DIPLOMOVÁ PRÁCE

Návrh inovací procesu softwarové implementace v pojišťovně z pohledu testingu

Innovations Proposal for Software Implementation Process in Context of Testing in Insurance Company

STUDIJNÍ PROGRAM

Projektové řízení inovací

STUDIJNÍ OBOR

Process management

VEDOUCÍ PRÁCE

doc. Ing. Marek Jemala, Ph.D.

PAVELKA

DAVID

2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pavelka** Jméno: **David** Osobní číslo: **475141**

Fakulta/ústav: **Masarykův ústav vyšších studií**

Zadávající katedra/ústav: **Institut ekonomických studií**

Studijní program: **Projektové řízení inovací**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Návrh inovací procesu softwarové implementace v pojišťovně z pohledu testingu

Název diplomové práce anglicky:

Innovations Proposal for Software Implementation Process in Context of Testing in Insurance Company

Pokyny pro vypracování:

Hlavním cílem práce je zanalyzovat specifika managementu procesu releasů softwaru v podmínkách IT oddělení pojišťovny, provést dílčí analýzu nástrojů a metod nové implementace procesu a navrhnout vhodné inovace procesu vyplývající z potřeby přizpůsobení podnikových zdrojů nové technologii v testingu, inovačnímu a release managementu. Mezi specifické cíle patří charakteristika stávajícího a inovovaného procesu releasů, jeho částí, návazností, výhod a nevýhod.

Seznam doporučené literatury:

Vaidyanathan, G. (2012) Project Management: Process, Technology and Practice. Pearson; 1st edition, ISBN-10: 0132807181.
Jemala, M. (2014): Technology identification: How to bring technology innovation to life? Saarbrücken: Scholars' Press, ISBN: 978-3-639-71044-1.
Jemala, M. (2010): Manažment technologických systémov. Identifikácia a prípadové štúdie. Bratislava: Ekonóm, ISBN: 978-80-225-3120-7.
Habart, M. C. - Janssen, J. (2018): Big Data for Insurance Companies (Innovation, Entrepreneurship and Management: Big Data, Artificial Intelligence and Data Analysis. Wiley-ISTE; 1st edition, ASIN: B07962N979.
Schilling, M. (2019) ISE Strategic Management of Technological Inovation 6th edition, McGraw Hill, ISBN 1260565793
Jorgensen, Paul C. Software Testing: A Craftsman`s Approach. 4. vyd. USA: Auerbach Publications, 2013, ISBN 1466560681
HAMMER, M., HERSHMAN, L. Rychleji, levněji, lépe. Devět faktorů účinné transformace podnikových procesů. Praha : Management press, 2013. ISBN 978-80-7261-253-6.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Marek Jemala, Ph.D., institut ekonomických studií MÚ

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **05.01.2022** Termín odevzdání diplomové práce: **28.04.2022**

Platnost zadání diplomové práce: _____

doc. Ing. Marek Jemala, Ph.D. podpis vedoucí(ho) práce	Mgr. František Hřebík, Ph.D. podpis vedoucí(ho) ústavu/katedry	prof. PhDr. Vladimíra Dvořáková, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

PAVELKA, David Návrh inovací procesu softwarové implementace v pojišťovně z pohledu testingu. Praha: ČVUT 2022. Diplomová práce. České vysoké učení technické v Praze, Masarykův ústav vyšších studií.

MASARYKŮV ÚSTAV
VYŠŠÍCH STUDIÍ
ČVUT V PRAZE

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně. Dále prohlašuji, že jsem všechny použité zdroje správně a úplně citoval a uvádím je v přiloženém seznamu použité literatury.
Nemám závažný důvod proti zpřístupňování této závěrečné práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Praze dne: 27. 04. 2022 Podpis:

Poděkování

Rád bych poděkoval své rodině a blízkým za jejich podporu, pomoc a trpělivost.

Mé díky patří také Pojišťovně, která mi psaní práce umožnila a všem jejím zaměstnanců, kteří mi s prací pomohli a vždy ochotně odpovídali na mé otázky.

V neposlední řadě bych rád poděkoval vedoucímu práce, panu doc. Ing. Markovi Jemalovi, Ph.D., za odborné vedení a konzultace.

Abstrakt

Práce předkládá návrh inovací procesu softwarové implementace v IT sekci nadnárodní pojišťovny, konkrétně z pohledu problematiky testování softwaru.

Práce obsahuje příslušnou teorii testování softwaru, implementace softwaru a modelování procesů. Na teorii navazuje praktická část, jež se skládá z analýzy současné situace v řešené pojišťovně, popisu aktuálního procesu softwarové implementace a následného návrhu inovací procesu. Navržený proces je doplněn diagramy. Práce je uzavřena doporučeními k implementaci navržených inovací.

Klíčová slova

Testování softwaru, životní cyklus vývoje softwaru, modelování procesů, inovace.

Abstract

The master thesis proposes innovations of software implementation process in the IT section of an insurance company. The process innovation is tackled from the perspective of software testing.

The proposal is based on a relevant theory from the fields of testing, software implementation and process modelling. The theory is followed by an analysis of the current situation in the company and by a description of the current software implementation process, and then is followed by the proposal of new process innovation itself. The process is completed by diagrams. The thesis is concluded by a list of recommendations regarding implementation of the newly proposed innovations.

Key words

Software Testing, Software Development Life-Cycle, Process Modelling, Innovations.

Obsah

1 Úvod

Implementace vyvinutého softwaru je jednou z nejzásadnějších a nejkritičtějších součástí životního cyklu vývoje softwaru. Právě ve fázi implementace dochází k napojení nového systému/nové verze systému do současné struktury a může dojít k řadě chyb, kterým je třeba předejít nastavením správného procesu. A právě tomuto procesu, s důrazem na testing, se věnuje tato diplomová práce.

