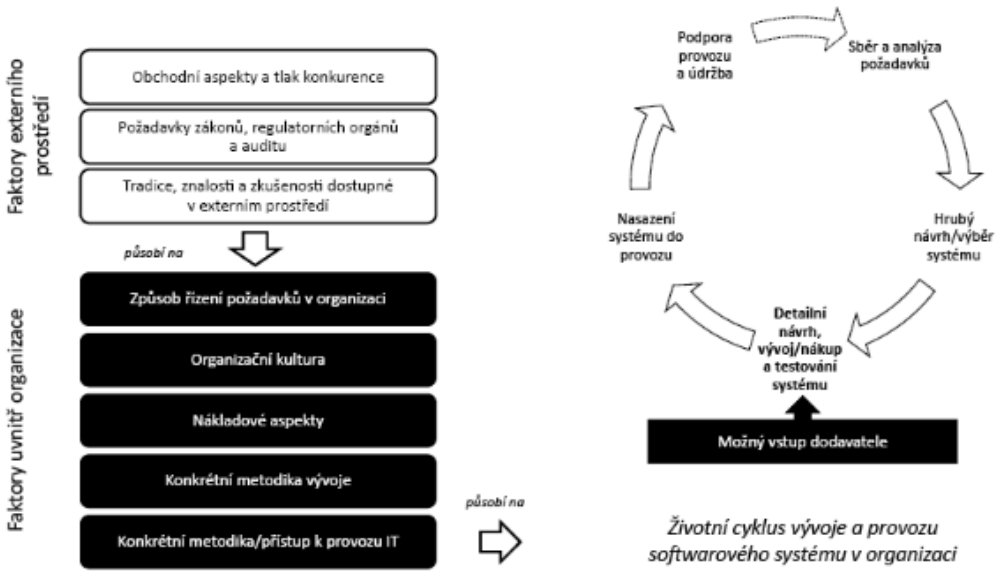
3.1 Vývoj softwaru

Proces, na jehož základě je vyvíjen software, a to již od prvotního záměru přes samotné uvedení na trh až po jeho údržbu, se nazývá model životního cyklu vývoje softwaru (Patton, 2002).

Do tohoto procesu může být začleněno desítky, stovky, ale i tisíce členů vývojářského týmu, kde každý má svou roli a všichni kooperují dle dopředu stanoveného plánu. Určitá náplň práce těchto lidí, způsob toho, jak mezi sebou komunikují a jejich rozhodování jsou nedílnou součástí procesu vývoje softwaru (Patton, 2002).

Metodika, která je použita během tohoto životního cyklu reprezentuje jeden z nejdůležitějších faktorů, jenž předurčuje filozofii testování v dané společnosti. Proto, je důležité přemýšlet, jak je ve společnosti software vyvíjen, zda jsou jednotlivé fáze vývoje odděleny a kdo za ně zodpovídá. Další důležitou otázkou je, zda jde jen o teorii nebo to tímto způsobem ve firmě skutečně funguje. Pokud není totiž sladěn formalizovaný životní cyklus vývoje softwaru a realita ve firmě, dochází k problémům (Bureš, a další, 2016).

Následující obrázek ukazuje řadu činností, které jsou vykonávány ve spojitosti s vývojem a provozem určité softwarové aplikace. Na obrázku je zdůrazněn dlouhodobý pohled v kontextu konkrétní organizace včetně následující údržby systému. V levé části je znázorněn obecný sled činností, které působí na vpravo znázorněné etapy vývoje softwarového systému (Bureš, a další, 2016).



Obrázek 1 - Model průběžného rozvoje systému v kontextu konkrétní organizace (Bureš, a další, 2016)

3.1.1 Životní cyklus vývoje softwaru

Vývoj softwaru probíhá řízeným způsobem, který se skládá z jednotlivých fází, kdy je po každé fázi posouzeno dosažení požadovaných cílů, které jsou podmínkou pro postoupení do fáze následující (Štědroň, 2006).

3.1.1.1 Analýza požadavků

Začátek životního cyklu je stanoven počáteční analýzou požadavků na systém. Cílem této analýzy je sběr požadavků, které definují funkčnost systému a jeho interakci s okolním světem (Brookshear, 2015).

V této fázi je důležitý zásadní vstup od zainteresovaných stran, kterými jsou budoucí uživatelé a další osoby ve vztahu s vyvíjeným systémem, ať už například právním nebo finančním. Je tedy nutné pokusit se pochopit to, co přesně uživatelé softwaru očekávají a provést nad tím analýzu, prodiskutovat se zainteresovanými osobami kompromisy nad potřebami, požadavky, náklady a proveditelností. Klíčová je v této fázi komunikace, jejíž nedostatek představuje hlavní důvody překročení stanovených nákladů a dodání softwaru po dohodnutém termínu (Brookshear, 2015).

Výsledkem této analýzy bude dokument zvaný specifikace požadavků na systém, jenž slouží jako písemná dohoda mezi jednotlivými stranami (Brookshear, 2015).

3.1.1.2 Návrh

Pokud je hotová fáze sběru a analýzy požadavků, vývoj se posouvá od otázky, co vytvořit, k otázce, jak to vytvořit. Dalším krokem je tedy sestavení plánu, jak na základě hotové analýzy navržený systém vytvořit (Brookshear, 2015).

Při návrhu softwaru je velice důležitá tvorba diagramů a modelů. Pomocí těchto nástrojů je potřeba definovat prvky systému, komponenty, úroveň zabezpečení, moduly, architekturu a typ dat, které prochází systémem (Banks, 2015).

3.1.1.3 Implementace

V této fázi se převádí návrh do zdrojového kódu v daném programovacím jazyce, vytváří se datové soubory a vyvíjí se databáze. Tato etapa ovlivňuje jak fázi testování, tak fázi údržby, proto je cílem co nejlépe napsaný kód (Brookshear, 2015).

3.1.1.4 Testování

Testování je hlavní měřítko kontroly kvality, které se používá při vývoji softwaru. Základní funkcí testování je detekovat chyby v softwaru. Cílem je tedy odhalení chyb požadavků, návrhu a kódování programu. Je to extrémně kritická a časově náročná aktivita, která vyžaduje správné plánování celého svého procesu (Agarwal, a další, 2010).

3.1.1.5 Provoz a podpora

Údržba systému je poslední, ale neméně důležitá etapa životního cyklu vývoje softwaru. Pokud je v hotovém softwaru jakákoliv chyba k opravě nebo je potřeba nějaké změny, je to aplikováno ve fázi údržby. Tato fáze může mnohokrát zabrat více času, než je čas strávený při vývoji a podepisuje se to také na nákladech. Náklady se pohybují okolo 50–80 % z celého vývoje pouze na samotnou údržbu (Agarwal, a další, 2010).

3.1.2 Metodiky vývoje

Metodika je určitý návrh, jakým způsobem realizovat konkrétní činnosti v průběhu životního cyklu softwaru. Metodikou vývoje rozumíme souhrn postupů, které vedou k dodání funkčního softwaru a odpovídají na otázky proč, kdo, kdy a co (Brookshear, 2015).

3.1.2.1 Vodopádový model

Přístupy v počátečním vývoji softwaru kladly požadavek na to, aby životní cyklus vývoje softwaru probíhal přísně sekvenčním postupem. Bylo tomu tak, protože se předpokládalo, že pokud se vyvíjí velký softwarový systém, bude potřeba tolik prostředků, že se nebude možné odchýlit od jednotného postupu. Tento postup procesu vývoje se nazývá vodopádový model, neboť se celý jeho směr ubírá pouze dolů (Brookshear, 2015).

Tento model je typický tím, že bez dokončení předchozí fáze nelze vstoupit do fáze následující, to ho činí časově náročným a nevhodným pro krátkodobé projekty (STH, 2022). Vzhledem k testování je to také jakási nevýhoda, protože se testuje až po ukončení fáze vývoje a na podstatnou chybu se může přijít těsně před vydáním neboli releasem.

Pro dlouhodobé projekty s jasně srozumitelnou definicí a disciplinovaným týmem je však model vodopádu velmi dobrým řešením a je zde velká výhoda v kvalitní specifikaci. Pokud se přechází do etapy testování, o každém detailu se již rozhodlo, vše bylo zapsáno a začleněno do softwaru, lze snadno vytvořit jasný testovací plán (Patton, 2002).

3.1.2.2 Inkrementální model

Na základě inkrementálního modelu vzniká software přírůstkovým způsobem. To znamená, že je nejdříve získána zjednodušená podoba produktu, který má omezené funkce. Tato podoba je vyzkoušena a následně zhodnocena budoucími uživateli. Poté jsou přidávány a testovány další funkce, dokud není software plnohodnotný (Brookshear, 2015).

3.1.2.3 Iterativní model

Iterativní model se může zdát podobný předchozímu inkrementálnímu modelu, ale tyto dva modely jsou však rozdílné. Princip iterativního modelu je ve zdokonalování jednotlivých postupných podob softwaru (Brookshear, 2015). Iterativní model akceptuje změny pomocí zpětnovazební smyčky, zatímco neiterativní model se snaží změnám vyhýbat, čímž zvyšuje riziko, že dodaný software nebude souhlasit s potřebami uživatelů a zainteresovaných osob (IBM, 2021).

3.1.2.4 Prototypový model

V metodice protypování je prototyp vyvinut před skutečným softwarem, jeho vlastnosti jsou omezené a v porovnání se skutečným softwarem má neefektivní výkon. Prototyp je vyvinut proto, aby získal zpětnou vazbu od zákazníka, která je velmi cenným podkladem. Zpětná vazba je pak implementována do té doby, dokud není software dle zákazníkovo očekávání (STH, 2022).

3.1.2.5 V-model

V-model je středobod firemních strategií, jelikož klade důraz na oblast testování. Tento model byl definovaný již ve druhé polovině osmdesátých let 20. století. Princip této metodiky je však založen na vodopádovém modelu s tím rozdílem, že plánováni společně s testováním začíná již v rané fázi vývoje.

Model je tvořen několika dvojicemi, kde na levé straně se nacházejí aktivity, které souvisejí se specifikací a návrhem a pravou stranu reprezentuje samotné testování. Tento způsob zdůrazňuje důležitost testování ve spojitosti s odpovídajícími analytickými aktivitami.

Model v podstatě tvrdí, že není pouze důležité systém správně navrhnout a naprogramovat, ale v kritických časových okamžicích je nutné zařadit testování jako nástroj ověřování návrhu a realizace (Bureš, a další, 2016).

3.1.2.6 Spirálový model

Spirálový model byl zaveden již v roce 1986 a dosud se ukazuje jako velmi efektivní prostředek, jelikož řeší problémy ostatních modelů. Princip modelu je založen na počáteční malé definici nejdůležitějších funkcí, které jsou vyzkoušeny. Poté je získána zpětná vazba od uživatelů a následně se přechází do dalšího stádia vývoje. Pro testera je tento model cenný, protože má možnost projekt ovlivnit již od samotného začátku za pomoci průběžného testování (Patton, 2002).

3.1.2.7 Agilní model

Agilní přístup, který je kombinací iterativního a inkrementální přístupu, je jedním z nejmladších, avšak ve firmách zaváděný teprve nějaký čas. Za vznikem tohoto přístupu stojí takzvaný Agilní manifest, který se stal v roce 2001 prostředkem pro projev nespokojenosti s tradičními metodami vývoje softwaru trpící přemírou dokumentace a byrokratickými procesními koncepty (Bureš, a další, 2016).

Jedna z mnoha změn, která se dotýká samotného testování je v oblasti vztahů s vývojáři. Kvalita softwaru je nyní nejdůležitějším aspektem pro celý tým, a tedy i jeho společná odpovědnost (Bureš, a další, 2016).

V agilním prostředí není software vyvíjen jako celek, ale je rozdělen na části, což umožňuje rychlý vývoj softwaru, jenž dokáže reagovat na případnou změnu požadavků v průběhu každého vývojového cyklu neboli sprintu. Na konci každého sprintu si vlastník softwaru ověří software a po jeho schválení je doručen uživateli (zákazníkovi). Zpětná vazba od uživatele je přejata pro zlepšení a návrhy, které jsou zpracované v dalším sprintu. Testování se provádí v každém sprintu, aby bylo minimalizováno riziko jakýchkoli selhání (STH, 2022).

Tento model však nepřináší pouze výhody. Pro agilní metodu je důležitá potřeba zkušených a kvalifikovaných zdrojů a pokud zákazník neví, co přesně očekává od softwaru, tak projekt může snadno selhat (STH, 2022).

3.2 Testování

Bohužel výše popsané metody jsou idealistickým pohledem na možnost způsobu vedení vývoje softwaru. V reálném životě neexistuje projekt, který by postupoval krok po kroku dle zvolené metody (Patton, 2002).

V projektech vývoje softwaru představuje průměrně 20-40 % pracnosti pouze samotné testování. Při vývoji řídicího systému, na který jsou kladeny vysoké požadavky na spolehlivost může pracnost testování převýšit pracnost samotného vývoje. Jestliže má být testování efektivní, musí k němu být přistupováno jako k nezávislé odborné disciplíně, jako tomu je u analýzy, návrhu, architektury, vývoje nebo u projektového řízení (Bureš, a další, 2016).

3.2.1 Definice

Testování softwaru je definováno jako činnost, která má za úkol ověřit, zda skutečné výsledky odpovídají těm očekávaným a zajistit, že bude software požadované kvality a bez závad. Testování pomáhá identifikovat chyby, mezery nebo chybějící požadavky v rozporu se skutečnými požadavky (Nordeen, 2020).

3.2.2 Proč je testování důležité

Testování odhalí v softwaru chyby, které mohou být drahé, ale zároveň nebezpečné, neboť mohou způsobit vysoké finanční ztráty ba dokonce ztráty na lidských životech. V historii je takových příkladů nemalé množství (Nordeen, 2020).

Pár následujících příkladů ukazuje, jak je testování nepostradatelné a jeho nedostatečné provedení může mít fatální důsledky.

- Roku 1962 musela být zničena raketa Mariner 1 kvůli opomenuté závorce. Po vypuštění mířila jinam, než měla a musela být pár minut po startu obsluhou zničena. Cena za tuto chybu byla 18,5 milionu dolarů (Hájek, 2010).
- Lékařský přistroj Therac-25, jehož úkolem v roce 1985 byla radiační terapie, zabil a zranil několik lidí, protože mohl být nastaven tak, že byly paprsky použity ve vysokoenergetickém režimu bez toho, aniž by se pacient chránil (Hájek, 2010).
- V roce 1994 byla vydána společností Disney první multimediální hra na CD-ROM, Lví král. Nespočet dětí však zažilo velmi smutné Vánoce, protože Disney hru

důkladně neotestovalo a nebylo tak možné ji spustit na několika různých modelech osobních počítačů (Patton, 2002).

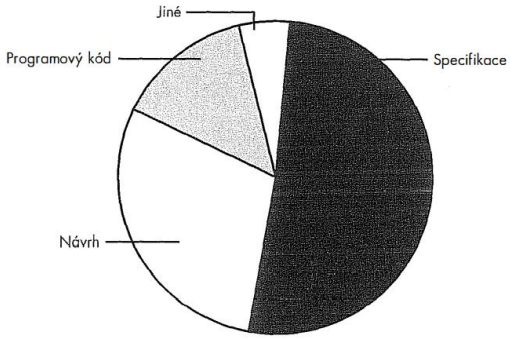
- V roce 1994 Intel vypustil do světa procesor Intel Pentium, který měl chybu v dělení s pohyblivou čárkou. O chybě prý díky testům Intel věděl, ale dle inženýrů nebyl důvod chybu opravovat nebo publikovat, dokud na chybu nepřišel jistý uživatel, který své zjištění zveřejnil na internetu. Tato chyba stála Intel 475 milionu dolarů (Patton, 2002).
- Raketa Ariane 5 se v roce 1996 pár sekund po vypuštění vypnula a zřítila do moře. Systém pro řízení nalezl chybu, vypnul se a náhradní systém, kterému bylo řízení předáno trpěl chybou stejného charakteru (Hájek, 2010).
- V prosinci roku 1999 se během pokusu o přistání na Mars ztratil přistávací modul sondy NASA. Při vyšetřování, proč se konkrétní chyba stala, se zjistilo, že šlo zřejmě o neočekáváné nastavení jediného datového bitu, což pravděpodobně zapříčinilo, že se při přistání vypnuly brzdící motory a modul se volným pádem zřítil několik stovek metrů na zem (Patton, 2002).
- Nissan musel svolat z trhu více než milion vozů kvůli chybě softwaru v senzoru airbagů. Z důvodu tohoto selhání se staly minimálně dvě nehody (Nordeen, 2020).
- Starbucks byl nucen dočasně zavřít přes 60 % svých kaváren v USA a Kanadě, protože mu selhal software v systému. V jednu chvíli podávali kávu zdarma, protože systém nebyl schopen zpracovat transakci (Nordeen, 2020).

Dle výše zmíněného výčtu historických softwarových chyb je patrné, že při testování jde o ověření kvality softwaru a o nalezení všech chyb, závad, defektů neboli bugů. O chybě v softwaru se dá uvažovat, pokud je splněna jedna nebo několik podmínek.

a) Software by měl dle specifikace něco dělat a nedělá to.
b) Software dle specifikace dělá něco, co by dělat neměl.
c) Software dělá něco, co není zmíněno ve specifikaci.
d) Software nedělá něco, co není zmíněno ve specifikaci, ale mělo by být.
e) Software nepracuje, jak by měl, je těžko srozumitelný, pomalý a koncový uživatel by ho nepovažoval za korektní (Patton, 2002).

Každý člověk může kdykoliv udělat chybu, bez ohledu na to, na čem pracuje. Business analytik může pochybit již při tvorbě specifikace požadavků, tester může chybovat v testovacím případě, programátor v kódu a technický spisovatel může chybně napsat příručku. Jakýkoli produkt může mít a bude mít chyby, protože jakýkoli pracovník může a udělá chyby (Black, 2014).

Nejčastější důvody vzniku chyb ukazuje následující koláčový graf, ze kterého je patrné, že chybou číslo jedna je nekvalitní specifikace. Proč tomu tak je, má několik důvodů. Buď je specifikace úplně opomíjena, je příliš stručná a nezachází dopodrobna nebo je neustále měněna anebo nebyla dostatečně prodiskutována celým vývojovým týmem. Druhou nejčastější příčinou chyb je fáze návrhu. Návrh je v mnoha případech uspěchaný, často se mění nebo není důkladně probrán týmem, stejně jako specifikace (Patton, 2002).



Obrázek 2 - příčiny chyb softwaru (Patton, 2002)

3.2.3 Náklady na kvalitu

Zkoumání kvality softwarových produktů a procesů tvorby softwaru, kterou se zabývá softwarové inženýrství, je široká a důležitá oblast řešená několika normalizačními orgány jako jsou ISO, IEEE, ANSI, ISTQB apod. (Agarwal, a další, 2010).

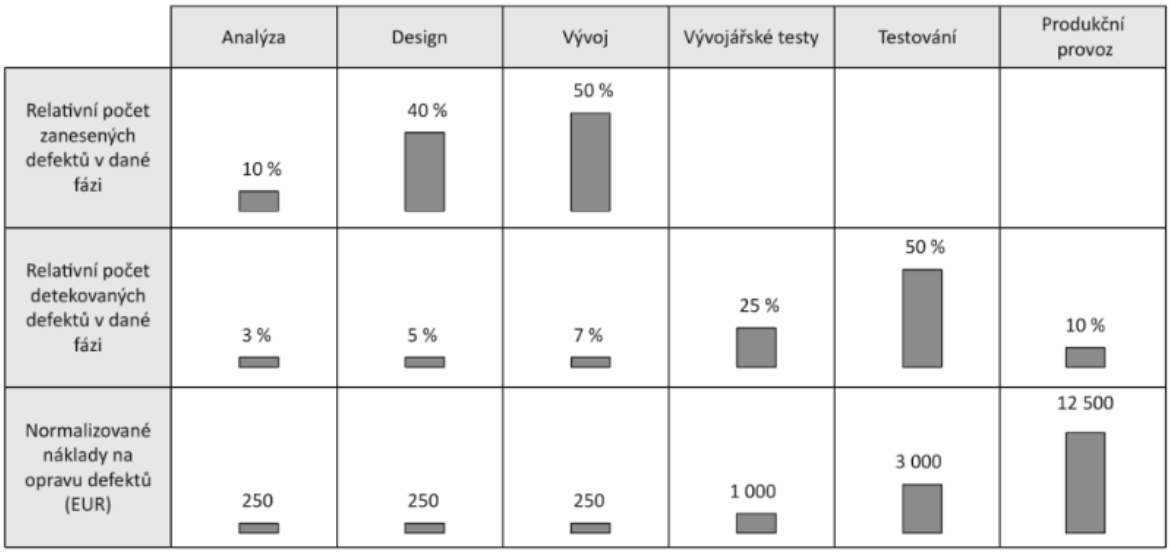
Jedna z definic zní, že se jedná o shodu s explicitně stanovenými funkčními a výkonnostními požadavky, explicitně dokumentovanými vývojovými standardy a implicitními vlastnostmi, které se očekávají od profesionálně vyvinutého softwaru (Agarwal, a další, 2010).

Otázka, která doprovází každý IT projekt a je žhavým tématem diskuse mezi zainteresovanými stranami, sponzorem, business vlastníkem řešení, dodavatelem a manažerem testování, se samozřejmě týká vynaložených prostředků. Kolik jich je žádoucích, potřebných a možných a proč je testování tak drahé (Bureš, a další, 2016).

Jedním z modelových příkladů, kromě těch zmíněných výše, proč platit „tolik“ za testování, je následující. V roce 2015 byl vydán Úřadem pro letectví bezpečnostní bulletin,

který se týkal všech modifikací Boeingu 787 Dreamliner. Jednalo se o softwarovou chybu v řídící jednotce generátorů palubního napětí, ve kterém mělo dojít po 248 dnech neustálého provozu řídící jednotky k přetečení rozsahu datového typu a následný kolaps softwaru do nouzového režimu. Nejnebezpečnější fáze byla vzlet a přistání. Letadel Boeing 787 bylo v té době v provozu 269, a toto byl bezesporu kritický defekt. Z tohoto důvodu bylo úřadem vydáno nařízení k urychlenému restartování všech jednotek, aby byl čas na odstranění této závažné chyby prodloužen o minimálně 248 dnů. Jaká byla cena pouze za tento čas, který měl sloužit na hledání závady? Téměř 23 tisíc dolarů za restartování jednotek a stažení letadel z provozu a ztráta dobré reputace za cenu zhruba o řád vyšší. Vyřešení samotného defektu, regresní testy, testy zátěžové a simulované testy pro dlouhodobý provoz se musely pohybovat v milionových hodnotách. A to je odpověď na otázku, proč je testování tak drahé. Pokud se investuje několik tisíc, v tomto případě dolarů, do vhodných typů testů, je možné předejít až milionovým ztrátám, které zapříčiní takováto nešťastná chyba v softwaru (Bureš, a další, 2016).

Následující obrázek ukazuje průzkum společnosti Capgemini, který se týká cen oprav defektů v různých fázích vývoje a do kterého byly zapojeny stovky integračních projektů po celé Evropě. Výsledkem je, že se přímé náklady na chybu, která je nalezena až při nasazení, pohybují ve výši 12,5krát vyšší, než by byly náklady na odstranění ve fázi vývojářských testů. A pokud budou do testování zapojené i například revize analytických dokumentů, může být chyba 50krát levnější, a to se dozajista vyplatí (Bureš, a další, 2016).



Obrázek 3 - Přímé náklady na opravu defektu v závislosti na fázi, kdy byl defekt nalezen (Bureš, a další, 2016)

Vzhledem k tomu, že chyby mohou být zavedeny do kterékoli fáze životního cyklu softwaru, a protože se náklady na odstranění zvyšují, čím déle existují, je důležité se snažit je odstranit v té fázi, ve které byly vytvořeny. Nebo ještě lépe zlepšovat procesy tak, aby předcházely chybám (Black, 2014).

Výše nákladů dle Pattona roste postupem času vždy na desetinásobek a má logaritmickou stupnici. Chyba na začátku ve fázi analýzy nemusí stát nic, ale stejná chyba nalezená ve fázi kódování nebo dokonce testování, se může pohybovat ve vysokých částkách (Patton, 2002).

Co všechno tedy spadá do ceny za kvalitu, je popsáno následujícími kategoriemi dle standardu ISTQB (International Software Testing Qualification Board), jenž je jedním z největších certifikačních systému na světě a jehož terminologie je používána jako globální jazyk v oblasti testování (ISTQB, 2022).

- Náklady využité na prevenci defektů, do kterých spadá školení vývojářů o psaní bezpečného a snadno udržovatelného kódu nebo na zavedení kódovacích standardů.
- Náklady na detekci defektů, které jsou tvořené všemi aktivitami spojenými s testováním, od vytváření testovacích případů, po revize specifikací, nastavení testovacích prostředí, vykonávání samotných testů a reportování o výsledcích.
- Náklady vyvolané při odstraňování defektů, kam spadají náklady na opravu ve specifikaci, v kódu a s tím spojené další aktivity jako jsou nasazování na různá prostředí, přetestování apod.
- Náklady způsobené selháními na produkčním prostředí. V těchto nákladech jsou zahrnuty všechny aktivity, které souvisejí s opravou možných produkčních chyb a distribucí již opraveného systému uživatelům (Bureš, a další, 2016).