1.1 Cíl práce

Cílem této diplomové práce je navrhnout inovace procesu softwarové implementace na konkrétním případu IT sekce nadnárodní pojišťovny. Navržené inovace procesu budou doplněny o doporučení k zavedení do provozu.

Jako první krok, pro získání kontextu, je třeba provést stručnou charakteristiku řešené společnosti (dále už jen „Pojišťovna"), uvést, jak v jejím rámci funguje IT sekce, a zvláště pak oddělení testingu.

Nezbytnou součástí práce je rešerše vybraných relevantních zdrojů. Teoretická část práce se zaměřuje na problematiku testování, vývoje softwaru, implementace softwaru a procesní řízení a modelování procesů.

Pro vlastní návrh procesu je nejdříve nutné provést analýzu současného stavu řízení softwarové implementace v Pojišťovně a popsat, jak v jejím rámci probíhá testování. Současné procesy i následně navržené nové procesy budou graficky zpracovány do podoby BPMN diagramů.

1.2 Úkoly práce

Cíl práce uvedený v předchozí kapitole lze rozvést do několika dílčích úkolů:

1. Představení Pojišťovny
2. Charakteristika IT sekce Pojišťovny
3. Uvedení teorie relevantní pro diplomovou práci
4. Analýza současného stavu řešené problematiky v Pojišťovně
    a. Popis současného stavu softwarové implementace
    b. Analýza zapojení testingu do procesu softwarové implementace
5. Návrh inovací procesu implementace softwaru
6. Doporučení pro zavedení navržených inovací
7. Shrnutí a vyhodnocení výsledků práce

1.3 Osnova práce

Úvod
První kapitola představuje cíl diplomové práce (DP), naznačuje postup jejího zpracování a vymezuje v práci využité technické a manažerské nástroje. Dále uvádí dílčí úkoly, jejichž splnění je nutné pro dosažení cíle DP. Kapitolu uzavírá stručný popis obsahu jednotlivých částí DP.

Charakteristika řešené společnosti
V druhé kapitole je představena Pojišťovna, pro kterou je návrh inovace procesu softwarové implementace vypracován. Kromě samotné Pojišťovny je v této kapitole také představena IT sekce, jejímiž procesy se tato práce zabývá.

Relevantní teorie
V rámci třetí kapitoly je sepsána základní teorie z relevantních zdrojů, o které se následně opírá vlastní návrh DP. Teoretická část této práce je rozdělena do 4 klíčových oblastí:

- Základy vývoje softwaru
- Základy testování softwaru
- Implementace softwaru
- Tvorba podnikových procesů

Analýza současné situace
Čtvrtá kapitola se zabývá analýzou současného procesu implementace softwaru v Pojišťovně. Důraz je kladen především na aktuálně používané techniky a metody, jejich výhody, nevýhody a návaznosti na zbytek vývojového cyklu. Analýza je vypracována na základě rozboru interní dokumentace a rozhovorů s pracovníky IT sekce Pojišťovny.

Návrh řešení
Pátá kapitola se zaměřuje na vlastní návrh inovací procesu implementace softwaru. Vychází z analýzy současné situace a relevantní teorie, představené v předchozích kapitolách.

Doporučení k zavedení inovací
Šestá kapitola se zabývá doporučeními pro co nejefektivnější zavedení navržených inovací procesu implementace do současné situace v IT sekci Pojišťovny.

Závěr
V závěru jsou shrnuty výsledky práce, hodnotí se, zdali se podařilo dosáhnout stanoveného cíle a dílčích úkolů diplomové práce. Jsou zde také uvedeny případné možnosti dalšího rozšíření DP.

Seznam použitých odborných pojmů
Po závěru následuje seznam nejdůležitějších odborných pojmů použitých v rámci této diplomové práce.

2 Charakteristika řešené společnosti

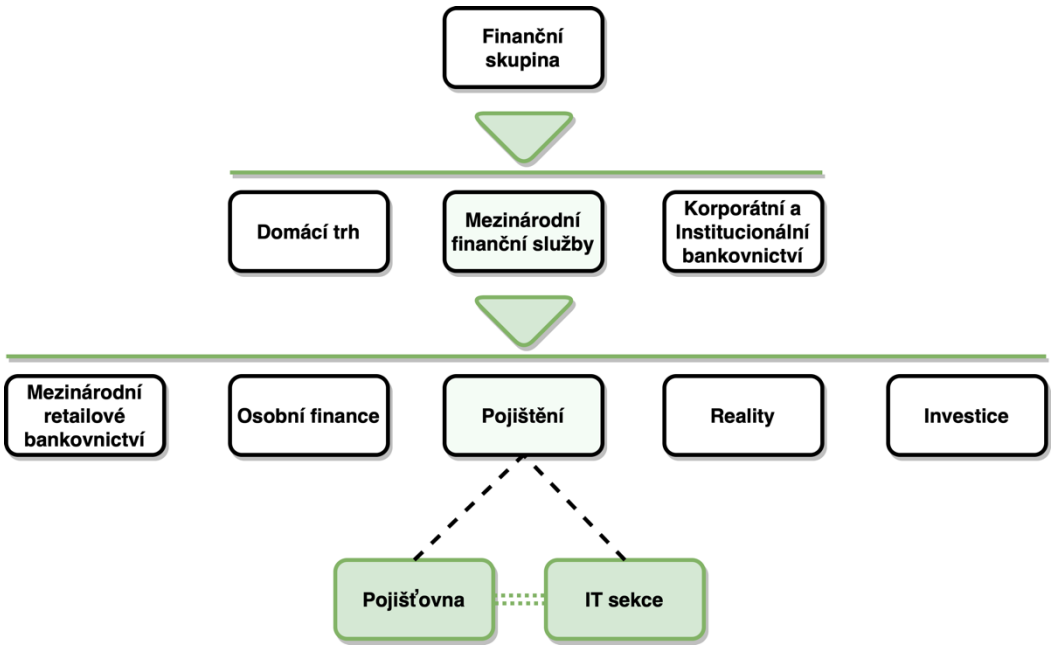
Tato kapitola stručně představuje českou pobočku nadnárodní pojišťovny, zvláště pak její IT sekci, pro níž bude návrh inovace procesu softwarové implementace vypracován. Jméno Pojišťovny bude na žádost vedení firmy v rámci této diplomové práce anonymizováno.

2.1 Pojišťovna

Pojišťovna, jíž se diplomová práce zabývá, má právní podobu akciové společnosti a je součástí jedné z největších finančních skupin světa. Historie jednotlivých bank a pojišťoven, ze kterých se dnes skupina skládá, sahá až do 19. století. V současnosti působí skupina ve více než 60 zemích světa napříč 5 kontinenty a zaměstnává přes 180 tisíc zaměstnanců. V rámci svého produktového portfolia nabízí svým zákazníkům (fyzické osoby, firmy, podnikatelé a korporace) služby zahrnující financování, pojištění, spoření a investice (1).

Na český trh se Pojišťovna uvedla v devadesátých letech pomocí produktu Pojištění schopnosti splácet finanční závazek, který v té době nabízela jako jedna z prvních. Dnes čítá portfolio Pojišťovny desítky pojišťovacích produktů, které nabízí ve spolupráci s řadou domácích i mezinárodních partnerů. Mezi partnery Pojišťovny se řadí banky, leasingové a úvěrové společnosti ale i prodejci elektra, telekomunikací, nebo dalších služeb/zboží (1).

Diagram na následujícím obrázku zobrazuje vazbu mezi Pojišťovnou, její IT sekcí a nadřazenou mateřskou skupinou:



Obrázek 1: Postavení Pojišťovny v rámci skupiny (vlastní zpracování)

2.2 IT sekce

IT sekce Pojišťovny má právní formu společnosti s ručením omezeným se sídlem v Praze a je vlastněna mateřskou firmou Pojišťovny. Důvodem pro její založení byla potřeba sdíleného servisu pro oblast zemí kontinentální Evropy.

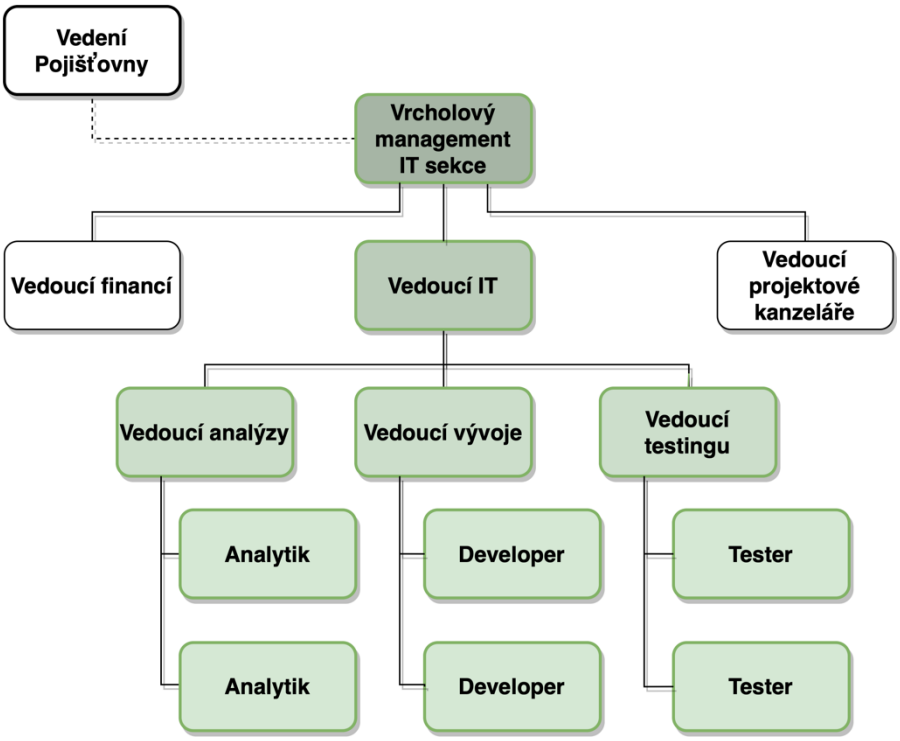
Cílem tohoto centra sdíleného servisu je centralizace, standardizace a efektivní využívání procesů, systémů a aplikací napříč pojišťovnami ve všech zapojených zemích. Tato diplomová práce se zaměřuje na IT sekci sdíleného servisního centra. Mezi další oblasti, kterými se centrum zabývá patří procesy, pojistná matematika, řízení projektů, finanční reporting, marketing a další. Výhradními odběrateli těchto služeb jsou pouze společnosti z mateřské skupiny. To činí Pojišťovnu spolu s mateřskou skupinou jedinými zdroji příjmů centra sdíleného servisu (1).

2.2.1 Organizační struktura

Vrcholový management sdíleného servisního centra je úzce napojen na vedení Pojišťovny a její mateřskou skupinu. Rozhodování o firemní politice, strategických, finančních a personálních záležitostech je tak přímo napojeno na strategické cíle skupiny.

Střední neboli taktickou úroveň managementu zajišťují vedoucí jednotlivých sekcí (IT, projektová kancelář, finance, …). Ti jsou zodpovědní za řízení, plánování a rozhodování ve střednědobém horizontu.