Náklady na to, aby byla zajištěna kvalita před tím, než se software nasadí na produkční prostředí se spočítají součtem prvních tří výše zmíněných nákladů.

Cena zajištění kvality = Náklady na prevenci + náklady na detekci + náklady na opravu

A přínos zmíněných investic pro to, aby měl software požadovanou kvalitu ještě před tím, než se nasadí do produkčního prostředí oproti případným nákladům, které by bylo nutné vynaložit, kdyby se neinvestovalo, ukazuje cenu za onu kvalitu (Bureš, a další, 2016).

Cena za kvalitu = Náklady způsobené selháním na produkčním prostředí – Cena zajištění kvality

3.2.4 Terminologie

Pro lepší pochopení teorie a vyhnutí se nejasnostem jsou v této kapitole vysvětleny nejpoužívanější pojmy v oblasti testování.

V první části je potřeba vysvětlit základní pojmy, které jsou často zaměňované a nesprávně používané. Ve druhé části budou vysvětlené důležité termíny, které jsou při testování používané na denní bázi a k nim i jejich anglický globálně používaný výraz (Patton, 2002).

3.2.4.1 Přesnost a správnost

První dvojicí zaměňovaných pojmů jsou přesnost a správnost. Pokud je software správný, nemusí být i přesný a naopak. Kupříkladu softwarová kalkulačka musí mít obě vlastnosti, ale pokud se zaokrouhluje na páté desetinné místo, za ním již může přesnost kolísat (Patton, 2002).

3.2.4.2 Verifikace a validace

Pojmy, které lze přeložit jako ověřování a kontrola se běžně zaměňují, ale jejich význam je různý. Pro verifikaci platí, že se ověřuje, zda software odpovídá zadané specifikaci. Pro validaci však platí, že probíhá kontrola, zda software vyhovuje uživatelským požadavkům (Patton, 2002).

3.2.4.3 Kvalita a spolehlivost

Kvalita už byla vysvětlena výše, stručným souhrnem se o kvalitní software jedná, pokud vyhovuje potřebám zákazníka, spolehlivost však vyjadřuje, jak často software chybuje nebo ničí data. Proto je pro testování důležité po celý vývoj softwaru verifikovat a validovat (Patton, 2002).

3.2.4.4 Testování a zajišťování kvality

Poslední porovnávaná dvojice je testování a velmi známý pojem zajišťovaní kvality, anglicky Quality Assurance. Cíl testera je jasný, vyhledávat chyby, ale člověk, který zajišťuje kvalitu má za úkol vytvářet a prosazovat metodologii, která vylepší vývojový proces a tím předejde vzniku chyb (Patton, 2002).

3.2.4.5 Plán testování (test plan)

Testovací plán je stěžejní dokument, který popisuje rozsah, přístup, zdroje, druhy a kategorie testů a harmonogram aktivit pro celý proces testování. Tento dokument by se měl vytvářet před počátečním testováním, protože poté slouží pro všechny členy testovacího týmu a odpovídá na všechny otázky týkající se onoho procesu (Hlava, 2011).

Následujících 7 kroků slouží pro vytvoření testovacího plánu dle IEEE 829.

1) **Analýza produktu** – před testováním je důležité důkladně prostudovat produkt. Kdo ho bude používat? K čemu je určen? Jak bude pracovat?
2) **Návrh strategie testování** – jedná se o kritický krok při vytváření test plánu. Jedná se o dokument vysoké úrovně, jenž vytváří test manažer. Jsou zde uvedeny cíle a prostředky k jejich dosažení.
3) **Definování cílů testování** – cílem je nalézt co nejvíce defektů, proto je důležité sepsat všechny funkce softwaru, které je nutné otestovat a na základě toho definovat cíle testu.
4) **Definování testovacích kritérií** – kritéria testu jsou pravidlem, na kterém je založen testovací postup. Jsou zde dva typy kritérií, a to vstupní a výstupní. Vstupní kritéria popisují, v jakou chvíli je možné začít testovat a výstupní kritéria popisují, v jakou chvíli je testy možné pokládat za uzavřené.
5) **Plánování zdrojů** – jedná se o kompletní souhrn všech typů zdrojů potřebných pro vyhotovení projektu.
6) **Plán testovacího prostředí** – plán se zde týká nastavení softwaru a hardwaru, na kterém bude testovací tým provádět testovací případy.
7) **Harmonogram –** vytvoření pevně daného plánu, který slouží jako nástroj pro sledování průběhu projektu má za úkol test manažer.
8) **Určení výsledků testování** – v každé fázi testování existují různé výstupy z testování, bývá tak před, během a po testování (Hamilton, 2022).

3.2.4.6 Testovací případ (test case)

Testovací případ je popisem konkrétní akce, která je prováděna v souvislosti s určitým softwarem a očekávaného výsledku této akce. Tento dokument je tedy popisem konkrétní činnosti, kterou je třeba otestovat. Typické vlastnosti testovacího případu jsou následující.

- **Identifikátor**, který poslouží k identifikaci v dalších dokumentech.

- **Účel** daného případu.
- **Podmínky**, ve kterých se typicky uvádí potřebná testovací data či prostředí potřebná pro otestování.
- **Specifikace kroků a vstupů**, jež slouží ke zdárnému a opakovatelnému uskutečnění případu.
- **Očekávané výsledky**, které obsahují potřebné informace pro rozhodnutí, zda byl test úspěšný či nikoliv (Hlava, 2011).

3.2.4.7 Testovací scénář (test scenario)

Testovací případy jsou hlavním podkladem pro tvorbu testovacích scénářů, které vystihují, jak by měla vypadat očekávaná funkčnost softwaru. Testovací scénář je popsán sadou jednotlivých kroků, jak prostupovat aplikací, co ověřovat a zda je vykazované chování dle očekávání (Knotek, 2014).

3.2.5 Role a odpovědnosti

Testovací tým má klíčovou roli při definování úspěchu konkrétního softwaru, a to i po dokončení projektu. Proto je důležité zajistit, aby byl testovací tým složen z talentovaných a schopných profesionálů, kteří mají stejný cíl, a to vytvoření kvalitního bezchybného systému. Následující role nesmí chybět v žádném projektu.

3.2.5.1 Tester

Obsah práce testera se může různit v závislosti na velikosti firmy ve které pracuje. V zásadě se však tester obeznámí se softwarem a následně provádí především manuální testování dle předem připravených testovacích scénářů. Úkolem testera je během provádění testů zaznamenávat defekty do testovacího nástroje včetně jejich závažnosti. Základními vlastnostmi testera by měly být zvídavost, komunikativnost, pečlivost a důslednost (Hlava, 2012).

3.2.5.2 Test analytik

Náplní práce test analytika je tvorba test analýzy. Jeho úkolem je na základě specifikace analytika pochopit, jak má vyvíjený software správně fungovat a efektivně připravit testovací případy, při kterých zjišťuje nedostatky v analýze. Při psaní testovacích scénářů pak analyzuje, jaká testovací data budou nutná připravit a při testování vypomáhá

testovat. Vlastnosti test analytika by měly být stejné jako pro testera, s tím, že navíc disponuje analytickým myšlením (Hlava, 2012)

3.2.5.3 Test manažer

Test manažer vede celý testovací tým, ve kterém hraje ústřední roli a přebírá veškerou odpovědnost za úspěch projektu. Zodpovědností test manažera je budování a vedení testovacího týmu, definování rozsahu testování, nasazení a správa prostředků pro testování, plánování, nasazení a správa testovacího úsilí pro danou zakázku a komunikace napříč vývojářským týmem (Hamilton, 2022). Vlastnosti, kterými disponuje manažer jsou schopnost plánování, delegování, musí umět kontrolovat, ale i motivovat svůj tým (Hlava, 2012).

3.2.6 Metody testování

3.2.6.1 Manuální a automatizované testování

Manuální testování je testování, u kterého je přítomen člověk, jenž musí používat svůj mozek a své ruce k vytváření scénářů, které jsou ukazatelem toho, že software selže či splní své poslání. Manuální testování je nejlepší možnost pro hledání chyb, které mají, co dočinění se základní business logikou daného softwaru. Business logikou jsou myšleny požadavky na funkcionalitu, která je složitá a vyžaduje, aby člověk ověřoval, zda je správná (Whittaker, 2009). Manuální testování má mnoho výhod a není v budoucnu očekáváno, že by mohlo být plně nahrazeno testováním automatizovaným. Aplikace, aby mohla být zautomatizovaná, musí být napřed dostatečně manuálně otestována a k tomu není zapotřebí žádný nástroj (Nordeen, 2020).

Automatizované testování má nejdůležitější motivaci, a to snahu ušetřit náklady na neustálé opakování testů a získat způsob, který je efektivnější nejen z hlediska časového. Úkolem automatizovaných testů není tedy nahradit testy manuální v plné šíři, ale nahradit jimi opakované, zejména regresní testy a pomoci testerům efektivněji využít kapacity pro testování nových funkcionalit. Jejich předními vlastnostmi jsou rychlost, kdy jsou neocenitelným pomocníkem při testování stovek případů. Další vlastností je efektivita. Pokud jsou spuštěny automatizované testy, zbývá čas na to plánovat a psát nové testovací případy. Další předností je správnost a přesnost. Testovací nástroj je na rozdíl od člověka, který testuje už po několikáté to samé, neomylný a jeho výsledky jsou vždy stoprocentní.

Neposlední vlastností, ale poslední zde zmíněnou je neúnavnost. Člověk se časem a opakováním činnosti unaví a již nemusí vidět chyby, které se v softwaru vyskytují, automat svou práci nikdy nevzdá (Patton, 2002). Aby to však nevypadalo, že automatizace přináší pouze benefity, je nutné zmínit i nějaké její slabší stránky. Největší slabinou je údržba automatizovaných testů, hlavně těch, které jsou založené na uživatelském rozhraní. Další ze slabin je, že automat narozdíl od testera není schopen reagovat na nečekanou situaci, stačí malá změna v uživatelském rozhraní nebo změna API a test přestane fungovat. Dalším příkladem situace, kdy je potřeba, aby zasáhnul tester je ta, když test skončí s negativními výsledky. Automatizovaný test zvládne analyzovat chybné chování, ale ne tak dopodrobna, jako tester. Rozdílů je možné najít ještě mnoho, proto je důležité na ně myslet při přípravě strategie automatizovaného testování a při výběru testovacích scénářů, které mají být automatizované (Bureš, a další, 2016).

3.2.6.2 Konfirmační a regresní testování

Tyto dva typy testování se používají po provedení určité změny v systému, ať už příčinou nějaké chyby nebo nasazením nové funkcionality. Oba typy se provádí na všech úrovních testování a jsou velice důležitým typem testů, protože změny v softwaru patří k jeho průběžnému vývoji a rozšiřování.

Konfirmační testování neboli retestování je provedení všech testů, které důsledkem defektu selhaly a je potřeba je znovu otestovat na nové verzi aplikace. Cílem tohoto testování je potvrzení, že změna v systému skutečně defekt odstranila (Učební osnovy – Certifikovaný tester, 2018).

Regresní testy na druhou stranu zajišťují neočekávané chování softwaru na základě provedené změny v kódu nebo konfiguraci prostředí. Regrese tedy znamená stav, při kterém po úpravě přestane fungovat něco, co před úpravou fungovalo dle požadavků. Příčinou nalezení defektů regresními testy může být nesprávná architektura kódu, nedomyšlené změny zdrojového kódu vývojářem, chybná správa verzí kódu a dokumentace projektu, změny na úrovni změnových komponent, které využívá systém nebo například chybná instalace do testovacího prostředí. Sady regresních testů se testují mnohokrát, proto by mělo být zamýšleno o automatizaci těchto testů. Speciální typ regresních testů je smoke test, který ověřuje, zda je systém vůbec připraven pro podrobnější testování (Bureš, a další, 2016).

3.2.6.3 Statické a dynamické

Statické testování nepotřebuje spuštění testovaného softwaru, ale spoléhá pouze na jeho manuální prověření, takzvanou revizi nebo na vyhodnocení kódu či dalších pracovních produktů (specifikace, scénáře, uživatelské příručky, smlouvy apod.) za pomoci nástroje, takzvanou statickou analýzu. Bez statické analýzy by nemohly fungovat kritické počítačové systémy používané pro letecký, jaderný nebo zdravotnický průmysl. Velkým přínosem tohoto typu testování je odhalení defektů v raném stádiu vývoje softwaru, což jak už bylo zmíněno, vyjde mnohem levněji než nalezení defektů v pokročilejší fázi životního cyklu vývoje softwaru. Typickými defekty, které statické testování odhalí snáze než testování dynamické, jsou v požadavcích, v samotném návrhu, při psaní kódu, mohou to být odchylky od norem nebo například zranitelnosti v bezpečnosti (Učební osnovy – Certifikovaný tester, 2018).

Dynamické testování se provádí se spuštěním kódu a ověřuje funkční chování softwaru a celkový výkon systému. Hlavní cíl tohoto testování je tedy potvrzení, že očekávání souhlasí s reálným výsledkem. Dynamické testování se na rozdíl od statistického vykonává na všech úrovních testování a náklady na něj jsou mnohem vyšší. Zásadním rozdílem je tedy ve stručnosti to, že statistické testování je určeno pro prevenci vzniku chyb, zatímco účelem dynamického testování je hledání a následná oprava chyb již vzniklých (Hamilton, 2022).

3.2.6.4 Funkcionální a nefunkcionální testování

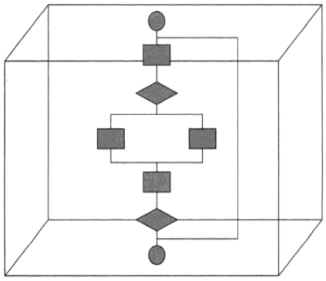
Funkcionální testování spočívá v testování funkcí, které má za úkol software vykonávat a hodnotí tak jeho chování. Tyto funkcionální požadavky jsou uvedené a popsané v dokumentech, kterými jsou specifikace, uživatelské scénáře nebo testovací případy. Pro návrh a vykonávání testů by měly být vyžadovány znalosti v konkrétní oblasti podnikání, kterou software řeší (Učební osnovy – Certifikovaný tester, 2018).

Oproti tomu nefunkcionální testování zahrnuje testy charakteristiky softwaru, kterými jsou použitelnost, bezpečnost a výkonnost. Odpovídá na otázku „jak“ se daný software chová, zatímco funkcionální testování odpovídá na otázku „co“ by měl software dělat. Pro návrh a vykonávání těchto testů je za potřebí znalostí slabých stránek daného designu či technologie a konkrétní základny uživatelů, kteří budou se systémem pracovat (Učební osnovy – Certifikovaný tester, 2018).

3.2.6.5 Testování černé, bílé a šedé skříňky

Technika dynamického testování černé skříňky se provádí na spuštěném softwaru bez předchozí znalosti vnitřní a programové struktury. Během testování jsou testeři obeznámeni pouze s funkcionalitami a specifikací softwaru, proto je tato metoda funkčním testováním. Na úrovni systému je celý software pokládán za černou skříňku a testovací případy jsou generovány a vybírány na základě funkčních specifikací nebo popisu modelu případu použití softwaru. Na úrovni jednotky je však funkce nebo metoda považována za černou skříňku a testovací případy jsou generovány a vybírány počínaje funkční specifikací a charakteristikou jejích rozhraní. Techniky testování černé skříňky na úrovni systému zahrnují testování scénářů na základě případu použití. Techniky testování černé skříňky na úrovni systému i na úrovni jednotek zahrnují analýzu hraničních hodnot, testování tříd ekvivalence, testování na základě rozhodovací tabulky nebo stromu, grafy příčin a následků a testování založené na stavu (Saleh, 2009). Výhodou testování černé skříňky je jeho nenáročnost, rychlost, transparentnost pro zákazníka a není zde nutnost poskytovat zdrojový kód. Nevýhodou však může být nižší kvalita kódu. Systém sice dělá to, co má, ale není známo jak a nežádoucí chování uvnitř, které není možné odhalit (Čermák, 2008).

Testování bílé skříňky se odlišuje tím, že je testování prováděno na spuštěném softwaru a tester je obeznámen se strukturou kódu a datovými a řídícími toky. Jedná se tedy o strukturální testování, kde na úrovni systému je poskytnutý kompletní zdrojový kód a dokumentace, na jejichž základě jsou pak tvořeny testovací případy. Na úrovni testování jednotek jsou testy prováděné obvykle vývojářem, který je autorem kódu nebo testerem, jenž byl v rámci jeho týmu (Saleh, 2009). Výhodou je zde včasné odhalení chyb, které umožňuje podrobná analýza kódu ještě před zkompilováním kódu anebo rovnou odhalení nežádoucího kódu. Nevýhodou je na druhé straně náročnost, protože toto testování vyžaduje hluboké znalosti programovacího jazyka a testovacích nástrojů a s ním i vysoké náklady na specializované nástroje (Čermák, 2008).



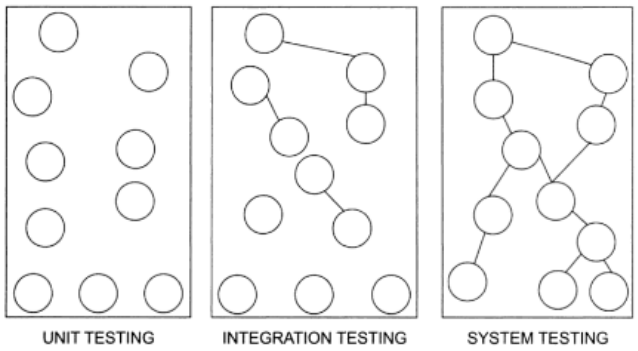
Obrázek 4 - Vizualizace testování bílé skříňky (Agarwal, a další, 2010)

Testování šedé skříňky je kombinací obojího. Na této úrovni je předpokládáno, že tester není dopředu do hloubky obeznámen s datovými toky a strukturou kódu. Koncept šedé skříňky je tedy velice jednoduchý, tester ví okrajově, jak software funguje uvnitř a je tedy schopen ho otestovat zvenku. Předností této metody je, že v sobě skýtá výhody černé i bílé skříňky a tester je díky pár znalostem schopen zpracovat velmi inteligentní testovací scénáře. Nevýhodou však může být neúplné otestování kvůli nekompletní znalosti všech toků a kódu a stejně tak, i kvalita kódu nemusí být příliš vysoká (Čermák, 2008).

3.2.7 Úrovně testování

Úrovně testování jsou seskupeny dle toho, kde jsou v dané úrovni vývoje softwaru provedeny nebo dle toho, jaké podrobnosti jsou jejich obsahem. Účelem úrovní testování je učinit samotné testování systematické a snadno definovat všechny testovací případy pro konkrétní úroveň (Hammad, 2020).

Pro každou úroveň existuje specifický testovací základ, objekt testování, typické defekty, specifické odpovědnosti a přístupy a prostředí, na kterém bude provedeno testování. Cíle mají však všechny úrovně testování téměř totožné, a to jsou snížení rizika výskytu defektů, ověření, zda se systém chová tak, jak bylo specifikováno, nalezení defektů v části a zamezení jeho proniknutí do vyšších vrstev (Učební osnovy – Certifikovaný tester, 2018).



Obrázek 5 - Úrovně testování (Agarwal, a další, 2010)

3.2.7.1 Jednotkové testování

Zaměřením jednotkové testování, známého také dle anglického unit testing, je ověřování nejmenších samostatně testovatelných částí systému, které je zpravidla prováděno samotným vývojářem kódu.

Objekty testování jsou zde jednotky nebo moduly, zdrojový kód a datová struktura, třídy a databázové systémy.

Typické příklady defektů a selhání, která se v této úrovni testování objevují jsou funkcionality, které nebyly zmíněny ve specifikaci, tudíž jsou nesprávné, problémy toků dat a špatný kód a logika. Tyto defekty jsou po nalezení ihned napraveny obvykle bez typických formálností (Učební osnovy – Certifikovaný tester, 2018).

3.2.7.2 Integrační testování

Integrační testování se zaměřuje na spolupráci neboli interakci mezi jednotlivými komponentami či systémy. Je zde vlastně kontrolováno, zda není datový tok nějakým způsobem narušen. Ověření komunikace mezi komponentami bývá obvykle úlohou vývojářů, zatímco komunikaci mezi systémy už testují samotní testeři.

Objekty testování v této úrovni jsou subsystémy, databáze, infrastruktura, rozhraní a mikroslužby.

Klasickými typy defektů jsou špatná nebo chybějící data nebo jejich nesprávné kódování, dále pak nesoulad rozhraní nebo například selhání komunikace mezi systémy a komponentami (Učební osnovy – Certifikovaný tester, 2018).

3.2.7.3 Systémové testování

Systémové testování již pohlíží na systém jako na celek a zkoumá tak jeho chování a schopnosti. V této úrovni se již provádějí end-to-end testy samotnými testery, to znamená, že se ověřuje kompletní průchod daným softwarem, aby se zajistilo, že je jeho chování dle očekávání.

Mezi charakteristicky testované objekty zde náleží aplikace, hardwarové nebo softwarové systémy, operační systémy nebo konfigurace systému a konfigurační data.

Typickými defekty jsou při systémovém ověřování nesprávné výpočty, funkcionality, které jsou neočekáváné nebo špatné, nesprávné řízení dat nebo datových toků nebo neočekávané výsledky end-to-end testů (Učební osnovy – Certifikovaný tester, 2018).

3.2.7.4 Akceptační testy

Akceptační testy neboli UAT (user acceptance test) se po úspěšném průchodu všemi úrovněmi testování již kompletně provádí na straně zákazníka. Toto testování je posledním krokem před nasazením do živého prostředí a zahájení používání koncovým uživatelem. Cílem této fáze již není nalézt defekty, i když je to stále pravděpodobné.

Objekty pro toto testování jsou testovací systém, konfigurace systému a konfigurační data, podnikové procesy pro úplnou integraci, provozní procesy, formuláře nebo reporty.

Chyby v této úrovni mohou mít podobu nesplněných podnikových nebo uživatelských požadavků, podniková pravidla nejsou vhodně implementována, systém nemusí splňovat smluví či regulatorní požadavky nebo jsou zde nalezena zranitelná místa z hlediska bezpečnosti (Učební osnovy – Certifikovaný tester, 2018).

3.2.8 Testovací prostředí a testovací data

Dle slovníku ISTQB je pojmem testovací prostředí myšleno prostředí, které obsahuje hardware, software, simulátory, nástroje a další vybavení nezbytné pro testování. Testovací prostředí oddělené od toho produkčního je doporučené či přikázané několika standardy a normami. Příkladem je standard ISO 27001, vyhláška š. 316/2014 Sb., o kybernetické bezpečnosti nebo doporučení od metodiky ITIL (IT Infrastructure Library).

Účelem testovacího prostředí je poskytnutí prostoru ať už pro zkoušení nově dodaných verzí systému, pro ověření integrace do prostředí společnosti, ale i zácvik zaměstnanců, jakožto nových uživatelů softwaru. Provádění takovéhoto testování na produkčním prostředí by znamenalo zásah do ostrého provozu a možnost způsobení nežádoucí změny nebo dokonce ztráty dat (Martin Šlancar, 2017)..

Testovací prostředí by mělo být shodné s produkčním, ale je tomu tak ve skutečnosti? Pro společnost by to znamenalo dvojnásobné náklady, nemluvě o nákladech za hardware, licence, velikost datových úložišť, síťovou infrastrukturu a lidské zdroje. V praxi jsou tedy činěny ústupky a prostředí je spouštěno na nejnižším počtu hardwarově slabších strojů. Dle zmíněného standardu a vyhlášky by měla mít každá organizace alespoň tři prostředí, a to prostředí vývojové, testovací a produkční (Martin Šlancar, 2017).

Mezi nejdůležitější testovací prostředí patří následující výčet:

- **Vývojové** (DEV – development) – toto prostředí spravuje tým vývojářů a jeho součástí jsou obvykle nástroje pro automatizované sestavování verzí, úložiště zdrojového kódu s podporou verzování a další nástroje pro vývoj. Co se týče samotných testů, je toto prostředí používáno k jednotkovým testům (Martin Šlancar, 2017).
- **Systémové** (SYS – systemtest) – jak z názvu plyne, slouží toto prostředí k systémovým testům. Do prostředí jsou nasazeny již hotové funkce, které prošly

jednotkovými testy. Systémové prostředí plní dvě zcela odlišně úlohy, první úlohou jsou finální testy vývoje, které provádí opět vývojáři a po jejich ukončení přichází úloha druhá, kdy se dodávka předává testovacímu týmu. Ten ji ověří pomocí továrních akceptačních testů (FAT) (Martin Šlancar, 2017).