Liniový management IT sekce má na starost tři hlavní oddělení, která jsou klíčová při vývoji a údržbě aplikací – analýzu, vývoj a testing. Linioví manažeři pak přidělují své podřízené na aktuálně probíhající projekty, či do stálých pracovních skupin složených ze členů všech tří oddělení. Základní organizační struktura IT sekce je graficky znázorněna diagramem na obrázku 2.



Obrázek 2: Organizační struktura IT oddělení (vlastní zpracování)

Následující seznam stručně popisuje základní činnosti vykonávané jednotlivými odděleními:

- *Analýza* – analytici jsou zodpovědní za zpracování návrhu koncepce řešení dle vstupů získaných od ostatních oddělení Pojišťovny, která používají software vyvíjený IT sekcí (likvidace, registrace, call centrum, finance, …) nebo od legislativních regulátorů (ČNB, EU, GDPR, …). Výsledná analýza pak popisuje, jak má řešená funkcionalita/aplikace fungovat, jak toho bude dosáhnuto a jak budou řešeny případné integrace s dalšími systémy. Analýzy bývají často doprovázeny schématickými diagramy (2).

- *Vývoj* – vývojáři, nebo také developeři, mají za úkol převést analýzu do finálního produktu. Aplikaci, nebo novou funkčnost pak nejenom vyvíjí, ale jsou také zodpovědní za opravy případných chyb v kódu (3).

- *Testing* – hlavní činností testingu je kontrola kvality vyvinutých funkčností/aplikací a jejich integrace s dalšími systémy používanými v rámci skupiny. Testeři jsou ale aktivní během celého životního cyklu vývoje softwaru – připravují testovací data, podílejí se na tvorbě a kontrole analýzy a testují jednotlivé verze vyvíjeného systému (4).

Role zapojené do procesu implementace softwaru jsou podrobně rozepsány, včetně jejich odpovědností, v kapitole 4.2.1.

2.2.2 Vyvíjený software

Na základě požadavků sesbíraných napříč zeměmi, kde centrum sdíleného servisu působí, čítá aplikační portfolio řadu systémů, které jsou průběžně rozšiřovány novými funkcemi žádanými od jejich uživatelů. Mezi nejvýznamnější aplikace patří:

- Správa likvidace – program zajišťující komplexní správu agendy spojené s likvidací pojistných událostí
- Správa importů – software sloužící ke zpracování dat od partnerů
- Portál pro zákazníky – online portál, přes který mohou zákazníci nahlásit své pojistné události
- Bankovní reporting – program sloužící ke generování reportů pro banky
- Bankovní příkazy – software pro automatickou správu bankovních příkazů

3 Relevantní teorie

Cílem této kapitoly je představit pojmy, postupy, metody a teoretická východiska použité v této diplomové práci. Kapitola se skládá ze čtyř hlavních částí. První se věnuje základům teorie životního cyklu vývoje softwaru. V druhé části je představena teorie testování softwaru. Třetí část se zabývá implementací softwaru a čtvrtá část řeší zásady tvorby a inovace firemních procesů.

3.1 Základy vývoje softwaru

Jak je psáno v (5), charakter podniku, jeho produktů a aplikací zásadně ovlivňuje výběr metod a postupů pro vývoj, údržbu a případný rozvoj podnikového softwaru. Tyto postupy a řešení vývoje softwaru jsou v IT označovány jako metodiky. Je běžné, že firmy používají některou ze standardizovaných metodik jako jsou Agile nebo Waterfall (vysvětleno v další kapitole), případně mají metodiku vlastní, ale inspirovanou nějakou již existující.

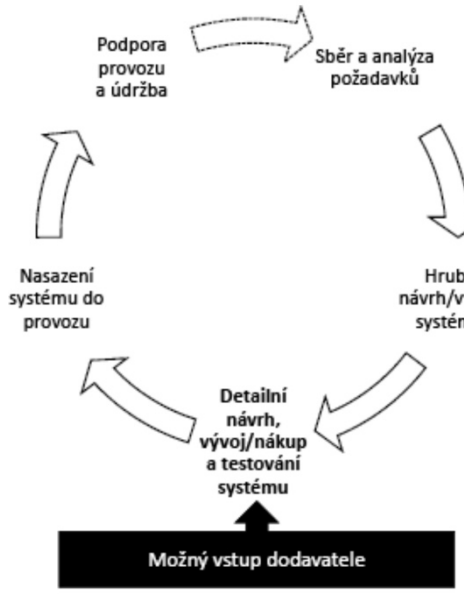
3.1.1 Životní cyklus vývoje softwaru

Životní cyklus vývoje softwaru, často také označovaný svým anglickým názvem Software Development Life Cycle (zkratka SDLC), představuje základní kroky potřebné k vývoji a následnému provozu jakéhokoliv systému složeného z hardwaru a souvisejícího softwaru (6). SDLC je v literatuře dělen do 5 fází (7):

- Sběr a analýza požadavků
- Hrubý návrh/výběr systému
- Detailní návrh, vývoj/nákup a testování systému
- Nasazení systému do provozu
- Podpora provozu a údržba