- **Integrační** (INT – integration) – toto prostředí je určeno pro akceptování funkčnosti a pro samotný business. Provádějí se zde integrační a uživatelské akceptační testy (UAT) vč. end-to-end testů (testují průchod celým procesem nebo aplikací), ke kterým bývá přizván i koncový uživatel aplikace. K tomuto testování je nezbytné zajistit monitoring integrovaných aplikací, protože výpadek jedné aplikace může způsobit netestovatelnost závislých aplikací (Martin Šlancar, 2017).
- **Předprodukční** (PRE – preproduction) – u tohoto prostředí je na rozdíl od ostatních vyžadována shodnost HW a SW s produkčním prostředím, protože se zde provádějí nezbytné zátěžové a jiné provozní testy (Martin Šlancar, 2017).
- **Produkční a záložní** (PRO – production and backup) – posledním, nejvíce zmiňovaným prostředím, je již samotné produkční prostředí, které již plně podporuje reálné obchodní procesy společnosti. Někdy k němu patří i prostředí záložní, které se spustí v případě poruchy na produkci (Bureš, a další, 2016).

Když je připravené testovací prostředí, je nutné ho doplnit testovacími daty. Na ta je důležité klást velký význam, jelikož mají značný vliv na efektivitu testovacího procesu. Stejná sekvence jednotlivých kroků testu nad odlišnými daty vede testera k rozlišným výsledkům. Jejich správné vytvoření tedy znamená úsporu času a umožňuje zachovat stejný stav systému pro retest defektu (Bureš, a další, 2016).

Na následujícím obrázku je znázorněn proces správy testovacích dat, který začíná sběrem požadavků. Pro vytvoření testovacích dat je potřeba znát specifikaci. Požadavky na vytvoření mohou být z odlišných zdrojů, a to z testovacích scénářů, business testerů pro UAT nebo z informací o rozdělení dat v produkční verzi systému. Další fází je samotná výroba dat, která se může provádět různými způsoby. Data mohou být manuálně vytvořená neboli testerem smyšlená, automaticky vytvořená, dále to může být kopie produkčních dat beze změny nebo se změnou anebo mohou být vytvořená anonymizací citlivých produkčních dat. Předposlední fází je přidělení testovacích dat testerům a poslední, ale také důležitou částí, je monitoring kvality dat. Monitoring je prováděn kvůli zjištěním, zda šlo o data

kvalitní a znovupoužitelná nebo zda byla data nekvalitní a je potřeba je v příštích testech změnit (Bureš, a další, 2016).

Obrázek 6 - Proces správy testovacích dat (Bureš, a další, 2016)

K vytvoření testovacího prostředí na úrovni produkčního dnes existují virtualizační a cloudové technologie, které jsou schopné vytvořit klon produkčního prostředí, přesunout ji do testovacího prostředí s menší změnou konfigurace. Vytvoření samostatně fungující síťové infrastruktury k účelům testování usnadňuje technologie virtuálních LAN (Martin Šlancar, 2017).

3.2.9 Management testování

Test management je proces řízení testovacích aktivit, který si klade za cíl zajistit testování od začátku do konce projektového cyklu a vysokou kvalitu dodávané softwarové aplikace, která splňuje všechny požadavky zákazníka. Pomocí test managementu může test manažer pečlivě organizovat, řídit, analyzovat a sledovat celý proces testování a zaručit tak, že běží dle očekávání. Pro firmu to znamená lepší dodržování termínů, zvýšení týmové spolupráce a zjednodušení přidělování zdrojů projektu. Proces správy testování má dvě hlavní části, a to plánování a exekuci, které se dále dělí na nižší kategorie (Black, 2014).

3.2.9.1 Plánování

Během fáze plánování procesu testování se vytváří komplexní plán životního cyklu testování, jehož součástí jsou následující kategorie.

1) **Analýza rizik** – rizikem je nejistá událost nebo aktivita, která by mohla negativně ovlivnit průběh projektu, proto je prvním krokem test manažera provést před zahájením jakéhokoliv projektu analýzu rizik, jejichž včasná identifikace zabrání potencionálním ztrátám a ušetří náklady na projekt. Správná analýza rizik by měla obsahovat tři kroky – identifikaci možného rizika, analýzu dopadu identifikovaného rizika a opatření proti tomuto riziku. Riziko může být charakteru organizačního, obchodního nebo produktového. Pokud jsou zjištěna rizika, měla by mít určené parametry pravděpodobnost výskytu a dopad na projekt. Poté by se měla přijmout

jistá opatření ke zmírnění identifikovaného rizika za pomoci zvolení strategie. Během projektu by měla být rizika monitorována, aby nedošlo k nečekaným změnám (Hamilton, 2022).

2) **Odhad testování** – odhad testování je prognóza toho, jak dlouho bude trvat dokončení jednotlivých úkolů. Potřeba odhadnout není pouze čas, který je sice nejcennějším zdrojem projektu, ale také prostředky, které jsou nutné k provedení všech úkolů, to je pracovní síla, vybavení, zařízení nebo například financování. Dále je potřeba odhadnout lidské dovednosti, které znamenají znalosti a zkušenosti jednotlivých členů týmu. Posledním odhadem je odhad nákladů neboli rozpočtu projektu. Základními kroky odhadu jsou rozdělení celého úkolu na dílčí úkoly, přidělení jednotlivých úkolů členům týmu, odhad intenzity práce k dokončení každého úkolu a na závěr ověření odhadu neboli kontrola a schválení osobou k tomu určenou (Hamiton, 2022).
3) **Plán testování** – jedná se o podrobnou dokumentaci, která popisuje strategii testování, plán, odhad, výstupy a zdroje, které jsou potřebné k zahájení testování. Úkolem plánu je určit úsilí potřebné pro ověření kvality testovaného softwaru. Důvodem k vytvoření tohoto dokumentu je také pomoci lidem z různých směrů pochopit podrobnosti testování, dalším důvodem je, že se jedná o jakousi knihu pravidel, kterými je potřeba se při procesu testování řídit a poslední přidanou hodnotou je, že se mohou tyto cenné informace použít pro jiné projekty (Hamilton, 2022).
4) **Organizace týmu** – v této části je potřeba definovat role členů týmu a jejich zodpovědnosti v procesu testování. Je to jeden z nejsložitějších úkolů v managementu testování, protože testovací tým má důležitou roli v každém projektu vývoje softwaru. Pro každý tým je zásadní spolupráce mezi členy, závazek ke společnému cíli, efektivní komunikace a sdílení informací, znalostí a zkušeností pro vzájemné zdokonalování. Pro manažera jakožto vůdce je klíčové spravovat tým pomocí nastavení cílů, pozorování a řízení konfliktů (Hamilton, 2022).

3.2.9.2 Exekuce

Druhou fází test managementu je samotné provádění testů, které zahrnuje tři následující oblasti.

1) **Monitorování a řízení testů** – aby nedošlo k odklonění od plánu a překročení stanovených zdrojů nebo časového plánu je potřeba monitorovat a řídit testovací aktivity. Při procesu monitorování dochází ke shromažďování, zaznamenávání a reportování o projektové činnosti. Během monitorování test manažer definuje cíle projektu nebo standardu výkonnosti projektu, sleduje výkonnost, tzv. porovnává skutečný a očekáváný stav a zaznamenává a nahlašuje zjištěné problémy. Řízení testovacích aktivit je proces využití získaných dat z monitorování k opravě odchylek s plánovanými cíli (Hamilton, 2022).
2) **Správa problémů** – test manažer má za úkol identifikovat, nahlásit a vyřešit všechny problémy, které se vyskytnou během testování. Většina důvodů vzniku problémů jsou lidského charakteru a manažer testování by za ně měl převzít plnou zodpovědnost. Rizika, kterých je potřeba se vyhnout, jsou zmeškání termínu, překročení rozpočtu a ztráta důvěry zákazníka, proto je potřeba všechny problémy včas hlásit a eskalovat jejich vyřešení (Hamilton, 2022).
3) **Report a vyhodnocení** – po ukončení testování je potřeba podat report, který obsahuje důkladné vyhodnocení procesu testování. Tento report je shrnutím a analýzou výsledků testování pro všechny strany, které se podílejí na projektu (Hamilton, 2022).

Pro podporu správy testování se používají testovací nástroje, kterých je na trhu celá řada. Funkce testovacího nástroje může být prostá ve formě evidence testovacích scénářů nebo komplexní v podobě systému řízení celého životního cyklu vývoje softwaru. Cílem všech je však práci zefektivnit, zrychlit nebo dokonce umožnit. Dle ISTQB se nástroje dělí na základní kategorie:

- nástroje pro provádění testů nebo pro přípravu testovacích dat,
- nástroje pro správu požadavků, testovacích případů, testovacích procedur, automatizovaných testovacích skriptů, výsledků testů, testovacích dat a nástroje k reportování a monitorování provádění testů,
- nástroje k analyzování a vyhodnocování,
- nástroje jiného charakteru pomocné při testování (např. MS Excel) (Učební osnovy – Certifikovaný tester, 2018).

3.2.10 Defekt Management

Z předchozích kapitol vyplývá, že testování se v kostce zakládá na třech hlavních oblastech, a to na plánování, testování a reportování o nalezených výsledcích. Ačkoliv se může zdát reportování tou nejméně náročnou oblastí, ve skutečnosti se může jednat o nejdůležitější činnost, kterou tester provádí (Patton, 2002). Každý zjištěný defekt by měl být pečlivě přezkoumán a po celý svůj životní cyklus pečlivě sledován. To je důvodem pro společnost vytvořit proces defekt managementu, který bude mít za cíl vyřešit všechny zjištěné defekty (Učební osnovy – Certifikovaný tester, 2018).

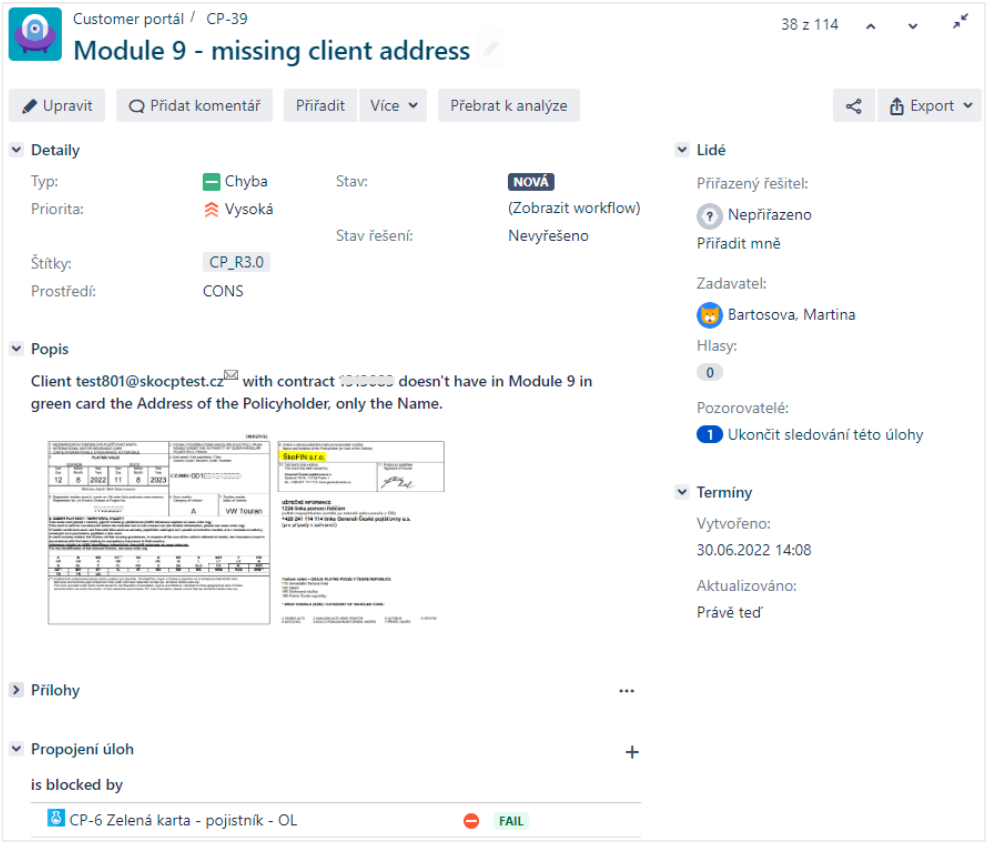
Nejprve je potřeba objasnit terminologii, protože následující pojmy bývají v praxi zaměňované.

- **Chyba** – příčinou vytvoření chyby je člověk při práci ve své oblasti – analytik, programátor apod.
- **Defekt** (bug) – důsledek chyby je defekt, jenž je odchylkou od očekávaného způsobu chování. Defekt je potřeba najít a opravit.
- **Selhání** (failure) – selhání systému nebo jeho části může nastat důsledkem defektu.
- **Incident** – v praxi je tento pojem používaný při nalezení defektu na produkčním prostředí, jeho náprava je dražší a mnohdy složitější (Bureš, a další, 2016).

S defekty přichází do kontaktu lidé různých rolí. Většinou první, kdo přijde do styku s defektem je tester, který defekt nalezne a zaznamená pomocí testovacího nástroje, do kterého zadá jeho podrobný popis. Úkolem testera je po opravě nalezeného defektu přetestovat místo, kde byl defekt nalezen a ověřit jeho správnou funkčnost. Analytik defektů je ten, kdo ověří, zda je zadaný defekt relevantní, odstraní opakující se defekty nebo vrátí zadaný defekt k doplnění kompletních informací o něm. Pokud je zadaný defekt relevantní zadá ho již k opravě odpovědným osobám. Další důležitou osobou, která se stará o defekty je test manažer. Jeho úkolem je dohlížet na celou správu defektů, eskalovat jejich řešení a informovat zainteresované strany o průběhu testování včetně nalezení defektů a jejích oprav. V neposlední řadě je vývojář, jehož úkolem je lokalizovat a opravovat zjištěné defekty a informovat o jejich opravě. A posledním zmíněným člověkem, jehož se defekty dotýkají, je projektový manažer, který má za úkol společně s vývojáři a test manažerem stanovovat priority oprav defektů (Bureš, a další, 2016).

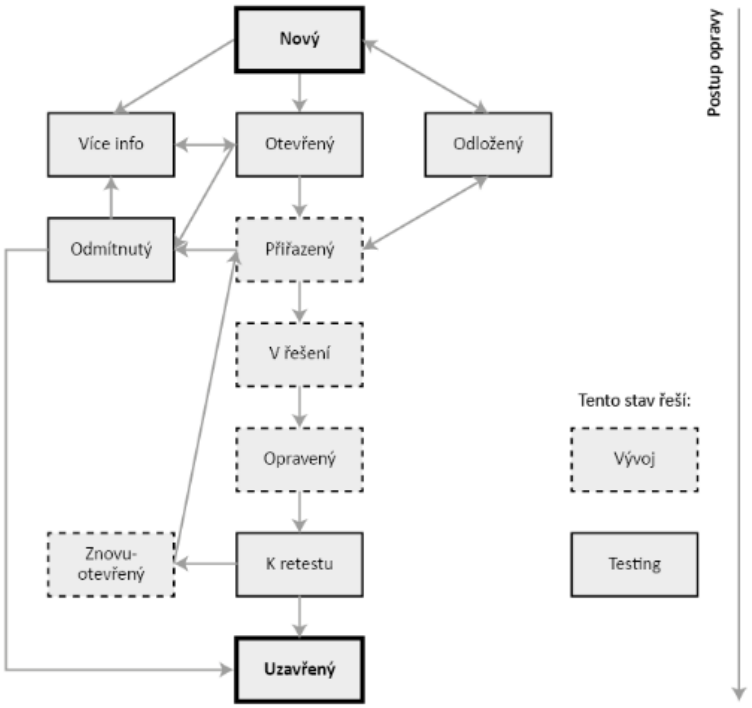
Při zadávání defektů do testovacího nástroje bylo zmíněno, že je potřeba dodat podrobný popis o nalezení. K tomu slouží atributy defektů, které pomohou s kategorizací a přetestováním po opravě. Základními atributy dle standardu ISTQB jsou:

- datum a čas nalezení
- autor
- očekávaný a skutečný výsledek
- identifikátor a prostředí, kde byl defekt nalezen
- podrobný popis zahrnující logy nebo snímky obrazovky
- míra dopadu na projekt či uživatelé
- závažnost dopadu na systém
- naléhavost (urgentnost) opravy
- priorita defektu
- stav defektu
- historie provedených změn
- odkazy na podpůrné dokumenty – testovací scénář, specifikaci apod. (Bureš, a další, 2016)



Obrázek 7 - Defekt zaznamenaný v systému Jira (Autor)

Defekt během svého životního cyklu vystřídá několik stavů, které jsou obvykle přednastavené v testovacím nástroji. Následující obrázek ukazuje algoritmus životního cyklu defektu. „Více info“ znamená, že defekt není dostatečně popsaný a je potřeba dodat důležité informace (Bureš, a další, 2016).



Obrázek 8 - Životní cyklus defektu (Bureš, a další, 2016)

Co se týče zmíněné priority defektu je často zaměňována za závažnost defektu, proto je nutné tyto pojmy vysvětlit. Závažnost je určována testery a jejich vnímáním, jak závažný defekt je. Zatímco priorita je určována test manažerem, projektovým manažerem a vývojářem a určuje pořadí, v jakém bude defekt opraven. Třetím pojmem je naléhavost neboli urgentnost defektu, kterou opět určuje tester a označuje jeho vnímání, kdy by měl být opraven. Pro upevnění terminologie bývá v každé organizaci definována vlastní škála, která tuto kategorizaci usnadňuje (Bureš, a další, 2016).

Defekt management je tedy důležitou součástí každého projektu, jelikož defekty mohou mít nespočet příčin a nelze se jim zcela vyvarovat. Důležité je odhalit defekt co nejdříve a dodržet správný postup a nalezené defekty formálně popsat, izolovat, přiřadit do kategorie, zapsat a sledovat proces řešení. Pokud má být tester efektivní, je potřeba testování naplánovat, a nejen vyhledávat defekty, ale také je vhodným systematickým a metodickým postupem nahlašovat. Špatně ohlášená nebo identifikovaná chyba by totiž nikdy nemusela být opravena (Patton, 2002).

3.3 Exploratory testování

3.3.1 Definice a cíle

Exploratory testování neboli doslovně průzkumné testování patří do skupiny neskriptového testování, ve kterém na rozdíl od skriptového neexistují žádné předem definované testovací případy, které je potřeba provést. Dle Jamese Bacha se jedná simultánní učení, návrh testu a jeho následné provedení. Exploratory testování není pouze technika nebo strategie, ale je to přístup k samotnému testování, ve kterém se návrh testu provádí při vykonávání testu, nikoliv před ním (Exploratory testing: A multiple case study, 2005).

Cíle průzkumného testování jsou popsány v následujících odstavcích.

- Prvním cílem je pochopení toho, jak aplikace funguje, jak vypadá její rozhraní a jakou funkcionalitu implementuje. Takovýto cíl si obvykle kladou testeři, kteří jsou v projektu noví nebo ti, kteří chtějí identifikovat vstupní body testu, konkrétní testovací výzvy a napsat testovací plány. Pochopit hloubku testovacích potřeb aplikace a nalézt nové neprozkoumané funkce je také cílem zkušených testerů (Whittaker, 2009).
- Dalším cílem je přinutit software ukázat své schopnosti, aby tvrdě pracoval a klást na něj velké nároky. Toto může ale i nemusí nalézt chyby, určitě to ale dokáže, že software plní funkci, pro kterou byl navržen a splňuje jeho požadavky (Whittaker, 2009).
- Posledním cílem je účelové hledání chyb. Specialitou průzkumné testera je prozkoumávat okraje aplikace a narážet na potenciálně slabá místa a identifikovat netestované a historicky chybné funkce (Whittaker, 2009).

Pro testery je Exploratory testování oblíbenou metodou díky proveditelnosti testování bez předem definovaných testovacích případů a jeho vedení výsledky z předchozích testů. Zaměřuje se na nalezení defektů pomocí průzkumu, takže lze předvídat a detekovat několik neočekávaných defektů, jenž při skriptovém testování mohou unikat. Charakteristika průzkumného testování je definovaná následujícími body (Exploratory Testing: An Overview, 2015).

- Testovací případy nejsou dopředu nadefinovány, testování je průzkumem s univerzálním posláním bez podrobných pokynů, jak jej splnit.

- Exploratory testování je řízeno výsledky testů, které bylo provedeny v předchozím testováním a poznatky, které na jejich základě byly získané.
- Toto testování se silně zaměřuje na odhalení defektů pomocí průzkumu narozdíl od tvorby testovacích případů, které jsou dále použity pro pozdější účely.
- Exploratory testování je současné učení testovaného systému, návrh testu a provedení testu.
- Efektivita průzkumného otestování je závislá na znalostech, dovednostech a zkušenostech testera (Exploratory Testing: An Overview, 2015).

3.3.2 Historie

Termín Exploratory testování poprvé představil profesor Cem Kaner a spol. v roce 1988 ve své knize Testování počítačového softwaru. V knize je tento přístup popsán jako prostředek k udržení testování po exekuci scénářových testů, pokud je software v nestabilním stavu a může být v blízké době předělán. V knize není zmíněno, co je součástí testování a jak by se mělo provádět (Exploratory Testing: An Overview, 2015).

Průzkum je dle autorů účelné putování prostorem, které má obecné poslání, ale nemá předem danou trasu. Představuje tak neustálé učení a experimentování. Podle nich je úkolem testera se neustále vzdělávat o produktu, o jeho trhu, rizicích a jeho předchozích chybách, aby mohl vytvářet stále nové a kvalitnější testy, než byly předchozí, jelikož jsou založeny na neustále narůstajících znalostech a dovednostech testera (Exploratory Testing: An Overview, 2015).

3.3.3 Rozdíl mezi skriptovým a Exploratory testováním

Scénářové testování	**Exploratory testování**
Využívá se v projektech, kde je dostatek času pro fázi testování.	Používá se v projektech, kde není dostatek času na plánování testování a vytváření scénářů.
K testování je nutná jasná a úplná specifikace požadavků.	K zahájení testování není potřeba žádná dokumentace, tester sám zkoumá a učí se.
Testeři testují dle kroků uvedených ve scénáři.	Testeři testují na základě svých dovedností a porozumění systému.

Testovací scénáře lze vysledovat k původním požadavkům a specifikacím, aby se prokázalo pokrytí testů.	V Exploratory testování neexistuje jasné a měřitelné pokrytí testů.
Aplikace je ověřena dle specifikace.	Aplikace je porovnávána s očekáváním a pochopením testera o tom, jak má aplikace fungovat.
Zpětná vazba je pomalejší.	Zpětná vazba je velmi rychlá.
Mohou být automatizované.	Nelze je automatizovat.
Na konci testování lze potvrdit, zda byly požadavky splněny či nikoliv.	Nelze potvrdit, že byly splněny všechny požadavky.
Nejsou vhodné pro neustále se měnící požadavky.	Je vhodné tam, kde se často mění požadavky.

Tabulka 1 - Scénářové vs Exploratory testování (Kuldeep, 2021)

3.3.4 Použitelnost

Důvody k tomu, proč tento přístup používat jsou jednoduché. V dnešní době je důležité uspokojit poptávku na trhu po kvalitních digitální prostředcích, které splní očekávání nejnáročnějších uživatelů. Rychlost uvedení na trh je důležitá, ale nesmí být důležitější než kvalita, protože chybný produkt může vést mimo poškození pověsti ke katastrofě za miliony dolarů až ke ztrátám na životech, jak bylo uvedeno v kapitole 3.2.2 (Parmar, 2022).

Většina testování je prováděna strukturovaně neboli se scénářem, kde jsou testovací případy vymezeny na základě již definovaných uživatelských scénářů. Co se týče testovacích dat, ta jsou strukturována podle předem určených testovacích případů. Exploratory testování ne. Během takzvaného prozkoumávání si testeři mohou „pohrát“ s uživatelským příběhem, jenž sleduje určitou posloupnost. Testeři vytvářejí dokumentaci za chodu a během testování si píší poznámky k chybám. Tímto způsobem se vytváří testovací případy (Parmar, 2022).

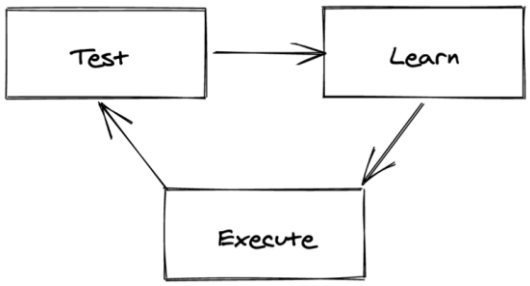
Efektivita bez vytváření testovacích kroků je podpořena podpůrnými nástroji, které se stávají předchůdcem automatizace. Nástroje s vizuální zpětnou vazbou a nástroje pro kolektivní testování přispívají tomu, aby se mohl kdokoliv zapojit do testování a okamžitě reagovat, a to je podpora agilních metod (Parmar, 2022).

Oblasti, které jsou vhodné pro průzkumné testování jsou popsány v následujících situacích, kdy je:

- nutnost rychlé zpětné vazby k nové funkci nebo aplikaci,
- potřeba se rychle aplikaci naučit,
- potřeba rozšířit sadu testů aplikace, která je již testována dle klasických testovacích scénářů,
- třeba v co nejkratším čase odhalit důležitý defekt,
- potřeba prošetřit stav konkrétního rizika (Exploratory Testing: An Overview, 2015).