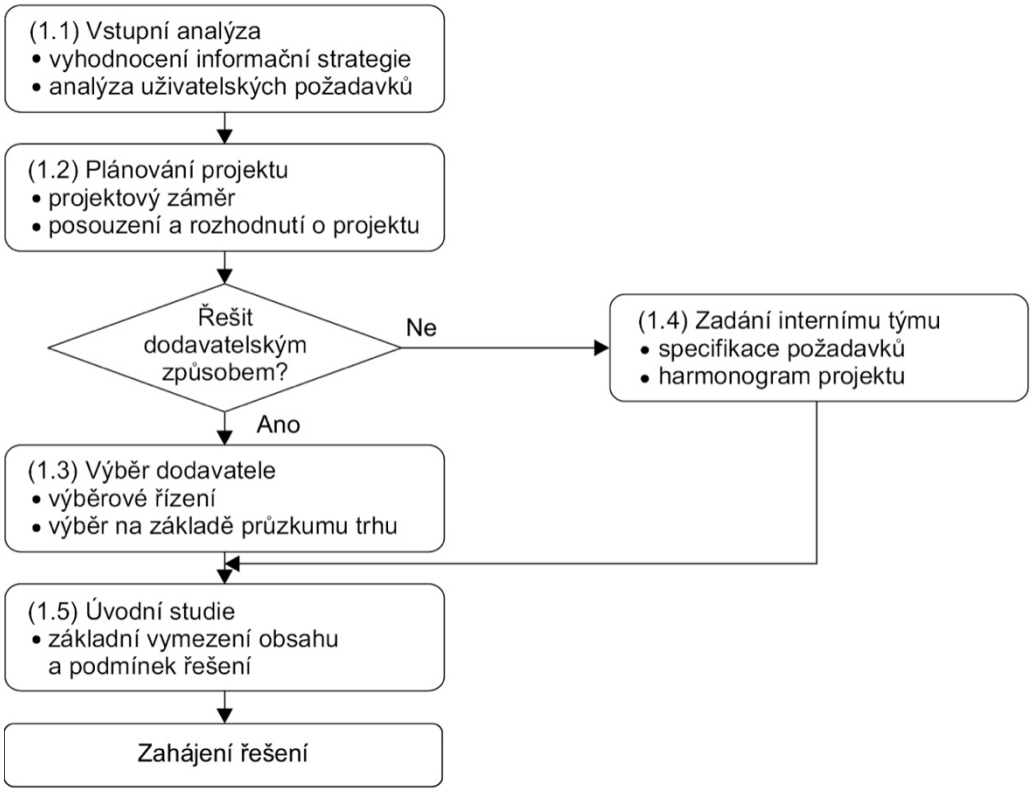
Průběh životního cyklu vývoje a provozu systému je graficky znázorněn na obrázku 3 a je následován popisem jednotlivých fází SDLC.

Cílem SDLC je vyjádřit dlouhodobou perspektivu, ve které je řešený software/systém používán. Životní cyklus končí až vyřazením softwaru z provozu. Označení cyklus vychází z toho, že SDLC popisuje celý průběh vývoje systému, včetně dalšího rozvíjení původního systému. Každý systém má jen omezenou životnost a po určité době nastanou u jeho uživatelů nové potřeby a požadavky, nebo dojde k posunu v technologických možnostech. V tomto okamžiku se proces vývoje vrací zpět do své první fáze a zacyklí se (5).



Obrázek 3: Životní cyklus vývoje a provozu systému (7)

– *Sběr a analýza požadavků* – první fáze SDLC, během které je vypracována vstupní analýza a plán projektu. Na základě těchto dokumentů je rozhodnuto, zdali se bude návrh realizovat, a když ano, tak jestli interně, nebo dodavatelským způsobem. Průběh fáze je zobrazen na obrázku 4.



Obrázek 4: Sběr a analýza požadavků na nový software (5)

– *Hrubý návrh/výběr systému* – během druhé fáze je vypracován konceptuální návrh systému, popisující cílové chování a pravidla systému (často používaný název Business Logic). Případně je vybrán již existující software.

– *Detailní návrh, vývoj/nákup a testování systému* – v této fázi probíhá vývoj řešeného systému/aplikace. Vývoj probíhá buď interně, nebo je řešen dodavatelským způsobem. Výstupem fáze je kompletní a otestovaný software. Další možností je koupit licenci na již existující software, vybraný v předchozí fázi.

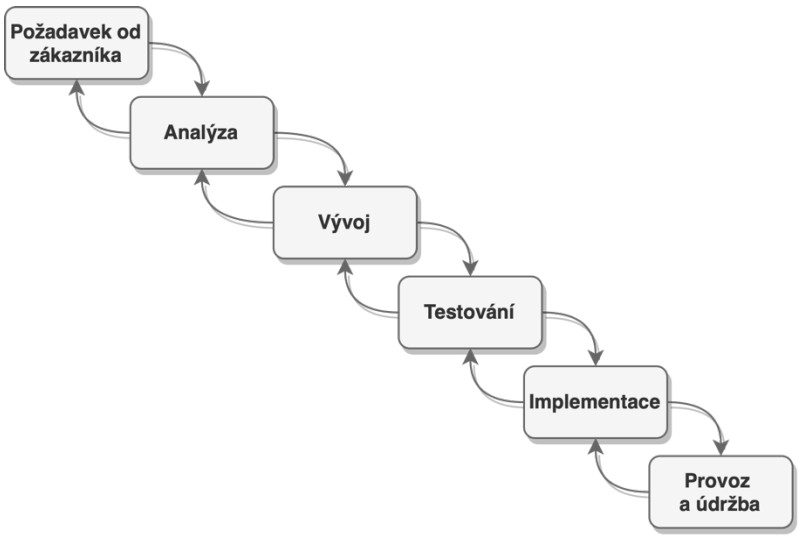
– *Nasazení systému do provozu* – jakmile je software vyvinut a řádně otestován svými budoucími uživateli, přichází fáze nasazení do provozu neboli implementace, kdy je software nasazen na produkční prostředí a je zpřístupněný svým koncovým uživatelům.

– *Podpora provozu a údržba* – po celou dobu, kdy je software v provozu, je třeba zajistit jeho údržbu a technickou podporu uživatelům. V případě, že vzniknou nové požadavky na funkčnosti systému, znovu se rozbíhá první fáze SDLC, tj. sběr a analýza požadavků.

Průběh druhé a třetí fáze životního cyklu vývoje softwaru, konkrétně návrh a vývoj systému, jsou nejvíce rozdílné napříč různými metodikami. Mezi nejběžnější metodiky patří vodopádový model a agilní metodika vývoje. Oba přístupy jsou podrobněji rozepsány v následujících kapitolách.

3.1.2 Vodopádový model

Vodopádový model vychází z tradičního inženýrského přístupu a aplikuje ho na problematiku vývoje softwaru. Jedná se o takzvaný sekvenční přístup. Vývoj je rozdělen do několika navazujících fází, které jsou vykonávané v přesně daném pořadí, kdy nelze začít s další fází, dokud není ta předchozí uzavřena, zdokumentována a schválena (6). Průběh vývoje dle vodopádového modelu je zobrazen diagramem na obrázku 5.



Obrázek 5: Proces vývoje softwaru dle vodopádového principu (vlastní zpracování)

Vodopádový model je jednoduchý na pochopení i řízení a poskytuje komplexní dokumentaci dodávaného softwaru, ale vzhledem k tomu, že se jedná o striktně sekvenční a lineární metodu, váže se k němu i několik zásadních nedostatků (8):

- *Nedostatečná flexibilita* – model nedokáže dostatečně pružně reagovat na změny v zadání. Jakákoliv změna vrátí proces na začátek.
- *Pozdní implementace* – zákazník uvidí produkt až na úplném konci procesu, v průběhu vývoje se nemůže podívat na prototyp.
- *Nekompletní požadavky* – je téměř jisté, že se v původní analýze něco opomene a odhalení tohoto nedostatku může nastat až ve fázi testování, či implementace. Zvláště pokud se jedná o náročný projekt s dobou vývoje v řádech měsíců.

I přes své nedostatky se vodopádový model stále používá, a v případě malých projektů, kde se neočekává změna zadání v průběhu vývoje, může být efektivní (6).

3.1.3 Agilní vývoj

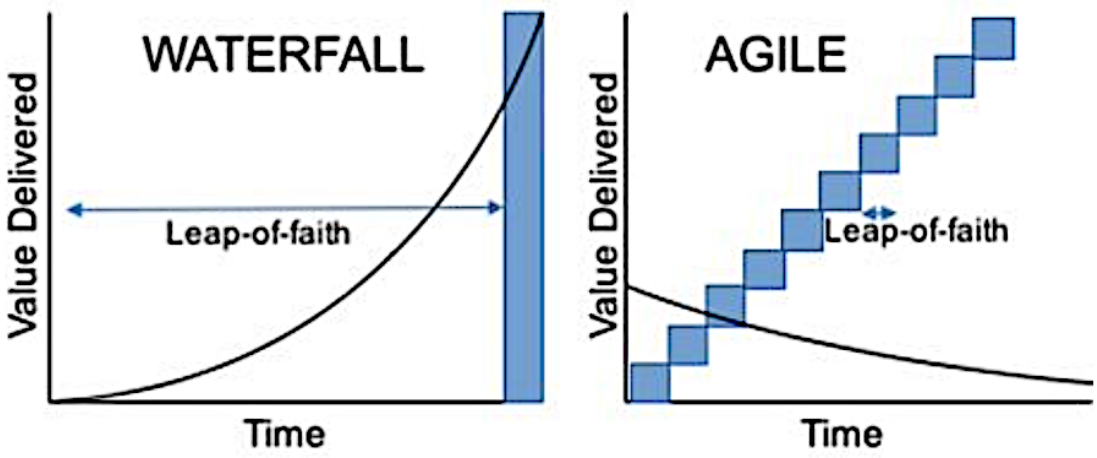
Idea agilního vývoje, mnohdy označovaného pouze jako agile, se objevila již v polovině devadesátých letech minulého století, ale formalizována byla až v roce 2001, kdy skupina sedmnácti IT odborníků společně sepsala Manifest agilního programování. Manifest předkládá čtyři hlavní principy, jimiž se agilní vývoj řídí (9).:

- Jednotlivci a interakce před procesy a nástroji
- Fungující software před vyčerpávající dokumentací
- Spolupráce se zákazníkem před vyjednáváním o smlouvě
- Reagování na změny před dodržováním plánu

Agile je svým přístupem k vývoji protikladem vodopádového modelu. Místo liniového sekvenčního procesu, který těžko zvládá změny, představuje agile iterativní metodu založenou na co nejčastějších dodávkách zákazníkovi a získávání průběžné zpětné vazby.

Proces vývoje softwaru následuje obdobný životní cyklus se všemi jeho fázemi jako vodopádový model, ale liší se ve dvou zásadních bodech (6):