3.3.5 Proces testování

Pro tento přístup neexistuje žádný daný přesný proces. Někomu se může zdát ad hoc, ale pro zkušeného a zručného testera se může stát mocným nástrojem. Zastánci jsou toho názoru, že Exploratory testování umožňuje využít plnou sílu lidského mozku při ověřování funkčnosti a vyhledávání chyb bez předem určených restrikcí (Whittaker, 2009). Průzkumné testování může být nazváno jako „freestyle testování“, ale lidi tento pojem odrazuje, protože dnes je trendem mít něco, co je zaručeně ověřeno (Danić, 2021).



Obrázek 9 - Proces průzkumného testování (Danić, 2021)

Proces tohoto testování je jinými slovy individuální a rozmanitý, a čím více jsou takoví testeři, tím zajímavější je testování a jeho výsledky. Rozmanitost zde může být zamýšlena následovně:

- každý člověk používá své zařízení jiným způsobem,
- každý má jedinečné zkušenosti a zázemí,
- lidé s rozlišnými profesemi používají rozlišný software a mají tak jiná očekávání (Danić, 2021).

3.3.6 Struktura Exploratory testování

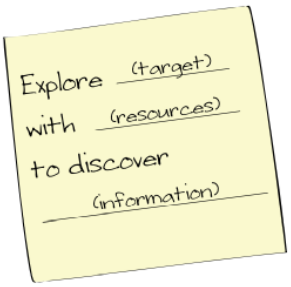
Exploratory testování má určitou strukturu s vnějšími a vnitřními dimenzemi. Vnější struktura je tvořena z jednotlivých prvků, kterými jsou čas, tester, produkt, poslání a reportování. Tester po určitou dobu interaguje s aplikací, aby splnil svou misi a nahlásil její výsledky. Při takzvaném plnění mise si tester představí řadu otázek o aplikaci, navrhne testy a provede je, aby získal odpovědi. Stav a výsledky testování jsou testerem důsledně reportovány. Vnitřní struktura existuje uvnitř mysli testera, jenž má následující vlastnosti:

- **Je návrhář testu** – být test designérem navrhující test, který systematicky zkoumá aplikaci.
- **Umí pečlivě pozorovat** – být obezřetným Exploratory pozorovatelem, který pozoruje cokoliv neobvyklého nebo tajemného a být schopný rozlišit mezi pozorováním a vyvozováním.
- **Má kritické myšlení** – být schopen posoudit a popsat svou logiku, která pomáhá v reportování stavu testování.
- **Má různorodé nápady** – vytváření nových nápadů použitím heuristiky jako jsou pokyny, obecné kontrolní seznamy, mnemotechnické pomůcky nebo orientační pravidla.
- **Má bohaté zdroje** – příprava seznamu nástrojů, informačních zdrojů a testovacích dat k využití při testování (Exploratory Testing: An Overview, 2015).

3.3.7 Charta

Není nutno vždy ověřovat celý systém, místo toho lze nastavit konkrétní cíle, které se označují jako charty. Konečným cílem testování je objevení informace, která je hodnotná pro zainteresované strany. Šablona základních prvků charty se skládá ze tří informací.

1) **Cíl** – kde je potřeba provést průzkum. Může se jednat o funkci, požadavek nebo modul.
2) **Zdroje** – jaké jsou dostupné. Ať už nástroje, datová základna, technika nebo konfigurace.
3) **Informace** – co je potřeba objevit. Jedná se o bezpečnostní chybu, výkonnostní, spolehlivosti, použitelnosti nebo designu? (Hendrickson, 2013)



Obrázek 10 - Šablona charty (cíl, zdroj, informace) (Hendrickson, 2013)

Takovým příkladem charty může být „Najděte způsoby, jak by zákazník nemusel dokončit platný nákup.“ nebo „Experimentujte s neplatnými hodnotami při registraci.“ (Exploratory Testing: An Overview, 2015).

Důvod k vytváření šablon je, aby byl zaměřen určitý cíl na zjištění informace nebo rizika. Je však důležité nezapomínat, že charta plní funkci šablony a je pouze vodítkem, není použitelná pro každý případ a situaci. Kvalitní charta nesmí být příliš specifická, jinak se stává jiným způsobem napsaným testovacím scénářem. U příliš obecných chart je však riskem, že nebudou dostatečně zaměřeny. Řešením je vytvořit několik chart, kdy každá cílí na jednu oblast anebo typ bezpečnostní mezery (Exploratory Testing: An Overview, 2015).

Vhodný čas k psaní chart je během diskutování o požadavcích na software, pak je jisté, že výsledky, které budou testováním zjištěné dosáhnou určité důležitosti. Během rozhovoru je důležité se ptát a se zainteresovanými osobami řešit své návrhy (Exploratory Testing: An Overview, 2015).

3.3.8 Techniky

Zde je zmíněno několik možných technik průzkumného testování.

3.3.8.1 Průzkumné testování ve volném stylu

V této technice nejsou použity specifické charty, tester volně zkoumá produkt tím způsobem, že se učí, navrhuje testy a provádí je. Jediný oficiální výsledek této techniky je reportování defektů. Toto takzvané „freestyle testování“ je řízeno dvěma způsoby, delegováním a účastí. Pokud je testování řízeno delegováním, vedoucí testů (test lead) specifikuje charty a testeři navrhují a provádějí testy, aby splnili chartu a následně podávají reporty. Druhý způsob řízení účastí znamená, že test lead testuje společně s testery, což eliminuje potencionální zmatky v testovacím týmu a společné úsilí tak přináší lepší nápady při odhalování chyb (Exploratory Testing: An Overview, 2015).

3.3.8.2 Session Based Test Management

Správa testů na základě relace je technikou, kterou konceptualizovali a aplikovali Jonathan a James Bachovi ve společnosti Hewlett Packard v roce 2000. Jedná se o metodu řízení testů založené na aktivitách, která je organizovaná pomocí relací. Je mnoho aktivit, které testeři provádějí mimo testování, například účasti na schůzkách, školení apod., které ale do testovací relace nepatří. Testovací aktivita v relaci lze rozdělit do 3 typů úkolů TBS.

- **Vykonání testu** (**T**est execution) – čas, který tester stráví průzkumem neboli provedením testu.
- **Hledání a reportování defektů** (**B**ug investigation and reporting) – aktivita, při které se vyšetřují a hlásí defekty, které přerušují průběh testu.
- **Nastavení a administrace** (**S**etup and administration) – aktivita, která je nutná ke splnění mise/charty a která předchází relaci nebo ji přerušuje. Zahrnuje návrh testu, konfigurace nastavení, čtení dokumentace, psaní reportu apod.

Testovací relace je tedy časové období, během kterého tester vykonává testování a má následující vlastnosti:

- **prakticky nepřerušovaná**
    - Během relace je nutné být minimálně rušený v podobě telefonních hovorů, e-mailů, chatování nebo schůzování.
- **zaměřena na konkrétní misi**
    - Testování má vždy jasné poslání nebo chartu, kde je specifikováno, co se testuje a jaké problémy je se hledají. Vše, co se odehrává při relaci, je výsledkem toho, že zodpovědný tester učinil řadu úsudků a rozhodnutí o splnění konkrétní mise nebo charty. Při hlášení se pak hodnotí, zda byla mise dokončena či nikoliv a pokud je to potřeba, relace se může opakovat.
- **může zahrnovat více testerů**
    - Pravidlem není, že se relace účastní pouze jeden tester, může tomu tak být nebo se může jednat o relaci skupinovou.
- **tester na konci podává výsledky o relaci**
    - Na konci každého sezení se koná jeho rozbor, který poskytuje představu o pokroku, jenž byl dosažen v testovací relaci. Report o sezení může obsahovat následující části:

        - charta
        - jméno testera
        - datum a čas zahájení
        - metriky TBS
        - datové soubory
        - poznámky k testování
        - problémy
        - defekty
- **provedení debriefingu s test manažerem**
    - Po dokončení relace je proveden tzv. debriefing, závěrečný skupinový rozbor mezi testery a test manažerem. Když se debriefingy nekonají, kvalita a podstata test reportů má tendenci trpět, protože testeři nemají šanci vyjádřit své důvody ostatním. Pokud tester hovoří o svém testování, často odhalí další informace, které by mohly být užitečné pro report. Tým má příležitost klást otázky a učit se. Smyslem tohoto je komunikovat mezi sebou o informacích, na kterých záleží, a učit se od sebe navzájem.

Jednou z hlavních výhod tohoto testování je, že testeři vytvářejí mnoho zpráv obsahující cenná data, která jsou užitečná pro projekt a jeho řízení. Tato technika, ostatně jako ty zbývající, lze implementovat mnoha způsoby, důležité je používat to, co dává smysl pro konkrétní testování (Selendic, 2021).

3.3.8.3 Thread Based Test Management

Správu testů na základě vlákna zavedl v roce 2010 James Bach. Vláknem je myšlen testovací nápad nebo testovací aktivita. Jedná se vlastně o zobecnění předchozí metody Session Based Test Management. Je to jednoduchý a rychlý způsob, jak začít, který není limitován časem. Cílem této techniky je uvedení testovacích nápadů jako vláken a jejich následné uspořádání do myšlenkové mapy. Vlákno je zde definováno jako soubor jedné nebo více činností, které jsou určené k vyřešení problému nebo dosažení konkrétního cíle (Exploratory Testing: An Overview, 2015).

3.3.8.4 Tour Based Exloratory Testing

Při průzkumném testování založeném na cestě průzkumný tester zdůvodňuje pět specifických vlastností, jako jsou uživatelské vstupy, stav, cesty kódu, uživatelská data a testovací prostředí. Při tomto procesu se tester rozhoduje o výběru mezi atomovými vstupy, uspořádáním atomových vstupů v kombinaci nebo sekvenci pomocí vstupních filtrů, používáním vstupních kontrol a obslužných rutin výjimek. Při rozhodování v oblasti interakce funkcí, datových toků a výběru cesty skrz uživatelské rozhraní používají testeři turistické metafory, kde se průzkum softwaru provádí pomocí nástrojů, jako jsou organizované výlety, průvodci, mapy a místní informace. Metafora „turista“ je vhodná pro Exploratory testování, kde se tester snaží prozkoumat novou oblast, vybírá kombinaci funkcí, a přitom provádí průzkumné testování podobné směsi orientačních bodů a míst, jak je vybírají turisté na prohlídce. Jak turisté rozdělují cíl na oblasti, jako je obchodní čtvrť, zábavní čtvrť, divadelní čtvrť atd. Průzkumný tester rozděluje vlastnosti aplikace na:

- **Obchodní čtvrť** – zde se nacházejí vlastnosti a funkce, na kterých závisí podnikání organizace.
- **Historická čtvrť** – cílem této části je otestovat funkčnost staršího kódu a ověřit spravené defekty.
- **Turistická čtvrť** – v této čtvrti se nachází vlastnosti a funkce, které přitahují spíše začínající testery než ty zkušené.
- **Zábavní čtvrť** – v této čtvrti se nachází podpůrné funkce jako je formátování textu.
- **Hotelová čtvrť** – zde jsou testovány sekundární a podpůrné funkce.
- **Pochybná čtvrť** – v poslední čtvrti se testují zranitelné části aplikace (Exploratory Testing: An Overview, 2015).

3.3.8.5 Hybridní techniky

První technikou je Scénářové a Exploratory testování, při kterém scénáře představují strukturu a Exploratory testování přidává variace ke zvýšení účinnosti kombinovaného přístupu. Při této technice se začíná klasicky pomocí scénářového testování, na které navazuje testování průzkumné (Exploratory testing: A multiple case study, 2005).

Druhá technika se nazývá Exploratory testování založené na scénáři. Zde se provádí průzkum pomocí existujících scénářů a podle potřeby se vkládají varianty. Tímto způsobem

se jeden scénář převádí do mnoha testovacích případů zvážením možností výběru dat, jejich využitím a podmínek prostředí (Exploratory testing: A multiple case study, 2005).

3.3.9 Pozorování

To, že je zobrazen očekávaný výsledek ještě neznamená, že je systém v pořádku. Systém lze přirovnat k ledovci, je možné vidět pouze špičku, ale co je pod vodou, není známo. To jest důvodem, proč je jednou z nejlepších dovedností Exploratory testera schopnost pozorovat detaily. Zní to jednoduše, ale není tomu tak. Kvalitní pozorování vyžaduje dívání se na to, co je očekáváno, aby bylo možné vidět, co je skutečně zobrazeno. To, co se skrývá pod ledovcem, je nutné sledovat pomocí použití konzolí, protokolů nebo monitorovacích aplikací. K ovládání aplikace je nutné používat privátní zobrazení a při jakémkoliv zvláštním chování je nutné být v pozoru. Pozorovatel se nesmí soustředit pouze na jednu věc, ale musí pečlivě sledovat různé aspekty softwaru a části obrazovky, jinak je pravděpodobné, že mu uniknou kritická překvapení, která musí neustále očekávat a záměrně hledat (Hendrickson, 2013).

3.3.10 Nástroje

Během Exploratory testování si musí tester pamatovat, co udělal, když našel defekt a jaké jsou jeho následky. Poznámky z procesu hledání chyb by mohlo postačovat zaznamenat do MS Excel, poznámkového bloku nebo pouze na papír. K usnadnění organizace, nahrávání a dokumentování testování je na trhu spousta nástrojů, které by byla škoda nevyužít. Tyto nástroje umožňují provádět návrh a exekuci testů současně bez velkého plánování a také přispívají k automatizaci procesu testování, ke spolupráci s týmy a k snadnějšímu reportování o defektech. V této kapitole bude představeno pár nejpoužívanějších (Medewar, 2022).

1) Testpad – jedná se o online nástroj sloužící k automatizovanému testování softwaru. Přístup přes jakýkoliv moderní prohlížeč je tedy bezproblémový a jednoduchý. Testpad poskytuje kontrolní seznam, jehož intuitivní rozhraní usnadňuje a urychluje proces testování a jeho flexibilní návrh testů pomáhá sledovat průběh Exploratory testování. Dalším benefitem je jeho bezproblémová integrace se systémem JIRA (Medewar, 2022).

2) PractiTest – tento nástroj je platformou založenou na SaaS (software jako služba) pro správu testovacích případů. Díky schopnosti spouštět průzkumné testy softwaru byl zvýšen zájem o PractiTest mezi vývojáři. Jeho funkcí je podpora požadavků na dokumentaci, poskytování mapování požadavků na testovací případy, dále pak vytváření, spouštění a úprava schopností testovacích případů, zaznamenávání defektů a jejich integrace do defekt managementu. Benefitem je bezproblémová integrace s aplikací JIRA, Jenkins, GitHub apod. Pro týmovou práci je to dobrá volba, protože usnadňuje spolupráci mezi členy týmu (Medewar, 2022).
3) Test IO – je jedním z nejoblíbenějších nástrojů pro průzkumné testování založených na TaaS (testování jako služba), ale je kompletním řešením pro testování kvality softwaru. Vhodným použitím je k testování webových stránek, mobilního testování, testování nositelného zařízení a IoT (internet věcí). Díky tomu, že se jedná o TaaS, umožňuje testování na několika zařízeních, operačních systémech a prohlížečích, jeho možnosti jsou takřka neomezené, kapacita a schopnosti jsou poskytované dle potřeby (Medewar, 2022).
4) Exploratory Testing Chrome Extension – zde se jedná o bezplatné rozšíření prohlížeče Google Chrome, které slouží přímo pro provádění Exploratory testování webových stránek. Tento nástroj umožňuje hlásit chyby, psát nápady a poznámky, pořizovat snímky obrazovky během relací a ukládat výsledky o relacích do sestav. Export relace je možný do různých formátů souborů jako je JSON, CSV nebo HTML (Rajkumar, 2022).
5) Session Tester – jedná se o open source bezplatný testovací nástroj pro průzkumné testování v operačních systémech Windows. Jeho cílem je spravovat a zaznamenávat proces testování na základě relací. Jednou z hlavních funkcí je časovač, pomocí kterého lze nastavit konkrétní délka testovací relace. Nabízí i poznámky ve formě XML, které lze převést například do HTML (Software.informer, 2022).
6) Zephyr – tento systém poskytující nástroje pro správu testů se zaměřuje na kvalitu a rychlost testování, díky čemuž zajišťuje výborné výsledky pro Exploratory testování. V systému JIRA pracuje nativně s nástroji Atlassian a mimo to nabízí jedním klikem integraci s dalšími systémy jako je Jenkins nebo například Bamboo. Zephyr je spolehlivou platformou pro profesionální analytiky, jenž umožňuje vytvářet, plánovat a spouštět testy v kratším čase, dále pak vytvářet testovací cykly,

provádět testy a spojovat defekty s konkrétními testy a sledovat metriky kvality. Zephyr je vhodným řešením pro jakkoli velké týmy, dle jejichž velikosti nabízí tři placené plány (Medewar, 2022).

7) Telerik Test Studio – tento průzkumný testovací nástroj je určený k testování webových, mobilních a desktopových aplikací, které jsou určeny pro všechny operační systémy Windows. Visual Studio integruje nástroj Test Studio, které je součástí kategorie nástrojů pro vizuální záznam umožňující testerům vytvářet testy na základě akcí nahrávání v prohlížeči. Zaznamenané testy lze dále rozšířit přechodem do testovacího skriptu a provedením pokročilých akcí v kódu. Jeho funkce spočívá v testování funkčnosti, zatížení a výkonu aplikací a s jeho pokrytím testů zajišťuje optimální kvalitu aplikace a řeší hlavní výzvy zajištění kvality softwaru (Hamilton, 2022).
8) Azure Test Plans – jedná se o sadu nástrojů pro manuální a průzkumné testování určené k testování ať už webových, tak i desktopových aplikací. Tento nástroj umožňuje současně testy navrhovat a provádět pro maximalizaci kvality v moderních procesech vývoje softwaru pomocí průzkumných testovacích relací (Azure).

9) Xray Exploratory App – tento nástroj může být použit samostatně jako webová, mobilní nebo desktopová aplikace pro Windows, Mac a Linux nebo může být přímo propojen s aplikací Jira. Pomocí aplikace lze nastavit testovací chartu nebo cíl relace, případně přidat podmínky a jakmile je testování započato, načasuje se relace. K nalezení defektů stačí tedy spustit relaci a prozkoumat systém z pohledu uživatele, zaznamenat důkazy pomocí videí, zvuků a snímků obrazovky a sdílet je s testovacím týmem (XRAY).
10) Test Rail – posledním zmíněným zde je komplexní nástroj s přívětivým uživatelským rozhraním, který je webovým softwarem pro správu testovacích případů a pomocníkem pro průzkumné testování. Opět je zde možná bezproblémová integrace s aplikací Jira, kam lze okamžitě odesílat hlášení o defektech pro zlepšení pokrytí a vybudování sledovatelnosti výsledků testů, aktivity a průběhu (Hamilton, 2022).

3.3.11 Výhody

Nejčastěji uváděným benefitem Exploratory testování je jeho schopnost zvýšit efektivitu testování z hlediska počtu a důležitosti nalezených defektů. V některých situacích, dle Jamese Bacha, může být průzkumné testování značně efektivnější než to scénářové. Tester se může zaměřit na oblasti softwaru, které se mu zdají být podezřelé, namísto toho, aby se věnoval pouze oblastem zmíněných ve specifikaci (Exploratory testing: A multiple case study, 2005).

Další je simultánní učení, jenž je při nedodržování určitého řádu pro testera velikou výhodou. Tester se aktivně učí o softwaru, který zkoumá a získává tak znalosti o jeho chování. Během testování pak tester přichází s lepšími a výkonnějšími testy (Exploratory testing: A multiple case study, 2005).

Třetím benefitem je možnost minimalizovat přípravu dokumentace před zahájením testování. Tato přednost se hodí zejména v situacích, kdy jsou neustále prováděny nějaké změny v požadavcích (Exploratory testing: A multiple case study, 2005).

Předposlední výhoda spočívá v možnosti provádění Exploratory testování bez komplexních požadavků nebo specifikační dokumentace. Je pouze na testerech, jak využijí své zkušenosti a znalosti softwaru, jenž získali z různých zdrojů (Exploratory testing: A multiple case study, 2005).

Poslední, ale neméně důležitá výhoda, je rychlý tok zpětné vazby z testování směrem k vývojářům. Testeři mohou opravdu rychle reagovat na změnu softwaru a výsledky rázem poskytovat nazpět vývojářům (Exploratory testing: A multiple case study, 2005).

3.3.12 Nedostatky

Jeden ze zjištěných nedostatků Exploratory testování je obtížnost sledování pokroku jednotlivých testerů a testování jako celku. K dispozici není žádná plánovaná struktura nízké úrovně, kterou by bylo možné použít pro sledování progresu, jenž by mapovala například pokrytí funkcí testování (Exploratory testing: A multiple case study, 2005).

Druhou slabinou je nemožnost detekovat nedostatky ve fázi návrhu jako je tomu u scénářového testování, které začíná odhalovat nedostatky již při fázi návrhu testovacích případů (Exploratory testing: A multiple case study, 2005).

Jedním z nedostatků je také potřeba zkušeného testera, který nepostrádá kritické myšlení. Tester nemá dostatek času k přípravě na proces testování, takže musí využít svých

znalostí a zkušeností. S tím souvisí i méně času na obeznámení se softwarem, což může být pro nové, a ne tolik zkušené testery problém a ti pak nemusí být schopni najít dostatek chyb (Kuldeep, 2021).

3.3.13 Východiska

Exploratory testování nabízí nespočet výhod, které silně převažují nad zmíněnými nedostatky, a proto je skvělým přístupem, který, ač není tradiční, tak je ale velmi užitečný a efektivní. Je vědecky dokázáno, že i nad mnohým převyšuje díky své efektivitě skriptové testování a dává testerovi svobodu prozkoumávat a ověřovat software svými způsoby a zkvalitňovat tak své dovednosti. Takový benefit dokazuje, že je Exploratory testování dobrým nástrojem pro detekci kritických chyb, který je zaměřen na kvalitu projektu bez přemýšlení o jeho přesné dokumentaci (Kuldeep, 2021).

4 Vlastní práce

4.1 Základní informace o zvolené společnosti

Zvolená společnost ŠkoFIN s.r.o., založená v roce 1992, je na českém trhu přední finanční společností v oblasti financování osobních a užitkových automobilů. Od září 2015 používá obchodní název Volkswagen Financial Services CZ, přičemž právní subjektivita zůstává nezměněna. Je součástí celosvětové skupiny Volkswagen Financial Services, která je obchodní divizí skupiny Volkswagen AG.

Společnost ŠkoFIN (Volkswagen Financial Services CZ) se specializuje na podporu prodeje dealerské sítě značek Volkswagen, Audi, SEAT, ŠKODA, Porsche, Ducati, koncernových ojetých automobilů všech značek a prémiových značek Bentley a Lamborghini na území České republiky.

Dále poskytuje úvěry pro financování nákupu osobních a užitkových vozů, operativní leasing a také značková pojištění vozů, servisní balíčky a další služby mobility soukromým osobám, podnikatelům i firmám.

4.2 IT struktura a aplikace ve společnosti

4.2.1 IT struktura ve společnosti

IT oddělení ve zvolené společnosti VWFS CZ je samostatný útvar, který zajišťuje klíčové IT služby pro chod celé společnosti. Organizačně je rozděleno na 3 úseky:

- **IT Operations** – zajišťuje všechny provozní služby z oblasti IT, tj. hardware, software, síťovou architekturu, přístupy v rámci aplikací apod.
- **IT Business Analysis and Security** – zajišťuje jednak přípravu, tj. analýzu požadavků na jednotlivé aplikace a systémy, dále pak má na starosti oblast bezpečnosti a compliance IT provozu dle nastavených standardů firmy.
- **IT Delivery** – zajišťuje přípravu systémové (technické) analýzy, realizaci dodávek nových úprav na aplikacích a systémech, které jsou obvykle dodávány externími dodavateli, a testování těchto dodávek.

Testovací tým je pak součástí oddělení IT Delivery, neboť je úzce spjatý s dodávkou úprav na jednotlivých aplikacích a systémech. Tým testování je metodicky veden Test and

release managerem, který společně s interním týmem IT test analytiků a externími spolupracovníky, odbornými testery a test analytiky, zajišťuje kompletní proces testování v rámci všech dodávek úprav aplikací a systémů.

4.2.2 Aplikace a systémy společnosti

Zvolená společnost ŠkoFIN s.r.o. provozuje několik desítek aplikací a systémů, které zajišťují podporu obchodního procesu a dalších neobchodních procesů společnosti. Lze je rozdělit do několika skupin:

- **Klíčové aplikace a systémy** – tyto aplikace zajišťují evidenci a zpracování obchodních případů, z hlediska testování je jim věnována největší pozornost, aplikace a systémy jsou mezi sebou integrovány. Jedná se tedy zejména o aplikace pro kalkulace a sjednání financování vozů a pojištění, evidenci smluvních vztahů s klienty, CRM (Salesforce), dealerské financování a datový sklad (DWH).
- **Klíčové aplikace pro finanční operace a účetnictví** – zajišťují správnou evidenci a zpracování oblasti účetnictví a financí. Jsou integrovány s klíčovými aplikacemi a systémy, např. SAP FI.
- **Aplikace zajišťující požadavky legislativy a třetích stran** – do této kategorie aplikací řadíme různé reportingy a hlášení např. pro rejstřík NRKI, hlášení pojišťovnám, zpracování silniční daně, poplatků za rádio atp.
- **Klientské aplikace a platformy** – jde o tzv. digitální kanály pro komunikaci s klienty (stávajícími i budoucími), např. webové stránky a formuláře, zákaznický portál, on-line proces sjednání financování, pojištění či doplňkových služeb.
- **Interní systémy** – HR systémy, evidence a schvalování objednávek a faktur, zpracování zpráv v datové schránce, evidence příchozí a odchozí pošty, Jira a tak dále.
- **Podpůrné a komunikační platformy** – aplikace zajišťující komunikaci mezi systémy, správa přístupů, technické nástroje a aplikace atd.
- **Běžný kancelářský software**.

4.2.3 Vývoj softwaru ve zvolené společnosti

Metodika vývoje softwaru ve společnosti Škofin s.r.o. se řídí příslušnou směrnicí a z důvodu provozování desítek různorodých aplikací (velikost, kritičnost, technologie atp.) nelze uplatnit na všechny aplikace jednotný přístup. Strategií společnosti je využívat moderní agilní přístup při vývoji aplikací tam, kde to dává smysl a kde je to možné, ať už ekonomicky, organizačně anebo technologicky.

Ve společnosti v posledních letech probíhalo a nadále probíhá celá řada školení na implementaci agilních metodik a postupně jsou prosazovány. Jak bylo zmíněno, dle Pattona neexistuje projekt, který by postupoval krok za krokem dle zvoleného přístupu a stejně tak tomu je i tady.

Většina dlouhodobých projektů, které jsou pro společnost klíčové, využívá především vodopádový model, kdy po prioritizaci businessových požadavků je vytvořeno zadání na vývoj nové funkcionality, případně aplikace, následuje detailní business analýza, poté systémová analýza a technická specifikace. Posléze je předáno dodavateli k vývoji. Jakmile je připravena kompletní analýza, je zahájena příprava testování, která zahrnuje standardní kroky jako je test strategie, test analýza a příprava testovacích dat. V rámci test strategie jsou identifikovány vhodné metodiky a techniky, které budou při testování konkrétních úprav využity a také jak konkrétně bude testování probíhat. Po dodání vývoje na testovací prostředí je zahájeno testování dle připravených kroků specifikovaných v test strategii a test konceptu. Nalezené defekty a chyby jsou standardně řízeny prostřednictvím defekt managementu (opravy defektů a retesty) a teprve poté je verze uvolněna pro nasazení do produkčního prostředí. Vzhledem k provázanosti aplikací a IT systémů je nutné velmi často koordinovat změny (vývoj testování a nasazení) na shodný čas.

Co se týče téměř čistého agilního přístupu, je na jeho základě vyvíjen například klíčový CRM systém, který má pravidelné dodávky v souladu s agilními metodikami. Nicméně i tento systém je závislý na integraci s dalšími klíčovými systémy, zejména v oblasti předávky dat, které využívají pro dodávky změn spíše vodopádový přístup, a tudíž musí být řešitelské týmy schopné oba přístupy propojit.

Většina aplikací využívá velmi často smíšený přístup, kdy obvykle vývoj změn na aplikaci probíhá v agilním přístupu, ale samotné zadání (tj. kompletní analýza) je připravováno vodopádovým přístupem a z důvodu zmíněného složitějšího propojení systémů je efektivnější koordinovat testování společně a s využitím vodopádového přístupu.

4.3 Testování ve zvolené společnosti

Testování v rámci zvolené společnosti je realizováno v souladu s nastavenými postupy, které jsou shrnuty v příslušné směrnici Test management. Tím je zajištěn jednotný přístup k problematice testování aplikací a tento přístup prochází pravidelnou revizí procesu a audity společnosti. V rámci směrnice jsou také definovány jednotlivé role a odpovědnosti, kterých je několik:

- Test manažer
    - řízení testování
    - plánování zdrojů
    - monitoring a řízení změn
    - zajišťování řádné dokumentace změn a problémů
    - koordinace časového rámce a testovacích aktivit
    - příprava test konceptu
    - zodpovídá za objednávání a plánování dostupnosti test. prostředí
    - kontrola výsledků dokumentování
- Test analytik
    - podílení se na strategii testování verze systému
    - zodpovědnost za přípravu testovacích případů
    - spolupráce na vytváření scénářů
    - sledování problémů a řízení změn při testování
    - hodnocení výsledků testu a konečná klasifikace testů
- Vedoucí IT dodávky
    - spolupráce s Test managerem, Test analytikem a Technickým vlastníkem systému a tvorba časového rámce pro proces testování v každém z testovacích prostředí
    - zodpovědnost za informování dodavatele o chybách či nedostatcích v programu, vedení aktuálního seznamu zjištěných nedostatků v programu a chyb
- Dodavatel
    - zodpovědnost za přípravu a řízení vývojového prostředí
    - zodpovědnost za dodávku požadovaných sestav a konfiguraci DEV
    - řešení chyb a testování scénářů, pokud je to vyžadováno, podpora testování

- Test metodik
    - zodpovědnost za přípravu testovacích scénářů
    - zodpovědnost za metodickou a obchodní správnost scénářů
    - zodpovědnost za řízení dat v rámci testování ve spolupráci s test analytikem
    - příprava žádostí o změny s pomocí test analytika
- Tester
    - zodpovědnost za používání definovaných testovacích nástrojů a provedení testu
    - dokumentování výsledků testu
    - opakované testování neúspěšných scénářů
- Vlastník procesu (produktu)
    - schvaluje kritéria přijetí
    - zodpovídá za jejich schválení

Stejně tak je zde ošetřeno, jakým způsobem je realizován defekt management. Nástrojem, který slouží k dokumentování každé aktivity související s projektem a také testování jakýchkoliv změn je aplikace Jira TestFLO. Tester je tak zodpovědný za zadávání chyby včetně co nejpodrobnějšího popisu. Test manažer sleduje stav testů během všech fází testování, což umožňuje centralizované a standardizované sledování závad a snadné podávání hlášení, včetně zpětné sledovatelnosti. Ve směrnici je také zmíněna definice chyby a problému. Závada softwaru je nedostatek v důsledku chyby a způsobuje jeho nesprávné chování. Defekty doplněné do Jira se nazývají problémy. Řízení problémů vyžaduje určitý typ kategorizace každého problému ve formě závažnosti a priority i určení, o jaký typ problému se jedná.

Součástí testovacího procesu jsou dále specializované testy, zejména performační a penetrační testy. Jsou také řízeny přístupy testovacího týmu na jednotlivá testovací prostředí aplikací a systémů (z důvodu bezpečnost nakládání s klientskými a obchodními daty atd.) stejně jako způsob pořizování testovacích dat z produkčních dat společnosti.

Proces testování je založen zejména na scénářovém testování. Pro jednotlivé úpravy či nové dodávky jsou vytvářeny testovací případy a testovací scénáře, které jsou následně exekuovány testovacím týmem odborných IT testerů nebo business testery. Součástí přípravy testování je také příprava vhodných testovacích dat. Testovací data jsou buď

vyhledávána v rámci dostupných testovacích dat na jednotlivých testovacích prostředích aplikací a systémů anebo jsou připravována nová specifická testovací data dle požadavků jednotlivých testů.

Obvykle jsou mimo vývojového prostředí na straně dodavatelů a produkčního prostředí k dispozici dvě úrovně testovacích prostředí pro každou z klíčových aplikací či systémů:

- **Nižší testovací prostředí** - tzv. INT – integrační testovací prostředí, kam směřují dodávky vyvinutých úprav aplikací a systémů a kde se ověřuje jejich úplnost, komunikace s integrovanými systémy a technická stabilita a výkonnost. Na tomto prostředí se používají anonymizovaná testovací data.
- **Vyšší testovací prostředí** - tzv. UAT – akceptační testovací prostředí, kam jsou nové úpravy posouvány po provedení testů na nižším testovacím prostředí (INT) a po odstranění většiny nalezených defektů (vždy musí být odstraněny defekty s vysokou a střední závažností předtím, než je dodávka posunuta z nižšího do vyššího testovacího prostředí).

V následující tabulce je porovnání testovacích prostředí, na levé straně je výtah ze standardu dle (Bureš, a další, 2016) a na pravé straně jsou prostředí používaná ve společnosti Škofin s.r.o. Vývojové prostředí je v zásadě na straně vývojářů (dodavatelských firem), kde provádějí jednotkové testy a FAT (Factory Acceptance Tests). Dá se říci, že integrační prostředí zde slouží jako systémové a integrační zároveň. Co se týče předprodukčního prostředí, to je zde zastupováno akceptačním prostředím, na kterém jsou prováděné uživatelské akceptační testy, performační testy, ale i některé integrační. Produkční prostředí je samozřejmostí, ale záložní prostředí zde neexistuje, data jsou samozřejmě bezpečně zálohována na několika místech.

Prostředí dle standardů			Používané prostředí		
Název prostředí	**Zkratka**	**Funkce**	**Název prostředí**	**Zkratka**	**Funkce**
Vývojové	DEV	unit testy, testy integrity verze	Vývojové	DEV	unit testy, FAT
Systémové	SYS	vývojové testy integrace, systémové testy, FAT	Integrační	INT	systémové a integrační testy, FAT, penetrační a performační testy
Integrační	INT	systémové integrační testy a UAT	Akceptační	UAT	UAT, performační testy, integrační testy
Předprodukční	PRE	zátěžové testy, technické akceptační testy, testy nasazení	Produkční	PROD	produkční prostředí
Produkční	PRO	produkční prostředí			
Záložní	BAC	záložní prostředí			

Tabulka 2 - Porovnání testovacích prostředí se standardy (Autor)

Dle charakteru jednotlivých testovaných úprav jsou do testování zahrnuty všechny dotčené aplikace a systémy. Podle rozsahu potřebných testů je pak vytvářen testovací tým a plánováno kompletní testování (test analýza, příprava testovacích dat, příprava testovacích prostředí, exekuce testů, defekt management) tak, aby bylo možné předat dodávku do produkčního nasazení v požadovaném termínu a kvalitě.

Vzhledem ke složitosti integrací mezi jednotlivými systémy jsou také na testovacích prostředích vytvořeny shodné integrace jako na produkčních prostředích v oblasti funkcionalit. Dále pak některé testovací prostředí mohou být vhodně upravené co se týče např. kapacity paměti, velikosti databáze apod.

Pokud aplikace nepatří mezi klíčové aplikace či není součástí složitější integrace, případně jde o standardní produkt dodavatele bez větších úprav pro zvolenou společnost, může být dostatečné pouze jedno testovací prostředí na úrovni UAT (např. Jira, HR systém).

4.4 Metodika Exploratory testování

Typů aplikací je celá řada, takže bylo nutné toto zohlednit v přípravě metodiky, která bude mít předpokládanou podobu a jejíž aplikovatelnost se ukáže až pomocí ověření při testování, které bude zahrnovat různé oblasti aplikací, pro které se Exploratory testování ukáže jako vhodné a bude vyzkoušen různý přístup k tomuto testování dle zvolené aplikace. Následně, pokud se testování prokáže za efektivní, bude sestavena konkrétní metodika pro zvolenou společnost.

Předpokládaná podoba metodiky Exploratory testování:

1) V první řadě bude určena oblast aplikací, kde dává Exploratory testování smysl, to bude součástí této práce.
2) Dále bude již při plánování testování určeno, zda je Exploratory testování pro konkrétní aplikaci vhodné.
3) V testovacím plánu bude určeno:
    a. technika, kterou bude provedeno Exploratory testování
    b. složení testerského týmu – zda se testování budou účastnit uživatelé anebo zkušení testeři
    c. nástroje – zařízení, které bude potřeba k testování, softwarové nástroje
4) Před začátkem testování je důležité připravit cíl testování – charty.
5) V průběhu testování je potřeba vedení test manažera či test koordinátora, který testování povede, bude testerům naslouchat, dělat si poznámky, případně bude mít připravený dotazník.
6) Po ukončení testování bude následovat meeting, na kterém test manažer shrne výsledky testů a případně nastaví pravidla pro další testování.

4.5 Zavedení Exploratory testování do procesu testování

V rámci změny organizace testování a metodického přístupu k testovacímu procesu jsou začleňovány další způsoby testování. Cílem této změny je zefektivnění celého testovacího procesu se zachováním či zvýšením jeho kvality. Jde především o realizaci automatizovaného testování pro opakované (regresní) testy na vybraných klíčových aplikacích a systémech a dále pak o efektivní začlenění Exploratory testování do procesu testování.

Exploratory testování je velmi často částečně intuitivně při testování jednotlivými testery používáno, obvykle však vychází ze zkušenosti testera s aplikací či systémem. Testování s použitím tohoto přístupu ale není standardizováno a evidováno, obvykle se tak skryje za scénářové testování. V rámci zlepšení a zefektivnění procesu testování bude tedy ověřen možný přínos zavedení Exploratory testování do standardního testování společnosti Škofin s.r.o.

Exploratory testování bude využito, na základě poznatků z teoretické části, při následujících testech:

- Funkcionální testy – ověřují očekávané chování/funkce,
- Nefunkcionální testy – ověřují mimo jiné použitelnost, která nám říká, zda je systém k uživateli přívětivý a dobře se s ním pracuje.

Předmětem této diplomové práce je tedy především vytvoření a začlenění metodiky Exploratory testování. Nicméně je potřeba hledat efektivní začlenění Exploratory testování do testovacího procesu a hledat vhodné části testovacího procesu, kde je možné toto testování efektivně využít.

Z tohoto důvodu byl uspořádán workshop s test manažerem a seniorními testery, na kterém byla na základě zkušeností a s pomocí brainstormingu vytvořena rozhodovací tabulka s přehledem aplikací, u kterých má smysl provádět scénářové testování, ale také je možné zavést Exploratory testování a popřípadě automatizované testy. Na základě této tabulky budou vytipovány konkrétní aplikace, na kterých bude provedeno ověření.

	Scénářové testování	Exploratory testování	Automatizované testy
Klíčové aplikace a systémy	✓	✓	✓
Aplikace pro finanční operace a účetnictví	✓	X	✓
Aplikace zajišťující požadavky legislativy	✓	X	✓
Klientské aplikace	✓	✓	✓
Interní systémy	✓	X	X
Podpůrné aplikace	✓	X	X

Tabulka 3 - Přehled testování pro oblasti aplikací (Autor)

4.5.1 Využití Exploratory testování v procesu testování – výzkumné otázky

Jak bylo zmíněno výše, testovací proces byl založen zejména na scénářovém testování ve všech fázích testování. Toto se ukazuje jako ne vždy efektivní postup, zejména z časových a kapacitních důvodů. Exploratory testování bylo vybráno jako nový přístup pro zvolenou společnost, který by mohl vést ke zlepšení procesu testování.

Hlavní výzkumná otázka a doplňující otázky:

- **Je Exploratory testování efektivní pro podnikovou praxi?**
    - **Je vhodným doplňkem ke scénářovému testování?**
    - **Může toto testování provádět obyčejný uživatel, tzv. tester bez znalosti aplikace nepodílející se na jejím vývoji?**
    - **Je vhodné pro rychlé ověření dodávky zkušeným testerem, který aplikaci zná?**
    - **Je vhodné jako doplněk k automatizovaným regresním testům?**

K zodpovězení položených otázek je potřeba na základě předchozí tabulky vybrat různorodé aplikace, na kterých bude provedeno ověření vhodnosti. Následující oblasti a aplikace možné přínosy ověří:

- **Oblast 1:** Ověření vhodnosti a úplnosti navrženého řešení z pohledu klienta v rané fázi dodávky aplikace – testy běžnými uživateli, kteří se nepodíleli na definici a analýze úprav či nových funkcionalit.
    - **klientská aplikace – Zákaznický portál**
- **Oblast 2:** Rychlé ověření dodané úpravy či funkcionality zkušeným testerem, obvykle z řad procesních či business analytiků, příp. seniorních testerů, dle standardizovaného postupu
    - **klíčový systém – Datový sklad**
- **Oblast 3:** Podpora automatizovaných regresních testů aplikace.
    - **klíčová aplikace – Náběr pojištění.**

4.5.2 Ověření využití Exploratory testování

Implementace tří vytipovaných oblasti uvedených v předchozí kapitole byly ověřeny samostatně formou "Proof of Concept", jak je tomu uvedeno v následujících podkapitolách.

4.5.2.1 Oblast 1: Ověření vhodnosti a úplnosti navrženého řešení z pohledu klienta/uživatele

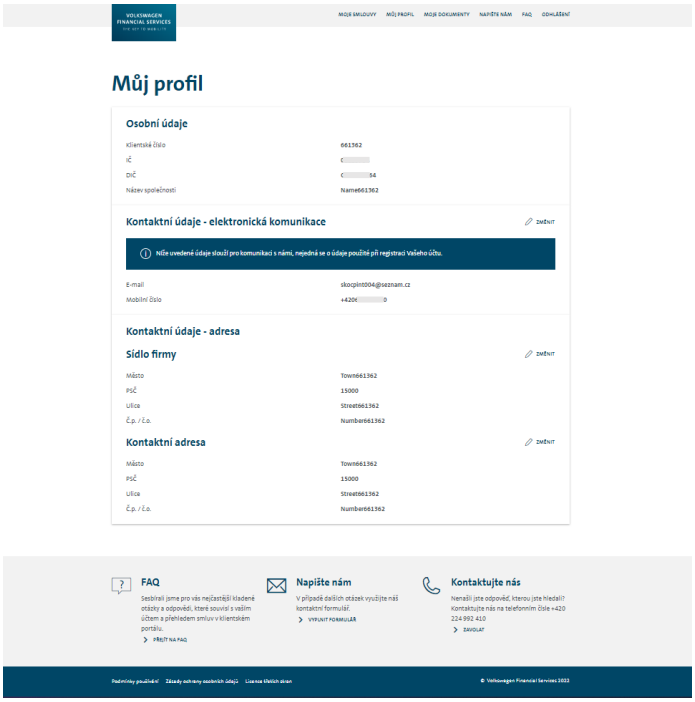
Ověření vhodnosti a úplnosti navrženého řešení z pohledu klienta nebo uživatele v rané fázi dodávky aplikace byla odzkoušena na klientské aplikaci Customer Portal – Zákaznický portál. Tento portál je vyvíjen v rámci digitalizace, kdy je pro klienta pohodlnější a jednodušší spravovat smlouvu online, klienti si tak mohou zobrazit přehled svých smluv, jednotlivých splátek, zažádat o změnu ve smlouvě nebo si například stáhnout zelenou kartu svého vozidla. Ověření proběhlo pomocí dvou testování, a to testování dodávky pro úpravy změn kontaktních údajů a testování pilotní verze chatbota.

A) Dodávka pro úpravy změn kontaktních údajů

Dodávka pro release 4.0, konkrétně část dodávky: Úprava změny kontaktních údajů u všech typů klientů. Všemi typy klientů rozumíme fyzické osoby nepodnikající (FON), fyzické osoby podnikající (FOP) a právnické osoby (PO).

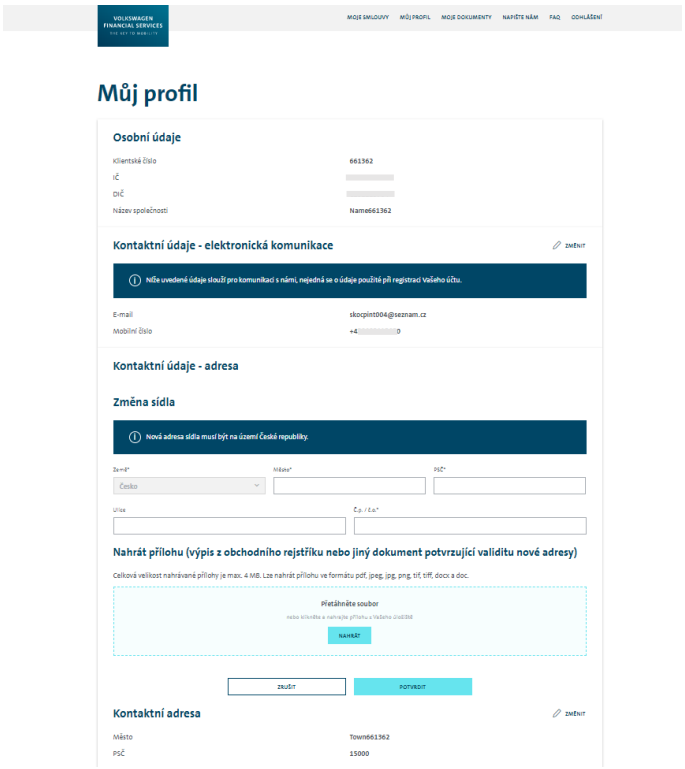
Testované změny spočívají zejména v tom, že klientovi je po přihlášení do Zákaznického portálu umožněno požádat o změnu jeho kontaktních údajů – e-mailu, telefonního čísla, trvalé adresy či adresy sídla firmy a kontaktní adresy, a to přímo v Zákaznickém portálu. Do této doby klient mohl o tyto změny žádat buď prostřednictvím formuláře na webových stránkách nebo kontaktováním klientského centra společnosti.

Na následujícím obrázku je zobrazen profil klienta na integračním prostředí INT, kde jsou data anonymizována, některá data jsou však upravena pro účely testování (e-mail, telefonní číslo). Klient se po přihlášení svou e-mailovou adresou a zvoleným heslem dostane na úvodní stránku, na které se nacházejí údaje o jeho smlouvách. Pro změnu kontaktních údajů, která byla testována, je zapotřebí přesunutí do záložky Můj profil, na které jsou již kontaktní údaje klienta zahrnující elektronickou komunikaci, trvalou adresu, respektive sídlo firmy pro osobu právnickou a adresu kontaktní.



Obrázek 11 - Profil klienta CP na INT prostředí (Autor)

Následující obrázek zobrazuje jednu z možných žádostí o změnu - detail změny sídla firmy právnické osoby. Pro odeslání změny je potřeba změnit alespoň jeden údaj oproti původním hodnotám a nahrát dokument potvrzující tuto změnu (tento krok není nutný u změny elektronických kontaktních údajů a kontaktní adresy).



Obrázek 12 - Detail změny sídla firmy na INT prostředí (Autor)

Pro danou funkcionalitu byly zpracovány testovací scénáře na základě analytické dokumentace pro realizaci standardního scénářového testování. Dále pak byl připraven set testovacích dat klientů pokrývajících všechny možnosti (FON, FOP, PO).

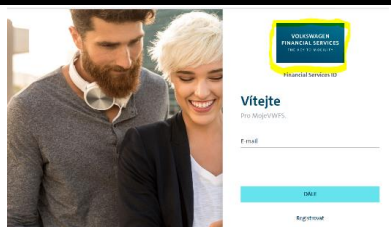
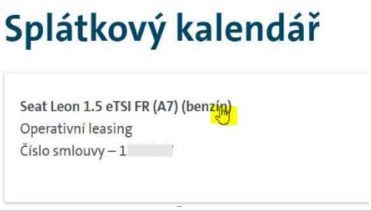
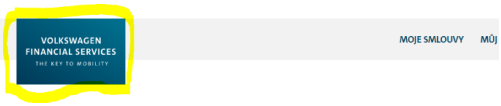
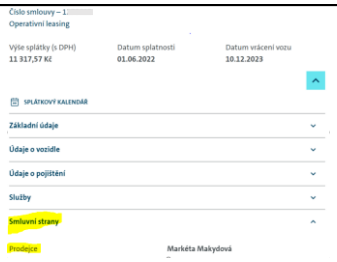
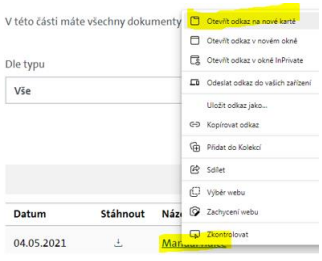
Porovnání obou metod testování, scénářové testování a Exploratory testování, proběhlo ve zvolené společnosti v květnu 2022 v rámci standardního testování. Testování se zúčastnilo celkem 9 zaměstnanců – testerů, kteří neměli žádnou nebo jen minimální znalost či zkušenost s danou aplikací. Ověřovalo se tedy, zda uživatelé bez znalosti zadání z dokumentace, ale pouze se základní informací o funkcionalitě, jsou schopni efektivně tuto funkcionalitu otestovat a zda se potvrdí předpoklad, že vhodným využitím Exploratory testování budou schopni odhalit jak shodné defekty jako při scénářovém testování, tak ideálně také skryté chyby či nesrovnalosti, se kterými by se mohl setkat běžný uživatel.

Testy probíhaly bezprostředně po první dodávce funkcionality na testovací prostředí, v tomto případě na UAT testovacím prostředí. Testeři byli rozděleni na dvě skupiny:

- Skupina A – testovala přesně podle kroků jednotlivých testovacích scénářů, bez časového omezení.
- Skupina B – testovala bez testovacích scénářů, tj. metodou Exploratory testování na základě techniky Session Based Test Management. Skupina byla seznámena pouze se základním ovládáním aplikace a businessovým zadáním funkcionality změny kontaktních údajů a pro testování měla stanovený časový limit 1,5 hod.