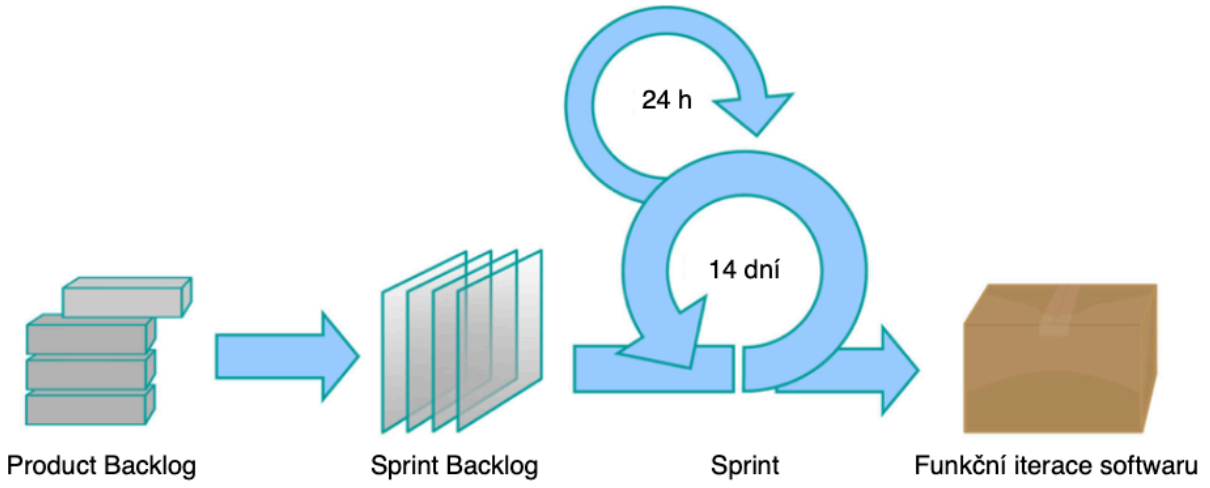
- Zadání a požadavky se v agile mohou změnit v jakékoliv fázi projektu, dochází k jejich postupnému upřesňování.
- Agile začíná s dodávkami softwaru zákazníkovi co nejdříve, což pomáhá včas identifikovat problémy a opravit je již v průběhu raných fází vývoje. Tím je minimalizován možný dopad chyb na celý projekt



Obrázek 6: Dodávky v průběhu času (10)

Obrázek 6 zobrazuje rozdíl v dodávkách funkčností softwaru v průběhu času při použití vodopádového modelu a agile. Vodopádový model má dodávku až na úplném konci vývoje, takže vyžaduje jasně stanovené a neměnné zadání. Agile oproti tomu dodává software po malých funkčních celcích, takže je možné „agilně" reagovat na změny v zadání.

Aby bylo možné aplikovat agilní způsob vývoje, je zapotřebí sestavit tým se členy jak z vývoje, tak i analýzy a testingu. Role a odpovědnosti členů se mohou lišit v závislosti na specifikách konkrétních projektů. Všichni členové tohoto týmu pak úzce spolupracují jak mezi sebou, tak i s koncovým uživatelem.



Obrázek 7: Vývoj funkčnosti pomocí Agile (vlastní zpracování)

Pro použití agile je klíčové rozdělit projekt do jasně definovaných časových úseků (často označovaných jako sprinty) a přiřadit jednotlivé požadované funkcionality a činnosti do těchto sprintů. Seznam sprintů s jejich plánovanými činnostmi se nazývá Product Backlog a jeho velikost se odvíjí od náročnosti projektu. Cílem jednotlivých sprintů je kompletně dokončit danou funkcionalitu a dodat novou iteraci softwaru ke kontrole zákazníkovi.

V závislosti na zpětné vazbě od zákazníka se pak další sprint věnuje buď dalším činnostem z backlogu, nebo vývoji a integraci úprav/oprav funkcionality z předchozího sprintu. Průběh vývoje iterace softwaru je zobrazen na obrázku 7. Časy uvedené u sprintů se mohou lišit v závislosti na použité agilní metodice a zvyklostech ve firmě/týmu.

Agile se dále dělí na jednotlivé metodiky, které sdílí základní principy definované v agilním manifestu, ale liší se v tom, jak je konkrétně aplikují. Mezi nejpoužívanější agilní metodiky se řadí (6):

- *Scrum* – metodika založená na čtrnáctidenních sprintech s každodenními krátkými schůzkami pro koordinaci práce v rámci týmu. Na konci každého sprintu probíhá vyhodnocení uplynulého sprintu a v závislosti na tom plánování dalšího.
- *Kanban* – na rozdíl od Scrumu nepoužívá Kanban striktní časové milníky jako jsou čtrnáctidenní sprinty. Jediným omezením je počet rozpracovaných úkolů. Tým nemůže zároveň pracovat na více než 5 úkolech, a ty se snaží dokončit co nejdříve. Další úkoly jsou z backlogu přidány teprve, až jsou ty předchozí hotovy.

3.2 Základy testování softwaru

Testování, nebo také testing, je specifickou oblastí managementu kvality zabývající se softwarem. Konkrétně se jedná o proces zabývající se systémy a jejich součástmi s cílem určit, zda splňují stanovené požadavky a odhalení jejich případných nedostatků (11). Jednotlivé části testovacího procesu jsou popsány v následující kapitole.

3.2.1 Testovací proces

Testovací proces se skládá z několika úrovní. M. Bureš a kolektiv (7) definují úrovně testování jako skupiny společně řízených a organizovaných testovacích aktivit. Každá z úrovní má vlastní odlišné cíle a odpovědnosti, aby byla detekce chyb v řešeném systému co nejúčinnější (8).

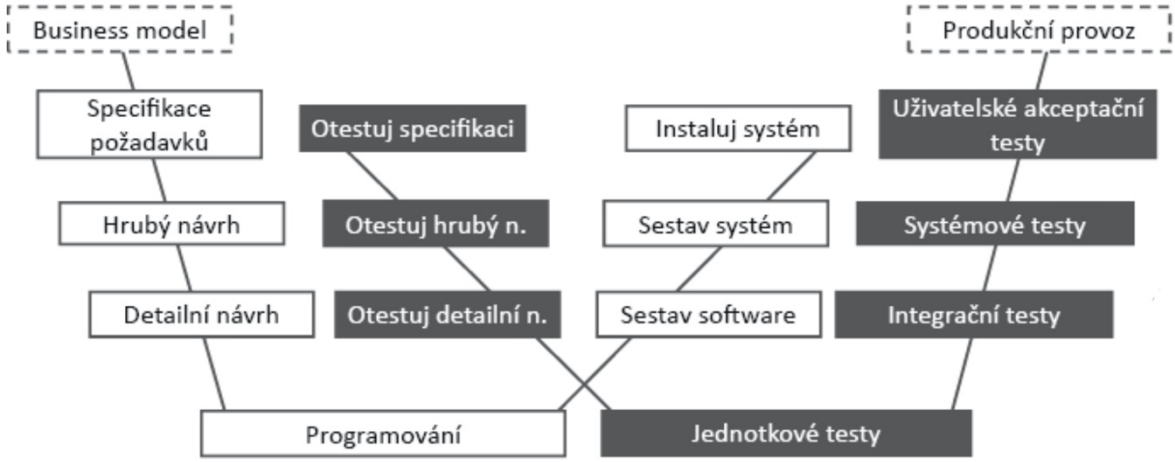
Koncept testovacích úrovní rozvádí W-model vysvětlený v další kapitole.