Ověřovalo se tedy, zda je možné efektivně použít metodu Exploratory testování pro otestování dodávky nové funkcionality bez podrobné znalosti dokumentace a zda tato metoda přinese očekávaný výstup. Očekáváním bylo, že využitím této metody testeři identifikují logické chyby v procesu vzniklé buď implementací funkcionality vývojem anebo nedostatky v analytickém či businessovém zadání oproti očekávání běžného klienta. Dále pak bylo zjišťováno, zda z pohledu běžného uživatele zde nenarazí uživatelé na nějaké uživatelské problémy v UX návrhu.

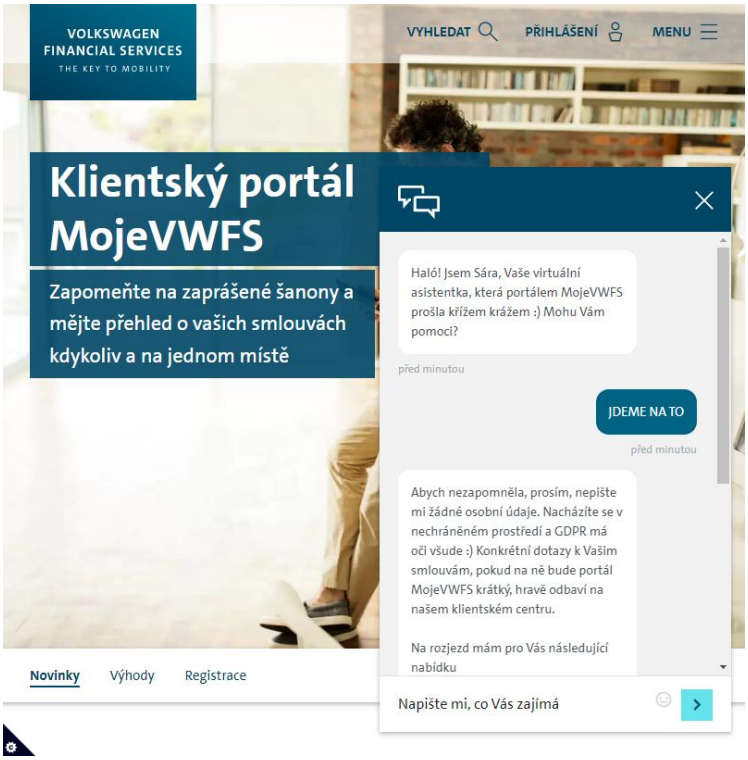
Výstupem testování je následující tabulka, kterou tvoří soupis defektů a změnových požadavků nalezených uživateli. Defekty a požadavky na změny byly nalezené navíc oproti scénářovému testování pomocí metody Exploratory testování. Některé defekty, jako je například očekávaný proklik loga, se několikrát opakovaly, přitom by se v běžném scénáři kontrola loga nevyskytovala.

Poř. číslo	Popis defektu/ změnového požadavku	Snímek obrazovky
1.	Logo VWFS na přihlašovací stránce není prokliknutelné, očekáváno je přesměrování na stránku Moje VWFS.	
2.	V sekci "Moje smlouvy", u položky "Celkový dluh na smlouvě" není jasné, zda se jedná o dluh po splatnosti či o standardní splátku se splatností v budoucnu.	
3.	Při otevření splátkového kalendáře se komponenta s informacemi o vozidle, po najetí myši, tváří jako odkaz/tlačítko, ale nejedná se ani o jedno.	
4.	Logo VWFS ve všech záložkách není prokliknutelné.	
5.	U přehledu smlouvy se u smluvní strany zobrazuje prodejce, očekáván je spíše název dealerství, případně tedy jméno prodejce. Samotný prodejce nemá vypovídají hodnotu o tom, kde byl vůz pořízen i pro případný návrat k dealorvi pro nákup dalšího vozu nebo reklamaci.	
6.	V zápatí v sekci "Kontaktujte nás" je telefonní číslo rozdělené na 2 řádky.	
7.	V sekci "Moje dokumenty" se při otevření odkazu v novém okně dokument neotevře ani nestáhne, ale zduplikuje se stránka "Moje dokumenty".	

Tabulka 4 - Přehled bugů/CR CP (Autor)

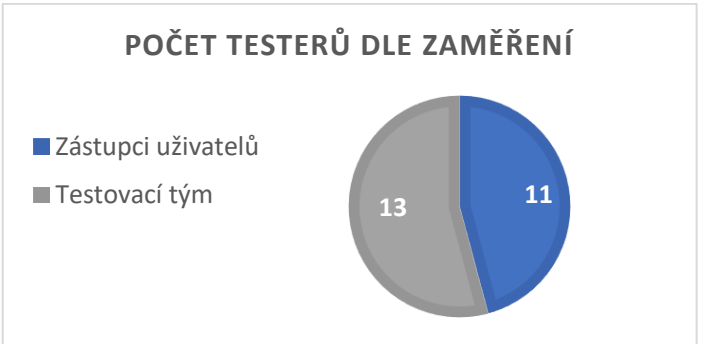
B) Nasazení Chatbota

Druhé ověření vhodnosti použití Exploratory testování bylo provedeno při nasazování pilotní verze chatbota, jehož úkolem je pomoci s registrací do klientského portálu a odpovědět na základní otázky ohledně jeho funkcionalit. Chatbot byl vytvořen s využitím technologie společnosti Mercury.ai, jenž má za úkol pomoci budovat lepší vztahy se zákazníky za použití živých zpráv a umělé inteligence s aktivním učením.



Obrázek 13 - Chatbot v testovacím prostředí (Autor)

Testování bylo rozděleno na více bloků dle preferovaných termínů a skupin a probíhalo na základě techniky Session Based Test Management. Testování se zúčastnilo celkem 24 osob, a to z řad zaměstnanců jakožto zástupců klientů, které doplnila skupina zkušených testerů. Z tohoto důvodu budou výsledky porovnány a zhodnoceny za jednotlivé skupiny.



Graf 1 - Podíl testerů na testování chatbota (Autor)

K testování byla potřeba několika málo nástrojů, nastavené testovací prostředí, pracovní notebooky anebo vlastní či připravené mobilní telefony s operačním systémem Android a iOS připojené do interní sítě.

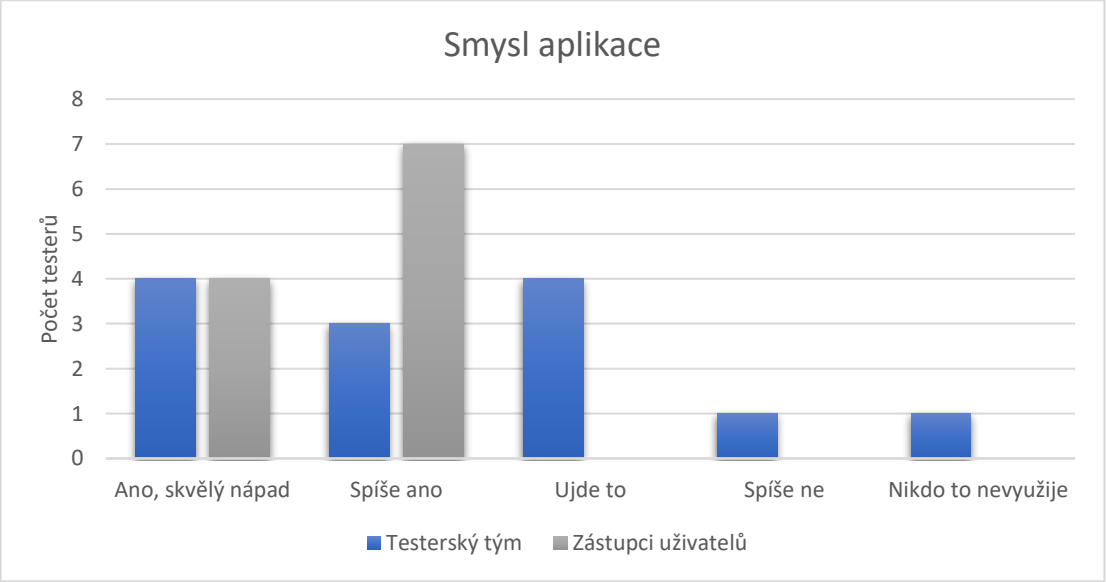
Testování probíhalo tedy formou relace, která měla nastavený časový limit na 30 minut. Nejdříve byla představena aplikace chatbot, její účel a funkcionalita, a následně mohlo začít testování. Během samotného testování probíhala diskuse pod dohledem vlastníka produktu, jenž si zapisoval poznámky a ke konci byl rozeslán anonymní elektronický dotazník.

Zaslaný formulář se skládal ze čtyř otázek, které budou představeny níže. Dvě otázky měly na výběr připravené možnosti a dvě se skládaly z otevřených odpovědí. Čas, který na vyplňování formuláře strávil průměrně jeden zástupce ze skupiny byl (uvedeno v minutách):

- 04:56 – zástupci uživatelů
- 04:08 – testeři z testovacího týmu

1) Dává ti aplikace smysl?

Na první otázku byly vcelku pozitivní odpovědi. Převážné většině dává aplikace chatbot smysl a hodnotí její využití velmi kladně. Na následujícím grafu lze vidět, že zástupcům uživatelů se chatbot opravdu líbil a našly by jeho využití, testovací tým byl však trošku na vážkách.



Graf 2 - Počet testerů, kterým dává aplikace smysl (Autor)

2) Je provázanost odpovědí a rozcestníků logická?

Co se týče druhé otázky, panují zde jisté obavy obou skupin nad logikou a provázaností s odpověďmi chatbota.

Graf 3 - Počet testerů, kterým připadá provázanost logická (Autor)

3) Co se ti na Sáře (chatbotovi) líbí nejvíce?

Třetí otázka byla mířená trochu jinam, kromě nesrovnalostí a případných defektů, je potřeba zjistit, co je na chatbotovi přidanou hodnotou pro uživatele, tj. klienta. Bylo zde zajímavé sledovat různé názory obou skupin, kde na testerské straně panovala spíše racionalita odpovědí v podobě rychlosti odezvy, ale na straně uživatelů převažovaly pozitivní názory na milé vyjadřování a kreativní odpovědi doplněné emotikony. Shrnutí všech odpovědí je následující:

- vtipné odpovědi vč. emotikonů
- schopnost reakce i na psané zprávy (ne pouze klikání předpřipravených odpovědí)
- rychlost a pohotovost reakce
- stále se učí – umělá inteligence
- nápomocná starším lidem
- velký potenciál
- dobrý nápad, který ušetří práci klientskému centru
- kreativní odpovědi
- klasický a intuitivní design chatbota

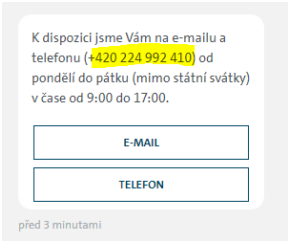
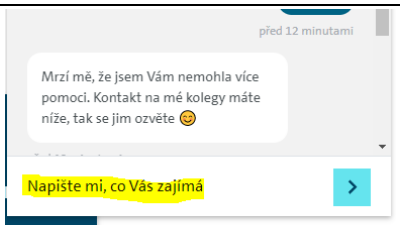
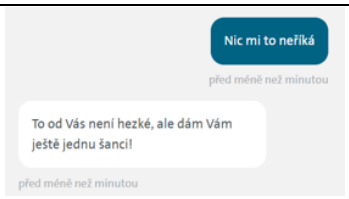
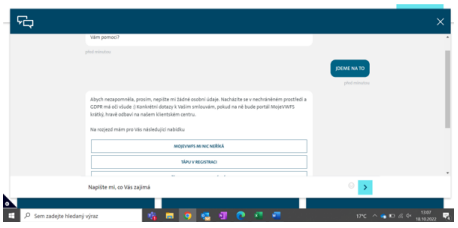
- rychlé dohledání potřebných informací bez telefonování
- poskytuje užitečné odkazy

4) Kde má Sára (chatbot) slabinu?

Nejdůležitější částí je samotné hledání chyb a nesrovnalostí, kterých nebylo málo. V první řadě bylo téměř všemi zjištěno, že Sára nereaguje na významná klíčová slova, jako jsou zelená karta, technický průkaz nebo například smlouva, což je závažný problém. Dále někteří z testerů brali jako nešvar právě zmíněné používání emotikonů. Části skupiny testerů toto přišlo jako něco zbytečného, zatímco skupina uživatelů to v předchozí otázce vychválila. Shrnutí všech odpovědí je popsáno opět v bodech.

- nereaguje na klíčová slova
- nereaguje na většinu dotazů/vět nebo odpoví jinak, než by měla
- není připravená na nasazení na produkci
- odpovědi občas nesouhlasí s dotazem
- příliš obsáhlé věty, nutnost rolovat
- potřeba se stále učit, aby mohla být vypuštěna do světa
- očekáváné byly spíše dotazy na produkty, nikoliv na portál
- měla by odkazovat na jiné oblasti než jen CP
- neodpovídá na výrazy, které má sama ve slovníku
- možnost pouze českého jazyka

Během testování probíhala zmíněná diskuse a vlastník produktu sbíral data. Někteří sdělovali své pocity rovnou, někteří si to nechali až do formuláře. Část testerů také zaznamenávala snímky obrazovky a následně je i s komentářem zaslala. Přehled těch nejdůležitějších je znázorněn v následující tabulce:

Poř. číslo	Popis defektu/ změnového požadavku	Snímek obrazovky	Stav
1.	Po napsání klíčového slova "kontakt" byly zobrazeny tlačítka "E-mail" a "Telefon", kdy telefon je použitelný pouze pokud uživatel chatuje z mobilního telefonu. Telefon by se měl zobrazovat i v odstavci.		opraveno
2.	Na mobilu se špatně zobrazuje nabídka emotikonů.		opraveno - vypnuta možnost emotikonů
3.	Barva písma by měla být v textovém poli světlejší, působí jako již napsaný text.		zadáno k opravě
4.	Na iPhonu SE je špatně zobrazené textové pole.		zadáno k opravě
5.	Některé výrazy jsou špatně vyložené.		zadáno k opravě
6.	Nedokáže reagovat, když se místo vybrání možnosti, možnost napíše.		zadáno k opravě
7.	Na neaktuální verzi prohlížeče Google Chrome se na desktopu zobrazuje okno chatbota roztažené přes celou obrazovku.		zadáno k opravě
8.	Na mobilním telefonu nefunguje tlačítko pro odeslání zprávy.		zadáno k opravě

Tabulka 5 - Přehled bugů/CR chatbota (Autor)

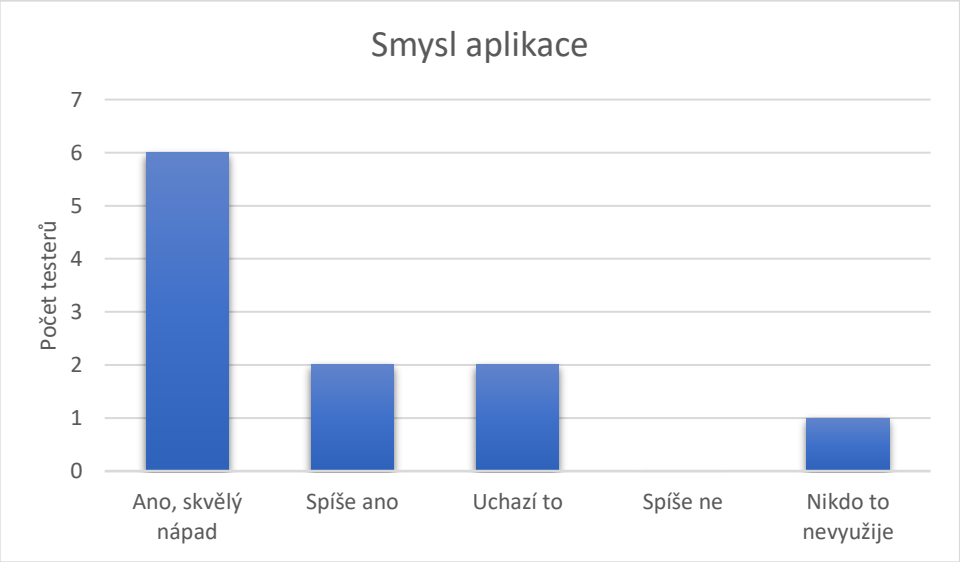
Některé defekty a změnové požadavky byly ihned opraveny, jiné byly zadány k opravě. Pro vlastníka produktu byla tato sezení velice cenným výstupem, díky kterého se dozvěděl, co je potřeba přidat, upravit nebo dokonce odstranit. Rozhodovací strom odpovědí byl hned následující dny přepracován a doplněn, aby bylo možné vést s chatbotem kvalitnější konverzaci. Termín nasazení na produkci byl na základě tohoto výstupu posunut a bylo naplánováno další kolo, ve kterém se otestují chystané změny. Nejen členy testovacího týmu, ale lidmi z business prostředí byla tato forma testování velice chválena.

Během následujících dvou týdnů byla aplikace přepracována a bylo uspořádáno 2. kolo testování chatbota. V tomto kole byl změněn přístup a na místo sezení (tj. osobní účasti testerů v místnosti pro testování) byla rozeslána pozvánka všem zúčastněným na testování online formou a připraven jeden dotazník namísto dvou (zvlášť pro testery a pro uživatele), který měl téměř totožnou podobu jako dotazník v 1. kole z důvodu možného porovnání.

Dotazníků se vrátilo 11, kde průměrná doba vyplňování (v minutách) jednoho dotazníku byla 7:49. Otázky byly tentokrát pouze tři, protože byla vynechána otázka, co se testerům na chatbotovi líbí nejvíce, tu již bylo zbytečné pokládat.

1) Dává ti aplikace smyl?

V porovnání s 1. kolem testování zde vychází nejčastěji první odpověď, a to ta, že v případě šesti odpovědí, má aplikace smysl a je skvělým nápadem. Jednomu člověku však stále připadá zbytečná.



Graf 4 - Počet testerů, kterým dává aplikace ve 2. kole smysl (Autor)

2) Je provázanost odpovědí a rozcestníků logická?

Druhá otázka, která je i vcelku klíčová, vychází tentokrát velice dobře. Oproti minulému kolu, kdy převážná většina hlasovala pro váhavé odpovědi „Někde ano, někde ne“ a „Ani moc ne“, nyní většina testerů odpověděla, že si myslí, že je provázanost logická. Pouze dva testeři si myslí, že někde ta logika stále chybí.

Graf 5 - Počet testerů, kterým připadá provázanost 2. verze logická (Autor)

3) Vidíš nyní nějaké slabiny nebo nesrovnalosti při chatování se Sárou?

Poslední otázka, kde se mohli respondenti rozepsat, tentokrát nedopadla vůbec špatně. Většina testerů vychválila pokrok oproti minulému testování, chatbot nyní odpovídá na většinu klíčových slov a dokáže navést, kam klient potřebuje, avšak nějaké nesrovnalosti se přeci jen našly.

- Neustálé scrollování – Sára píše několik zpráv za sebou.
- Ještě nezná pár slov, na která se mohou klienti zeptat.
- Na angličtinu nyní reaguje, ale nelze přepnout do jiného jazyka.
- Na některé výrazy odpovídá jinak, než by měla – záměna proč/jak.

Testování dopadlo úspěšně a dle očekávání, dalo chatbotovi směr, kterým se ubírá a chatbot Sára se připravuje na produkční nasazení do konce listopadu 2022 do pilotního provozu. Tam bude již konfrontována s dotazy reálných klientů, kteří jí budou dále testovat a aktivně učit další slovní spojení. Po vyhodnocení pilotního provozu se předpokládá realizace drobných zlepšení v logice aplikace a uvolnění do plného provozu.

Identifikované problémy, doporučení a příležitosti pro zlepšení v oblasti 1: Ověření vhodnosti a úplnosti navrženého řešení z pohledu klienta/uživatele

Pokud má být do testování zapojeno více osob, ať již zaměstnanců či v budoucnu i klientů, kteří s testováním nemají hlubší zkušenost, je potřeba věnovat čas přesné přípravě zadání a postupů a dopředu ověřit, zda jsou opravdu všichni schopni se na dané prostředí bez problémů připojit. Dále se potvrdilo, že online testování nemá takový smysl, jako testování na základě relace, protože na potvrzenou naplánovanou schůzku se tester dostaví, ale online něco otestovat, když ho netlačí čas, může zapomenout nebo z důvodu pracovní vytíženosti odložit.

Vždy je nutná koordinace zkušeného testera/test analytika či test manažera, který je schopen řešit problémy a pomáhat testerům jejich práci realizovat. Je dobré při testování naslouchat a dělat si poznámky a přijímat veškerou zpětnou vazbu. Dále je poté vhodné s výstupy testování seznámit celý tým, který se testování účastnil a stejně tak je potřeba nezapomínat na předání zpětné vazby, jak se dále pracuje s nálezy. Např. které nálezy byly skutečně defekty a zda byly odstraněné, které nálezy byly dodatečně zařazené do vývoje ve formě tzv. change requestů apod.

Pro tento typ Exploratory testování se osvědčila realizace formou workshopu, kdy v jasně stanoveném čase běžní uživatelé – testeři společně testují a mají k dispozici podporu ze strany zkušeného člena odborného testovacího týmu.

Závěry z testování

Bylo potvrzeno, že pro určitý typ aplikace a to aplikace, které využívají běžní uživatelé – klienti, dává smysl využít metodu Exploratory testování, a to formou testování v předem stanoveném kratším časovém úseku. Je vhodné zařadit tento typ testování do co nejranější fáze dodávky funkcionality (případně také při testování prototypu), protože mohou být odhaleny některé logické procesní chyby nebo problémy s navrženým UX z pohledu uživatelského komfortu.

4.5.2.2 Oblast 2: Rychlé ověření dodané úpravy či funkcionality

Pro tuto oblast byl vytipován systém DWH neboli datový sklad, kde se již podařilo standardizovaně nastavit předávky nových dodávek k testování. Protože se jedná o specializovaný a zároveň klíčový IT systém společnosti, je k dispozici odborný Business Intelligence analytický tým, který se podílí na definici zadání pro vývoj a následně také na testování. Tím pádem má tento analytický tým jasnou představu o tom, jak má dodávka nového vývoje vypadat. Dodávky jsou poměrně rozsáhlé, nicméně způsob testování jednotlivých částí se do jisté míry opakuje.

Testování dodávek DWH probíhá v těchto kolech:

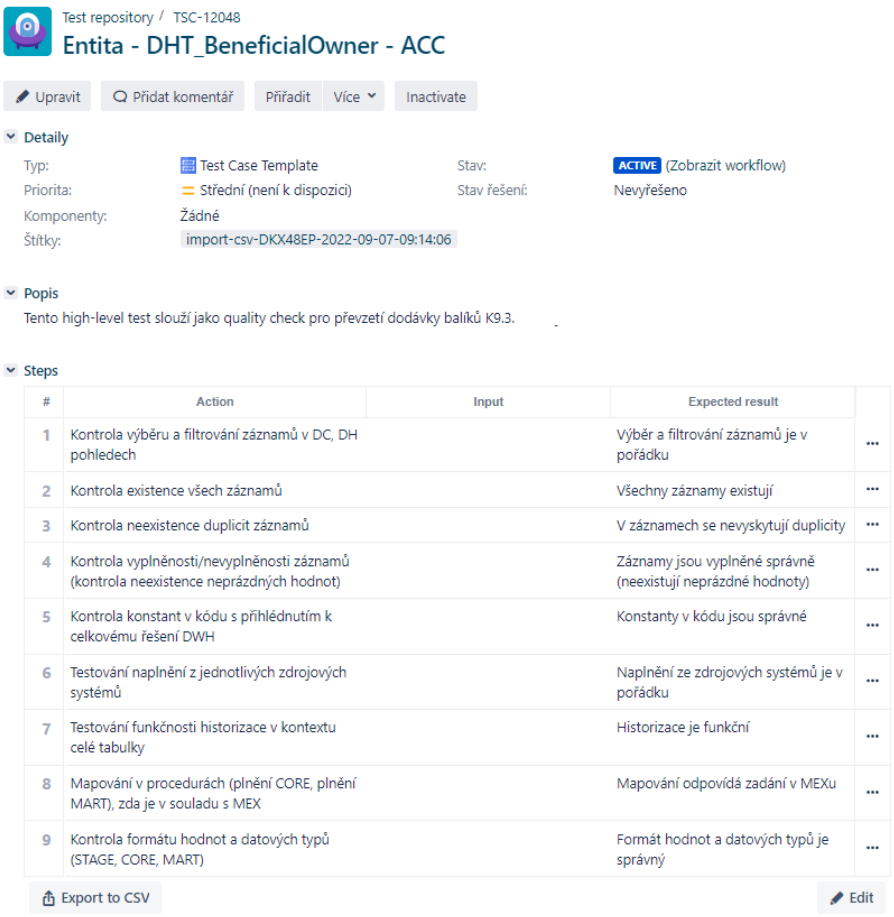
- 0.kolo – FAT na vývojovém prostředí + SMOKE testy na INT prostředí, které realizuje dodavatel. Zjištěné defekty jsou dodavatelem neprodleně odstraněné a takto je dodávka předána k testům, které zajišťuje zadavatele (zvolená společnost).
- **1. kolo – Quality Check – Exploratory testování na INT prostředí – testy realizují BI analytici, tj. Business Intelligence analytici pro oblast dat. Podrobněji je toto testování popsáno níže.**
- 2. kolo – Systémové testy – Scénářové testování na INT prostředí – testy realizují odborní testeři.
- 3. kolo – Akceptační testy – Scénářové testování na UAT prostředí – testy realizují business testeři, tzv. Business Data Stevards – vybraní kolegové mimo IT s příslušnou znalostí práce s daty.

Metoda Exploratory testování byla vytipována jako vhodná metoda pro realizaci 1. kola testování - tzv. Quality Check fáze, a to z toho důvodu, že než dojde k zahájení komplexního podrobného testování implementací jednotlivých atributů, tj. datových položek v tabulkách datového skladu, je nutné ověřit funkčnost dodávky jako celku pohledem nad jednotlivými entitami (tabulkami). Toto velmi často probíhá opakovaně nad konkrétní entitou, protože jsou postupně rozšiřovány o další položky. Z druhé strany, pro různé entity je vhodné provádět různé testy dle uvážení odborných BI analytiků. Podrobná příprava testů pro toto testování by byla časově velmi náročná a musela by být realizována přímo BI analytiky, kde není k těmto účelům disponibilní kapacita. Navíc je zde ale potřeba určitým způsobem standardizovat přístup k testům prováděným v rámci 1. kola – Quality

Check testování. Cílem testování BI analytika je zhodnotit, zda je dodávka úplná a nenarušuje funkčnost jednotlivých entit.

Před zahájením testovaní metodou Exploratory testování byl v tomto případě zrealizován workshop se všemi BI analytiky a dalšími odbornými kolegy zabývajícími se vývojem datového skladu. Na workshopu účastníci navzájem sdíleli své postupy při testování a hledali stručné vodítko neboli definici cílů testů pro testování 1. kola – Quality Check. Na základě toho byly vytvořené kroky v podobě chart, tzn. toho, co je potřeba téměř vždy otestovat a čím se následně budou řídit všichni BI analytici při realizaci testování. Testování bude tedy probíhat na základě Hybridní techniky – Průzkumné testování založené na scénáři, kde scénáře představují strukturu a Exploratory testování přidává variace.

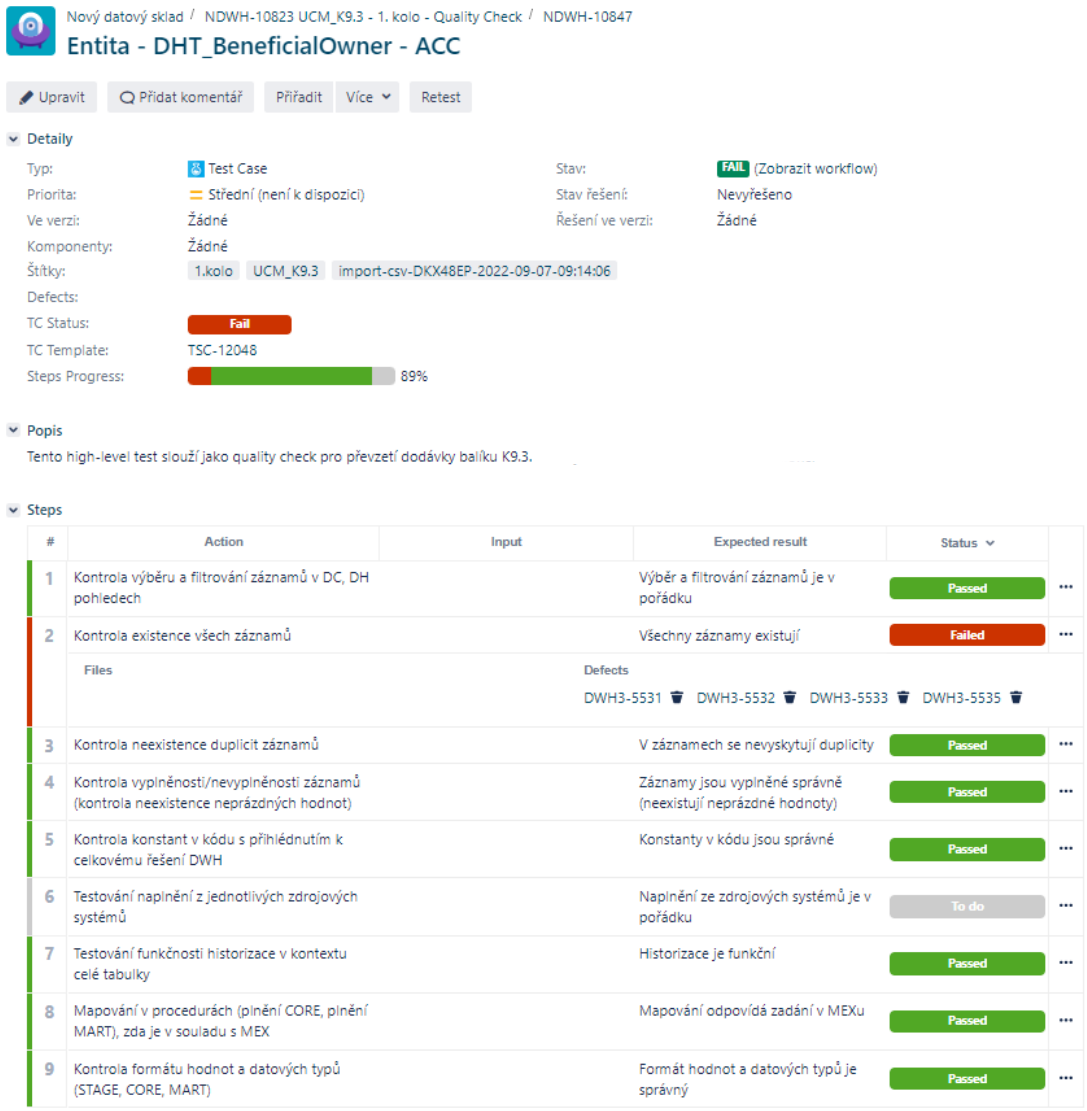
Tým se shodl na následujících krocích a pro každou entitu tak mají vystaven záznam, tzv. ticket v systému Jira, který je vede po jednotlivých oblastech, kterými je vhodné se zabývat při testování na úrovni entit (tabulek). Dle charakteru jednotlivých entit mohou být některé oblasti vynechány, protože nedává smysl takovou oblast nad danou entitou testovat. Níže je ukázka takového záznamu (ticketu) v Jira pro konkrétní entitu.



Obrázek 14 - Výstup ze systému JIRA – charty (Autor)

Jak je vidět ze záznamu (ticketu) z Jira, je k evidenci využit modul pro standardní scénářové testování, nicméně jednotlivé kroky jsou pouze vodítkem pro specifikovanou oblast, která je testována. Jednotlivé kroky zde představují charty, jejichž úspěšné otestování zaznamená tester standartním způsobem v Jira.

Výstupem z testování je tedy vyplněný ticket, který obsahuje jednotlivé charty, nad kterými bylo potřeba udělat mnoho kroků, které vhodně volí dle charakteru entity zkušení BI analytici.



Obrázek 15 - Výstup ze systému Jira – hotový test (Autor)

Identifikované problémy, doporučení a příležitosti ke zlepšení v oblasti 2: Rychlé ověření dodané úpravy či funkcionality zkušeným testerem, obvykle z řad procesních či business analytiků, příp. seniorních testerů, dle standardizovaného postupu.

Při diskusi s jednotlivými testery BI analytiky vyplynula potřeba pravidelně opakovat workshop, na kterém vzájemně sdílí své postupy při testování 1. kola a revidují jednotlivé kroky – charty. Doporučená frekvence je minimálně jednou za půl roku a iniciativa by měla vycházet od test managementu.

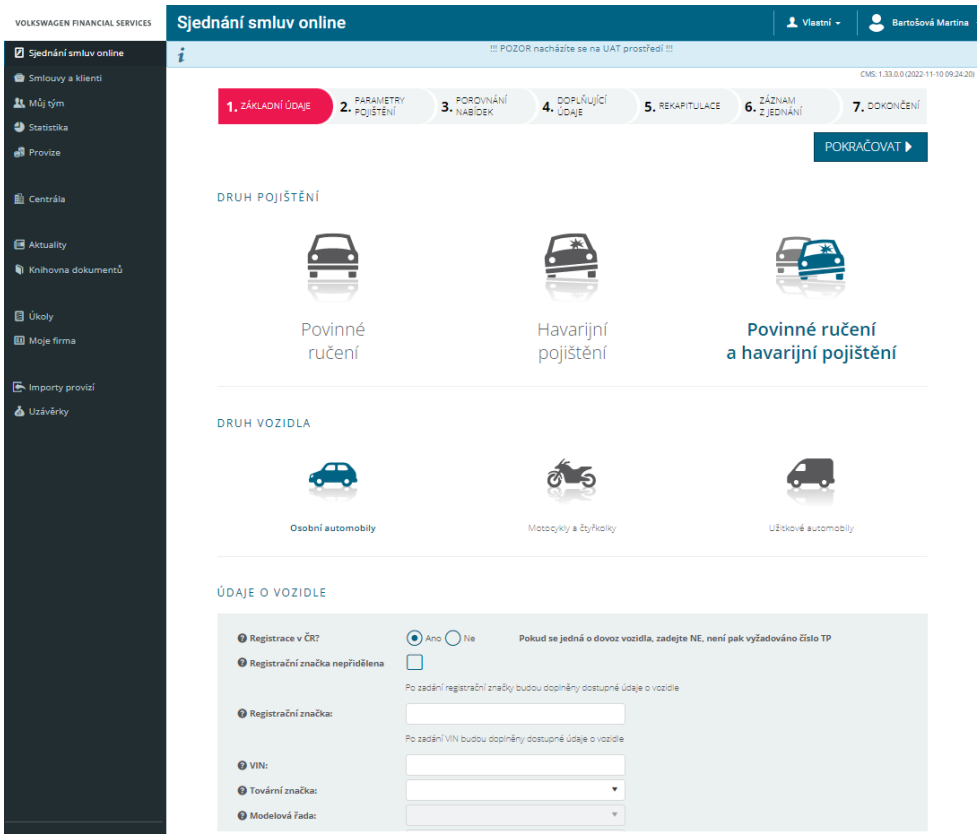
Závěry z testování

Takto upraveným a řízeným procesem testování 1.kola – Quality Check v případě dodávek DWH se podařilo standardizovat postup a rozsah testů různými testery – BI analytiky, aniž by bylo nutné zpracovávat velmi podrobné testovací scénáře pro jednotlivé testy. Použití metody Exploratory testování v tomto případě využívá odbornou znalost testované problematiky testery – BI analytiky a standardizuje jejich postupy při zachování efektivního využití jejich času (kapacit). Tento postup je ale možné aplikovat pouze ve správně vybraných fázích (kolech) testování a je nutné přihlédnout k problematice, která je testována, a odbornosti osob, které testování provádějí. Například podrobné testování jednotlivých atributů není vhodné touto metodou realizovat.

4.5.2.3 Oblast 3: Podpora automatizovaných regresních testů aplikace

Pro poslední oblast, kterou byla zvolena podpora automatizovaných regresních testů, byla vytipována webová aplikace pro náběr pojištění (NIA). Tato aplikace slouží k zajištění pojistných smluv bez financování. To znamená, že pokud klient kupuje auto u dealera (nové, ojeté, po leasingu apod.) a nechce ho financovat přes společnost Volkswagen Financial Services, je mu alespoň v rámci spolupráce nabídnuto pojištění. Toto pojištění se sjednává přes aplikaci spravovanou touto společností.

Aplikace NIA je stabilní aplikace a potřební změny (obchodní, legislativní) jsou dodávány vývojem s využitím agilního přístupu s pravidelnými dodávkami 1x za 14 dní, kdy na testování dodaných změn zpravidla bývají pouze 2 dny, tudíž se jich účastní kromě odborného testera také business analytici. To znamená opakované testování stejných scénářů. Protože průchod aplikací (testovacího scénáře), tedy sjednání pojištění, trvá pro jeden testovací případ kolem 15-20 minut a odehrává se celkem na 7 záložkách. Je nutné protestovat větší množství kombinací (různé pojišťovny, různé typy pojištění, různé typy klientů a dopravních prostředků), tak z tohoto důvodu bylo rozhodnuto zavést automatizaci regresních testů, která byla nasazena k pravidelnému použití v říjnu 2022.



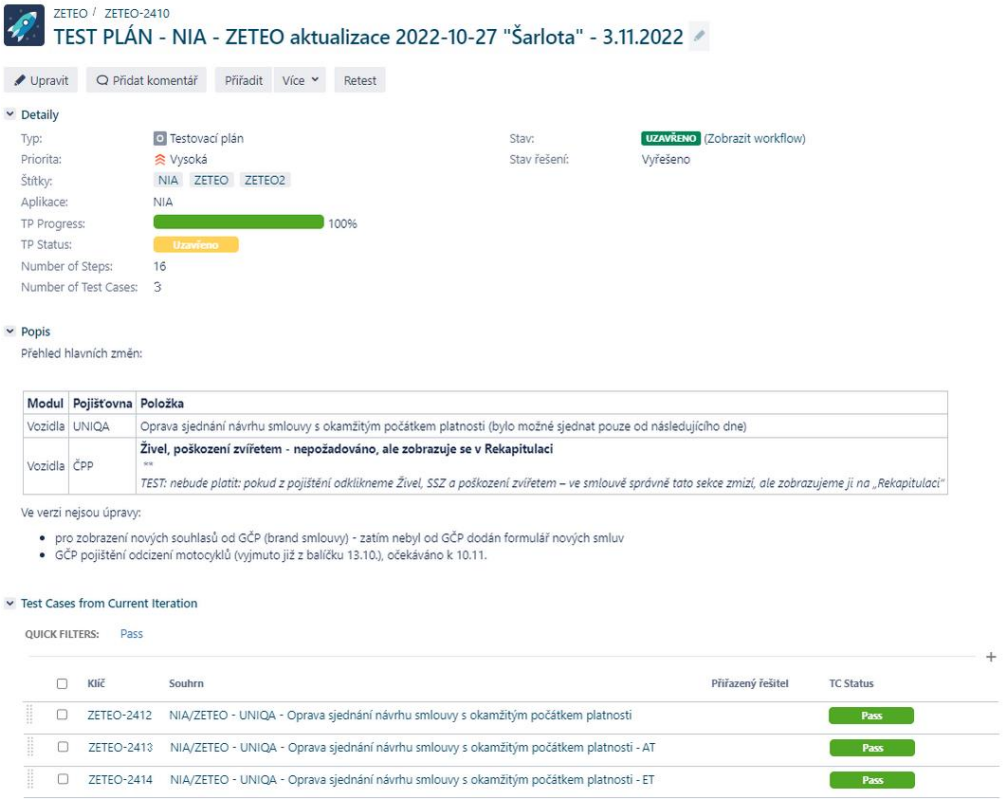
Obrázek 16 - Aplikace NIA (Autor)

V tuto chvíli tedy na této aplikaci probíhá regresní testování kombinovaným způsobem s využitím automatů a Exploratory testování. Proces testování byl stanoven tímto způsobem:

1) Nejdříve se připraví testovací plán zahrnující:
    a. testovací scénáře nově dodaných funkcionalit či úprav
    b. automatizované regresní testy
    c. Exploratory test
2) Dodavatel dodá změny na testovací prostředí aplikace NIA.
3) Provedou se standardní scénářové testy nových funkcionalit či úprav. Nalezené chyby se řeší standardním způsobem.
4) Spustí se regresní automatizované testy. Případné nalezené chyby se řeší standardním způsobem.
5) Vykoná se Exploratory test. Případné nalezené chyby se řeší standardním způsobem.
6) Dodávka se nasazuje na produkční prostředí.

Na následujícím obrázku je sestaven test plán pro dodávku s názvem „Šarlota“ pojmenované dle jména v kalendáři, kdy je předpokládané nasazení na testovací prostředí. Tato dodávka je malého rozsahu a změna spočívá pouze v tom, že lze u pojišťovny UNIQA sjednat pojištění s okamžitou platností, kdy doposud to bylo možné pouze od následujícího dne.

V testovacím plánu v Jira lze vidět 3 testovací případy dle navrženého plánu testovacího procesu. Všechny testy byly provedeny dle plánu, a pro tuto dodávku nebyly nalezeny žádné defekty.



Obrázek 17 - Test plán pro dodávku "Šarlota" (Autor)

Identifikované problémy, doporučení a příležitosti ke zlepšení v oblast 3: Podpora automatizovaných regresních testů aplikace.

Bohužel zde zatím kvůli vytížení kapacit nezbývá tolik času na vytvoření a procházení chart pro Exploratory testy, takže se zatím testuje na základě zkušeností odborných testerů. Doporučením tedy je vypracování chart pro Exploratory testy, aby mohlo být Exploratory testování delegováno i na další kolegy z testerského týmu. V praxi se také kvůli časovému omezení osvědčuje změněný postup testování, kdy se po nasazení změn otestuje aplikace pomocí Exploratory testování, poté se projdou scénáře, a nakonec se spustí automatické regresní testy. Doporučením v této oblasti testování je pravidelně sdílet nálezy a zkušenosti s testováním ve skupině testerů a do budoucna připravit charty mapující komplexně celou aplikace.

Závěry z testování

Přínosem zavedení tohoto postupu jsou kapacitní a časové úspory a dále pak standardizace v rozsahu regresních testů. Odbornému manuálnímu testerovi zbývá čas na detailnější průzkum aplikace, zejména se může zaměřit na UX grafické prvky aplikace a záložky, kde je obtížné a časově náročné vytvořit automatizovaný test.

4.5.3 Vypracování metodického postupu využití Exploratory testování

Z provedeného Exploratory testování a vyzkoušení několika přístupů k němu bylo zjištěno, že se toto testování jeví jako efektivní, ať už z časového a kapacitního hlediska, ale také z důvodu rychlé zpětné vazby, a je rozhodně vhodné začlenit jej ho do postupů specifikovaných ve směrnici test managementu. Bylo však prokázáno, že pro toto testování není možné sjednotit metodiku zahrnující jednotný přístup ke všem testovaným aplikacím. Proto bude upravena a následně zahrnuta předpokládaná metodika Exploratory testování do testování zvolené společnosti, kde pro každé plánované testování, které je něčím specifické, bude vytvořen konkrétní přístup.

4.5.3.1 Použití

Aplikace, u kterých bude použito Exploratory testování zobrazuje následující přehled aplikací a typů testování, který bude sloužit jako rozhodovací tabulka při vytváření testovacího plánu určité aplikace.

	Scénářové testování	Exploratory testování	Automatizované testy
Klíčové aplikace a systémy	✓	✓	✓
Aplikace pro finanční operace a účetnictví	✓	X	✓
Aplikace zajišťující požadavky legislativy	✓	X	✓
Klientské aplikace	✓	✓	✓
Interní systémy	✓	X	X
Podpůrné aplikace	✓	X	X

Tabulka 6 - Přehled testování pro oblasti aplikací (Autor)

4.5.3.2 Proces

Aplikace, která byla určena jako vhodná pro Exploratory testování bude mít při vytváření plánu testování jasně dáno, jak bude Exploratory testování provedeno. Ve zvoleném testovacím nástroji bude připraven scénář v podobě charty o min. jednom kroku, ke kterému budou vázány zjištěné defekty.

Při tvorbě plánu bude rozhodnuto, kdo se Exploratory testování zúčastní, zda se bude jednat o skupinové testování nebo testování jednotlivce a zda budou přizvány nezúčastněné

strany nebo se testování bude dotýkat těch, kdo s aplikací denně pracují. Toto bude rozhodnuto dle účelu a složitosti aplikace. Na výběr bude:

1) Tester, který má na starosti scénářové testování zvolené aplikace.
2) Business analytik, který aplikaci velmi dobře rozumí a zajišťuje také zadání pro vývoj a podílí se na jejím testování.
3) Nezúčastněné strany (nezúčastněné je zde myšleno jako nezapojené do vývoje):
    a. Zástupci uživatelů
    b. Testerský tým

4.5.3.3 Nástroje

Pro zvolení vhodného testovacího nástroje byl v kapitole 3.3.10 proveden průzkum trhu a favority se, díky svým možnostem, které nabízejí, staly aplikace Xray Exploratory App a Zephyr (obě jsou doplňky pro systém Jira). S testovacím týmem byl proto uspořádán workshop, na kterém odborní testeři sdíleli své zkušenosti a přání ohledně testovacího nástroje a bylo jednohlasně rozhodnuto, že stávající nástroj Jira TestFLO naprosto vyhovuje a je v mnoha ohledech lepší než navrhované nástroje. Ačkoliv není přímým nástrojem pro podporu Exploratory testování, tak lze těmto potřebám přizpůsobit. Jako další podpora testování se ukázal dobrým nástrojem Microsoft Forms, který je součástí Office 365, jenž je používán ve firmě.

4.5.3.4 Výsledky testování

Co se týče zaznamenání a správy výsledků testování, nelze taktéž dopředu formulovat konkrétní podobu, protože se ukázalo, že někde je vhodné zaznamenat výsledky do připraveného formuláře, kde se po analýze ukáže, zda jsou zjištěné výsledky defekty (bugy) nebo pouze změnové požadavky a dle toho se zadají do Jira – TestFLO. Pokud však aplikaci testuje business analytik nebo zkušený odborný tester, ten dobře ví, zda se jedná o defekt nebo požadavek na změnu a může tedy rovnou provést příslušný záznam v rámci Jira. Pokud se však testování budou účastnit nezúčastněné strany, jako jsou přizvaní business testeři či zástupci běžných uživatelů, jeví se jako vhodný prostředek formulář a dále také vedoucí testování či jiný člen testovacího týmu, který zaznamenává zjištěné nesrovnalosti a další náměty ke zlepšení. Těmto stranám je důležité sdělit závěry z testování a sdílet s nimi, jakým směrem se vývoj bude ubírat.

5 Výsledky a diskuse

Na základě předpokládané podoby metodiky bylo provedeno ověření, zda je Exploratory testování vhodné pro podnikovou praxi. Byla vytvořena rozhodovací tabulka, která pomáhá identifikovat, zda se zvolená aplikace hodí pro Exploratory testování či nikoliv, a na jejímž základě byly položeny výzkumné otázky. Dle těchto otázek byly vytipovány tři oblasti testování, kde každou představuje jiná aplikace.

5.1 Oblast 1

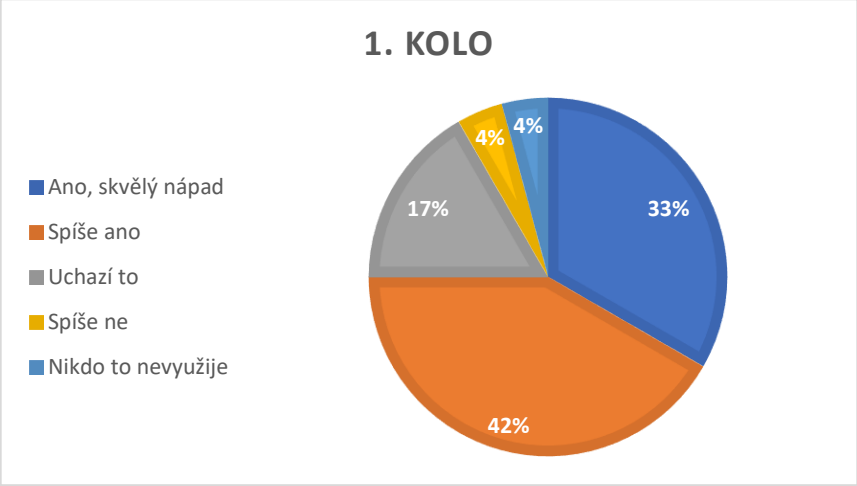
První oblast představuje klientskou aplikaci Zákaznický portál, která byla zvolena vhodným kandidátem na přístup Exploratory testování. Tato oblast měla za úkol ověřit, zda je Exploratory testování vhodným doplňkem ke scénářovému testování a zda je vhodné testovat ranou fázi dodávky aplikace z pohledu klienta bez předchozích zkušeností s aplikací a odpovědět na výzkumnou otázku. Testování bylo rozděleno na dvě samostatné části, dodávku pro úpravy změn kontaktních údajů klientů a testování chystaného chatbota.

První část testování byla provedena porovnáním scénářového a Exploratory testování, jenž bylo realizováno pomocí dvou skupin, kde každá představovala jednu stranu. Skupina A testovala dle přesně daných kroků testovacích scénářů a bez časového omezení. Skupina B neměla žádný scénář, byla pouze obeznámena s informacemi, co dodávka nové funkcionality přináší a pro testování dostala vyhrazený časový limit 1,5 hodiny. Toto testování bylo provedeno dle techniky Session Based Test Management, jejímž úkolem je testovat ve zvoleném časovém úseku bez přerušování. Výsledkem testování bylo nalezení celé řady defektů a změnových požadavků od obou skupin. Zjištěním však bylo, že Exploratory testování nalezlo mnohem více defektů, které scénáře neobjevily. Jedním z těch významných byl očekávaný proklik loga na jednotlivých stránkách, který byl nefunkční, tento krok se ve scénáři vůbec nevyskytoval, ale „běžné uživatele“ to při průzkumu webu okamžitě napadlo vyzkoušet. Po uběhnutí časového limitu byly nad provedeným testováním sdíleny výsledky a vedena diskuse o nalezených chybách. Důležité je při tomto testování také seznámit skupinu testerů, zastupující uživatele, s výsledky testování, což bylo také provedeno na nadcházející společné schůzce. Po tomto testování je možné konstatovat, že Exploratory testování se osvědčuje jako vhodný doplněk testování. Scénářové testování na rozdíl od Exploratory testování otestuje přesně vymezené požadavky na textaci a očekávané chování nové funkcionality. Exploratory testování toto testování však dotáhne téměř

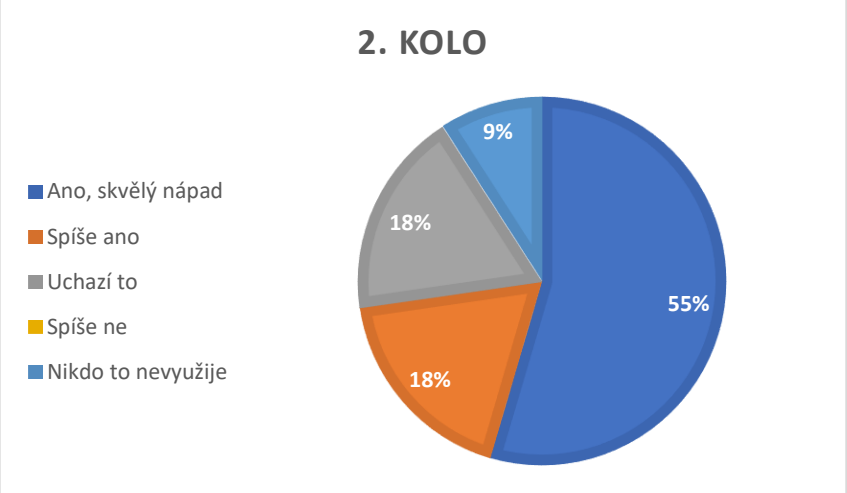
k dokonalosti tím, že se věnuje i dalším oblastem, jako je například záhlaví a zápatí stránky, které přímo vyzývá k vyzkoušení. Dále také otestuje, zda je stránka a samotná textace vůbec srozumitelná někomu, kdo aplikaci nezná, protože pro klientskou aplikaci je zásadní, aby byla jasná a srozumitelná pro široké spektrum uživatelů.

Druhá část testování klientské aplikace Zákaznický portál byla naprosto jiného charakteru. Pro klienty, kteří již mají úvěr nebo operativní leasing u zvolené společnosti, ale stále nejsou registrovaní v portálu, byla vytvořena pilotní verze chatbota, který má za úkol s touto registrací pomoci. Jeho klíčovou funkcí má být odpovídat na otázky proč a jak se do portálu registrovat. Pro toto testování bylo shledáno scénářové testování jako nevyhovující, takže bylo rozhodnuto o oslovení zaměstnanců, zda mají zájem o testování chatbota metodou Exploratory testování, opět na základě Session Based Test Management. Zájemců se přihlásilo 24, z čehož 13 tvořilo skupinu zastupující uživatele a 11 bylo z řad zkušených testerů. Při testování, které mělo nastaven časový limit 30 minut, bylo zjištěno množství nedostatků, které byly zaznamenávány do rozeslaného elektronického dotazníku anebo byly diskutovány přímo na sezení. Bylo zde zajímavé pozorovat různorodé názory těchto dvou skupin, kde se ukázalo, že zkušení testeři mají na tuto aplikaci kritičtější pohled a testovali věci, které zástupce uživatelů nenapadli. Po tomto testování, které přineslo vlastníkovi produktu úplně nový pohled na aplikaci, bylo nutné změnit a doplnit logiku aplikace a bylo zjištěno, že aplikace není v žádném případě připravena na vydání do produkce. Pro úspěch tohoto testování bylo po provedených opravách a doplnění logiky aplikace uspořádáno i druhé kolo, kde byla vyzkoušena tentokrát forma online. Byli osloveni testeři z předchozího kola a další zaměstnanci, s tím, že jim byl s pozvánkou rovnou rozeslán dotazník s téměř totožnými otázkami. Tato online forma se neukázala tolik efektivní z hlediska účastnosti, jako předem domluvená a naplánovaná schůzka. Druhého kola se zúčastnilo tentokrát pouze 11 zaměstnanců, respektive bylo vyplněno 11 dotazníků, otestování mohlo být provedeno více. Pro porovnání dvou kol jsou zde zobrazené odpovědi ve formě grafů.

První otázka „Dává ti aplikace smysl?“ byla z velké části v obou kolech zodpovězena kladně, jde však vidět rozdíl, kdy se zdá aplikace skvělým nápadem o 22 % více ve druhém kole testování, po provedení razantních změn.

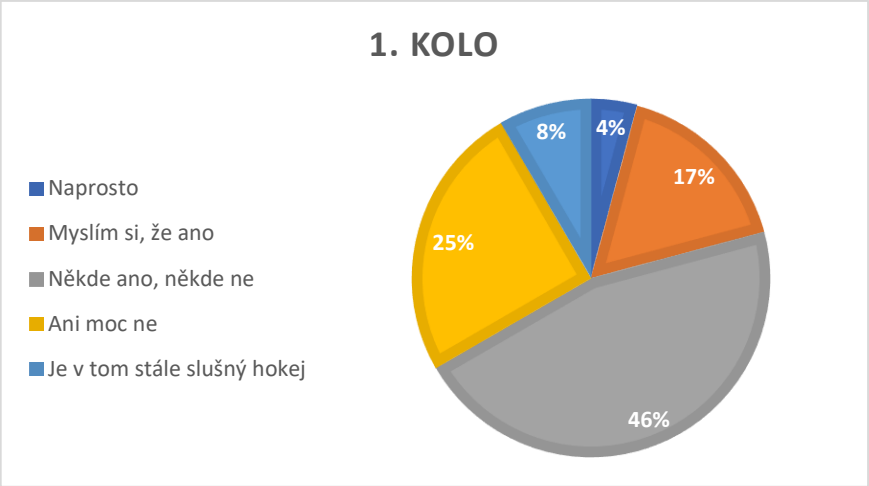


Graf 6 - 1. otázka 1.kola testování chatbota v % (Autor)

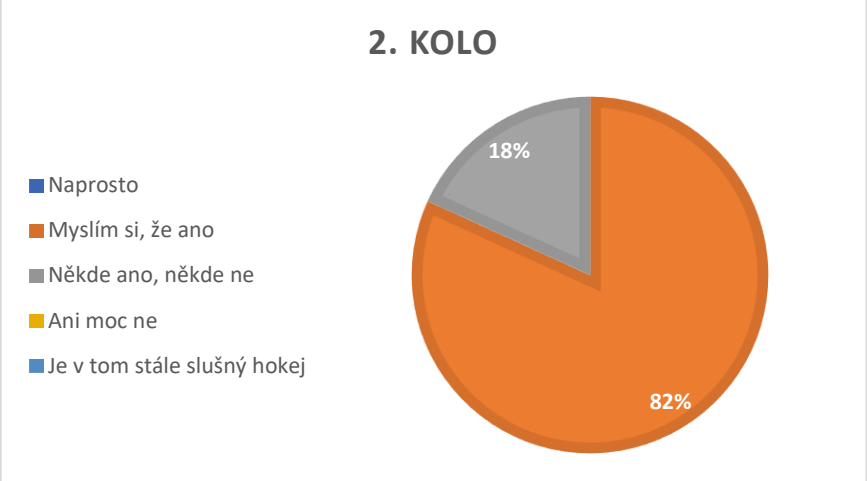


Graf 7 - 1. otázka 2.kola testování chatbota v % (Autor)

Druhá otázka „Je provázanost odpovědí a rozcestníků logická?“ byla v prvním kole hodnocena ze 46 % váhavě, někde smysl dává, někde ne. Ve druhém kole bylo však z 82 % odpovězeno, že smysl spíše dává, váhání projevilo pouze 18 % testerů.



Graf 8 - 2. otázka 1.kola testování chatbota v % (Autor)



Graf 9 - 2. otázka 2.kola testování chatbota v % (Autor)

Třetí otázka „Co se ti na Sáře líbí nejvíce?“ byla položena pouze v prvním kole. Odpovědi jsou shrnuté ve druhé části kapitoly 4.5.2.1.

Poslední otázka, týkající se slabin a nesrovnalostí při chatování, byla položena v obou kolech a uživatelé se zde skutečně rozepsali. V prvním kole bylo nejdůležitějším zjištěním, že chatbot nereagoval na očekávaná klíčová slova v kontextu smluv nebo odpovídal na jiné otázky, než byly ty položené. Ve druhém kole byly odpovědi na tuto otázku i kladné, bylo zde viděno velké zlepšení a bylo již možné vést „normální“ konverzaci. Jedinou slabinou, na které se testeři shodli, bylo neustálé rolování textu, protože chatbot občas napíše více zpráv za sebou. Po druhém kole bylo provedeno ještě pár změn a chatbot je nyní ve fázi přípravy na nasazení do produkce, která je plánována na prosinec 2022. Chatbot bude v půlročním pilotním provozu, kde ho tentokrát budou testovat reální klienti, kteří teprve o jeho smyslu a využití skutečně rozhodnou.

Výsledky těchto dvou testování, které představovaly oblast 1, skutečně potvrdily, že je Exploratory testování vhodné pro klientské aplikace. Bylo také potvrzeno, že je do tohoto testování vhodné zapojit lidi bez zkušeností s aplikací, kteří vhodně reprezentují skupinu klientů, jenž budou s aplikací/novou funkcionalitou také pracovat v budoucnu poprvé a nebude jim známa žádná dokumentace ani požadavky, se kterými jsou většinou obeznámeni manuální testeři, kteří testují pomocí scénářů.

5.2 Oblast 2

Druhou oblast představuje klíčový systém, reprezentovaný datovým skladem (DWH). Úkolem této oblasti bylo ověřit, zda je Exploratory testování vhodné k rychlému ověření dodávky, které provádí zkušený tester, jenž aplikaci velmi dobře zná. Testování datového skladu probíhá v několika kolech, kdy pro Exploratory testování, bylo vybráno první kolo zvané „Quality Check“. Toto testování je prováděno na integračním prostředí samotnými Business Intelligence analytiky. Pro oblast 2 bylo Exploratory vybráno kvůli složité a časově náročné přípravě testovacích scénářů nad mnoha entitami, kdy každá vyžaduje různé testy. Pro datový sklad byla zvolena hybridní technika Exploratory testování, konkrétně Exploratory testování na základě scénáře. Po samotné myšlence přišel nápad zrealizovat workshop, jehož součástí byly zúčastněné strany, které si předaly své postupy při testování prvního kola dodávky DWH. Na základě zkušeností všech BI analytiků a dalších kolegů, byl připraven scénář, který obsahoval kroky v podobě chart. Jedná se o tzv. návod na to, kterou oblast je potřeba téměř vždy otestovat, ale ne však konkrétně jak, jde zde pouze o strukturovanou cestu. Toto testování bylo ověřeno a pro svůj úspěch bylo všemi BI analytiky standardizováno. Odpověď na tuto otázku je tedy ano, Exploratory testování je vhodné, a oproti scénářům znatelně méně časově náročné, pro rychlou dodávku, která bude ověřena někým zkušeným, kdo aplikaci skutečně rozumí. Určitě je však doporučeno takto univerzální scénář, složený z kroků představujících charty, pravidelně projít a aktualizovat.

5.3 Oblast 3

Třetí oblast ověřila klíčová webová aplikace zajišťující náběr pojistných smluv vozidel, které nejsou financované ve zvolené společnosti. Cílem zde bylo ověřit, zda je Exploratory testování vhodné jako doplněk neboli podpora automatizovaných regresních testů. Tato aplikace byla vytipována z toho důvodu, že je každé dva týdny doručována

dodávka nových funkcionalit nebo změn a na psaní testovacích scénářů a samotné testování jsou určeny pouze dva dny. Z kapacitních a časových důvodů zde byly v rámci této změny vytvořeny automatizované regresní testy, které zde spolu se scénářovým testováním nově dodaných funkcionalit a změn testují aplikaci. Exploratory testování, jako podpora, bylo zvoleno tentokrát ve „volném stylu“, kdy tester, po dodání nových funkcionalit projde aplikaci a očekávané změny prozkoumá bez scénáře, zaměří se i na grafické prvky a poté projde scénář. V praxi byl vyzkoušen i přístup, kdy tester zodpovědný za aplikaci, testuje dle scénářů a BA prozkoumává aplikaci pomocí Exploratory testování. Možné jsou oba přístupy, avšak do budoucna je doporučeno vytvoření scénáře obsahujícího charty, jako tomu bylo při testování DWH v kapitole 4.5.2.2, aby testování získalo určitý řád. Po tomto testování je však možné konstatovat, že Exploratory testování je vhodná podpora k automatizovaným regresním testům, avšak, pokud jsou zde i scénářové testy, nejedná se o nepostradatelný doplněk. V rámci dalšího ověřování efektivity by bylo možné provést ověření, zda by bylo možné scénáře nových funkcionalit a změn zcela nahradit Exploratory testováním, které provádí tester, jenž je dobře seznámený s touto aplikací.

6 Závěr

Tématem práce bylo ověření metody Exploratory testování v podnikové praxi. Hlavním cílem bylo vytvoření metodiky, která popisuje, jakým způsobem testovat s využitím metody Exploratory testování, identifikace vhodných aplikací a systémů a ověření této metodiky v praxi. Dílčí cíl práce byla analýza a porovnání používaných metod ve zvolené společnosti. Nejdříve byla zpracována teoretická část práce, která byla rozdělena na tři celky, a to vývoj softwaru, testování a Exploratory testování. Při zpracování literární rešerše bylo získáno mnoho nových poznatků potřebných pro realizaci cílů.

Praktická část byla rozdělena na několik částí dle cílů práce. Nejdříve byl v kapitolách 4.2 a 4.3 realizován cíl analýzy používaných metod testování ve zvolené společnosti. V této části práce byla popsána samotná společnost a její IT struktura včetně rozdělení používaných aplikací a systémů. Byl zde popsán vývoj softwaru v dané společnosti a posléze samotné testování, kde byly zmíněny role, používané nástroje, testovací prostředí a jeho porovnání se standardy.

Další část se zaobírala samotnou metodikou Exploratory testování. Na základě získaných poznatků teoretické práce bylo zjištěno, že Exploratory testování nabízí mnoho podob, a proto byla v kapitole 4.4 připravena pouze předpokládaná podoba této metodiky. K zavedení Exploratory testování bylo potřeba identifikovat vhodné systémy a aplikace, což bylo provedeno v kapitole 4.5 vytvořením rozhodovací tabulky, pomocí které byly vybrány konkrétní aplikace. Po identifikaci vhodných aplikací byl částečně realizován cíl a bylo potřeba tyto aplikace otestovat v praxi. V kapitole 4.5.1 byly položeny výzkumné otázky a testování bylo rozděleno na tři oblasti, které otestují klientské aplikace a klíčový systém společnosti. První rozsáhlá oblast byla rozdělena na dva celky a bylo zde pomocí porovnání scénářového a Exploratory testování zjištěno, že Exploratory testování je vhodným doplňkem ke scénářovému testování. První oblast dále testovala, zda může tento typ testování provádět nezkušený uživatel, který nemá s daným softwarem žádnou zkušenost a nepodílel se na jeho vývoji. Závěrem z obou celků je, že ano, toto testování je vhodné pro skupinu nezkušených testerů – uživatelů, kteří zastupují budoucí zákazníky či uživatele. Druhá oblast testování se zaobírala rychlým ověřením dodávky za pomoci Exploratory testování, které provede zkušený tester, jenž je s aplikací dobře obeznámen a podílí se na jejím vývoji. Tato otázka byla zodpovězena také kladně, pro tuto oblast testování, je Exploratory vhodné. Třetí oblast potvrdila otázku, zda je Exploratory testování vhodné jako

doplněk k automatizovaným regresním testům. Zde bylo zjištěno, že není tak efektivní, jako v předchozích oblastech, ale určitě je užitečné z pohledu testování například UX prvků. Celým tímto testováním byla zodpovězena hlavní otázka a to tak, že pokud je Exploratory testování správně uchopeno, tak může být v podnikové praxi považováno za efektivní.

Cílem této práce bylo vytvořit metodiku, jejíž předpokládaná podoba byla v kapitole práce dle výsledků testování upravena a bylo konstatováno, že nelze vytvořit jednotný přístup k tomuto testování z důvodu jedinečného testování každé aplikace. Proto byla v kapitole 4.5.3 vytvořena metodika obecná, která popisuje, jakým způsobem testovat s využitím metody Exploratory testování. Výhody takového postupu jsou blíže uvedeny v kapitole 3.3.11.

7 Citovaná literatura

Agarwal, B. B., Tayal, S. P. a Gupta, Mahesh. 2010. *Software Engineering and Testing.* místo neznámé : Jones & Bartlett Learning, 2010. 1934015555, 9781934015551.
Azure. Azure Test Plans. *Azure Microsoft.* [Online] https://www.guru99.com/exploratory-testing-tools.html.
Banks, Frances. 2015. What is System Development Life Cycle? *Airbrake.* [Online] 8. 1. 2015. https://blog.airbrake.io/blog/sdlc/what-is-system-development-life-cycle.
Black, Rex. 2014. *Advanced Software Testing Vol. 2.* Santa Barbara : Eock Noot, 2014. 978-1-937538-50-7.
Brookshear, Glenn J. 2015. *Informatika.* Brno : Albatros Media a.s., 2015. 978-80-251-3805-2.
Bureš, Miroslav, a další. 2016. *Efektivní testování softwaru.* Praha : Grada Publishing, a.s, 2016. 978-80-271-9389-9.
Čermák, Miroslav. 2008. Black box test. *Clever and Smart.* [Online] 25. 12. 2008. https://www.cleverandsmart.cz/black-box-test/.
—. 2008. Grey box test. *Clever and Smart.* [Online] 25. 12. 2008. https://www.cleverandsmart.cz/grey-box-test/.
—. 2008. White box test. *Clever and Smart.* [Online] 23. 12. 2008. https://www.cleverandsmart.cz/white-box-test/.
Danić, Filip. 2021. A Guide to Exploratory Testing. *Danić Filip.* [Online] 14. 8. 2021. https://danicfilip.com/2020/guide-to-exploratory-testing/.
Exploratory testing: A multiple case study. **Itkonen, Juha a Rautiainen, Kristian. 2005.** Helsinki : Helsinki University of Technology, 2005.
Exploratory Testing: An Overview. **Rashmi, N. a Suma, V. 2015.** Bangalore : International Journal of Computer Applications, 2015.
Hájek, Petr. 2010. Deset největších softwarových chyb v historii lidstva. *Root.* [Online] 29. 9. 2010. https://www.root.cz/clanky/deset-nejvetsich-softwarovych-chyb-v-historii-lidstva/.
Hamilton, Thomas. 2022. 10 BEST Exploratory Testing Tools (Nov 2022). *Guru99.* [Online] 5. 11. 2022. https://www.guru99.com/exploratory-testing-tools.html.
—. 2022. https://www.guru99.com/how-precaution-becomes-cure-risk-analysis-and-solutions-in-test-management.html. *Guru99.* [Online] 22. 10. 2022. https://www.guru99.com/how-precaution-becomes-cure-risk-analysis-and-solutions-in-test-management.html.
—. 2022. Project Team: Develop, Building, Manage: Step by Step Process. *Guru99.* [Online] 12. 11. 2022. https://www.guru99.com/how-to-organize-a-test-team.html.
—. 2022. Role & Responsibilities of Test Manager / Test Lead. *Guru99.* [Online] 22. 10. 2022. https://www.guru99.com/introduction-to-test-management-for-curious-manager.html.
—. 2022. Static Testing vs Dynamic Testing: What's the Difference? *Guru99.* [Online] 29. 10. 2022. https://www.guru99.com/static-dynamic-testing.html.
—. 2022. Test Management Process in Software Testing. *Guru99.* [Online] 14. 9. 2022. https://www.guru99.com/test-management-phases-a-complete-guide-for-testing-project.html.
—. 2022. TEST PLAN: What is, How to Create (with Example). *Guru99.* [Online] 17. 9. 2022. https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html.

Hamiton, Thomas. 2022. Software Test Estimation Techniques. *Guru99.* [Online] 30. 9. 2022. https://www.guru99.com/how-precaution-becomes-cure-risk-analysis-and-solutions-in-test-management.html.
Hammad, Madhuri. 2020. Levels of Software Testing. *Geeks for Geeks.* [Online] 25. 1. 2020. https://www.geeksforgeeks.org/levels-of-software-testing/.
Hendrickson, Elisabeth. 2013. *Explore It!* Dallas : The Pragmatic Programmers, 2013. 978-1-937785-02-4.
Hlava, Tomáš. 2011. Test Case - Testovací případ. *Testování softwaru.* [Online] 12. 7. 2011. http://testovanisoftwaru.cz/dokumentace-v-testovani/test-case/.
—. 2011. Testovací strategie - test plán. *Testování softwaru.* [Online] 12. 7. 2011. http://testovanisoftwaru.cz/dokumentace-v-testovani/test-plan/.
—. 2012. Testovací tým. *Testování softwaru.* [Online] 24. 4. 2012. http://testovanisoftwaru.cz/manualni-testovani/testovaci-tym/.
IBM. 2021. Iterativní vývoj. *IBM.* [Online] 1. 3. 2021. https://www.ibm.com/docs/cs/elm/7.0.0?topic=scenarios-iterative-development.
ISTQB. 2022. About Us. *ISTQB.* [Online] 2022. https://www.istqb.org/about-us/who-we-are.
Knotek, Martin. 2014. *Chci uspět v IT.* Brno : Computer Press, 2014. 978-80-251-4162-5.
Kuldeep, Rana. 2021. Scripted Testing – How is it different from Exploratory Testing? *Art of Testing.* [Online] 16. 4. 2021. https://artoftesting.com/scripted-testing.
Martin Šlancar. **Šlancar, Martin. 2017.** Praha : SecurityWorld, 2017, Sv. 2.
Masopust, Jan. 2017. Tvora navigační aplikace pro potřeby leteckého fotogrammetrického snímkování. *Katedra geoinformatiky Univerzity Palackého v Olomouci.* [Online] 2017. https://www.geoinformatics.upol.cz/dprace/bakalarske/masopust17/?page=metody.
Medewar, Sameeksha. 2022. 14 Best Exploratory Testing Tools for Software Testing. *TechGeekBuzz.* [Online] 21. 9. 2022. https://www.techgeekbuzz.com/blog/best-exploratory-testing-tools/.
Nordeen, Alex. 2020. *Learn Software Testing in 24 Hours: Definitive Guide to Learn Software Testing for Beginners.* [online]. Guru99, 2020.
Parmar, Deepak. 2022. Exploratory testing. *Atlassian.* [Online] 2022. https://www.atlassian.com/continuous-delivery/software-testing/exploratory-testing.
Patton, Ron. 2002. *Testování softwaru.* Praha : Compter Press, 2002. 80-7226-636-5.
Prytulenets, Alesya. 2022. A Brief History of Software Testing. *DogQ Blog.* [Online] 9. 2. 2022. https://dogq.io/blog/a-brief-history-of-software-testing/.
Rajkumar, Manikanta . 2022. Best Exploratory Testing Tools in 2022. *Software Testing Material.* [Online] 5. 10. 2022. https://www.softwaretestingmaterial.com/exploratory-testing-tools/.
Saleh, Kassem A. 2009. *Software Engineering.* Fort Lauderdale : J. Ross Publishing, 2009. 978-1-932159-94-3.
Selendic, Djuka. 2021. A Journey in Test Engineering Leadership: Applying Session-Based Test Management. *InfoQ.* [Online] 2. 10. 2021. https://www.infoq.com/articles/session-based-test-management/.
Software.informer. 2022. Session Tester. *software.informer.* [Online] 26. 8. 2022. https://session-tester.software.informer.com/.
STH. 2022. SDLC (Software Development Life Cycle) Phases, Process, Models. *Software Testing Help.* [Online] 25. 10. 2022. https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/.

Štědroň, Bohumír. 2006. *Manažerské řízení a informační technologie.* Praha : Grada, 2006. 978-80-247-2052-4.
Učební osnovy – Certifikovaný tester. **ISTQB. 2018.** místo neznámé : ISTQB, 2018.
Whittaker, James A. 2009. *Exploratory Software Testing.* Boston : Pearson Education, 2009. 978-0-321-63641-6.
XRAY. XRAY Exploratory Testing. *XRAY.* [Online] https://www.getxray.app/exploratory-testing?utm_source=guru99&utm_medium=banner-toptools&utm_campaign=exploratory-testing.

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam grafů