3.2.1.1 W-model testování v průběhu SDLC

W-model spojuje životní cyklus vývoje softwaru s jednotlivými úrovněmi testování a klade důraz na zapojení testingu v průběhu celého SDLC, od začátku procesu, přes sběr požadavků, až po samotné testování dodávaného softwaru a jeho implementaci.

Název W-modelu je odvozen od jeho podoby připomínající písmeno dvojité V, jak lze vidět na obrázku 8. Bílé obdélníky znázorňují činnosti vývoje aplikace od prvotního požadavku až po nasazení systému do produkčního prostředí. Tmavé obdélníky značí aktivitu testingového oddělení, která probíhá zároveň s danou fází vývoje. Levá část testingového V reprezentuje statické testování (nespouštíme při něm kód aplikace –

například validace analytických dokumentů) a pravá dynamické testování (při něm je již spouštěn nově vyvinutý kód) (7).



Obrázek 8: W-model (7)

Popis různých typů testů použitých v průběhu SDLC je poskytnut v kapitole 3.2.3 Typy testů.

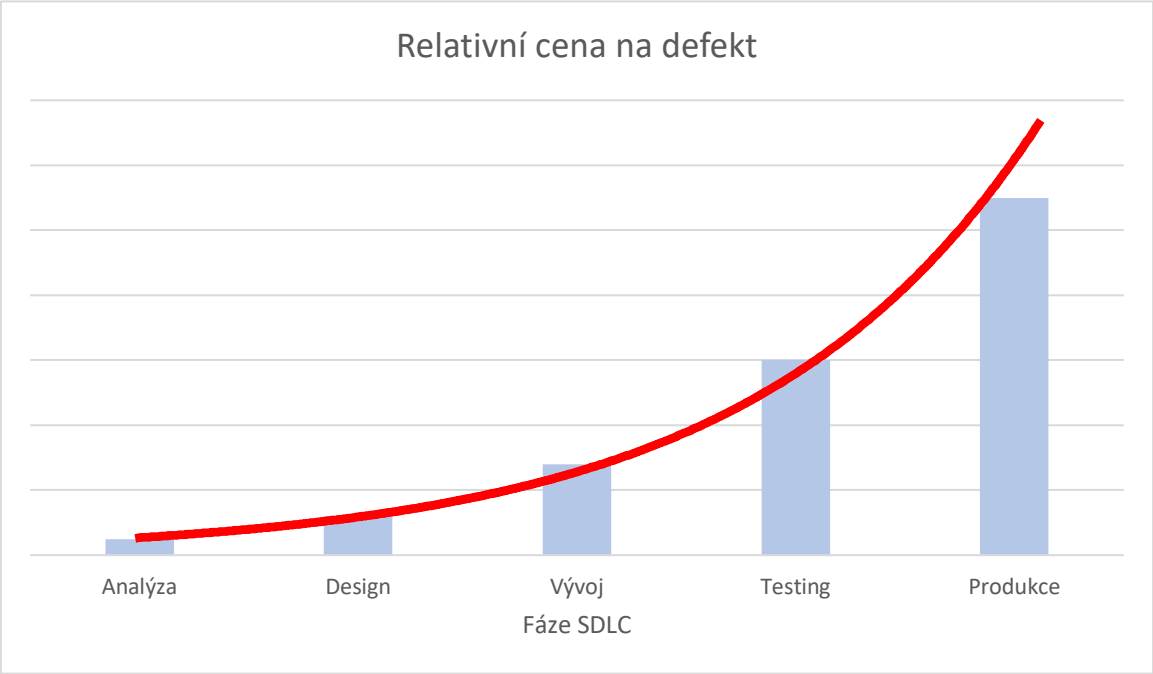
3.2.1.2 Cena testování v průběhu SDLC

Čím dále v SDLC jsme, tím náročnější a nákladnější je provádět změny v kódu a opravovat nalezené chyby (6). V reakci na to byl představen takzvaný model „Testovací pyramidy", který ukazuje ideální vzájemný poměr různých typů testů prováděných v jednotlivých fázích SDLC (7). Jak lze vidět na obrázku 9, největší důraz je kladen na testy v nižších úrovních, kdy je cena na opravu chyby nejnižší a zároveň její opravou zabráníme tomu, aby defekt přešel do další úrovně testování. Typy testů v pyramidě jsou převzaty z W-modelu.

Obrázek 9: Testovací pyramida (7)

Jak již bylo zmíněno, čím později v SDLC odhalíme chybu, tím dražší je její oprava. To je dáno tím, že čím později chybu najdeme, tím více už bylo odvedeno práce a zapojeno lidí a k opravě chyby je potřeba se vrátit do fáze, ve které vznikla. Tedy v případě, že se v produkci odhalí chyba v samotném návrhu funkcionality, je třeba se vrátit až na úplný začátek životního cyklu a začít znovu. Proto je třeba nalézt co nejvíce chyb již v prvních fázích SDLC, abychom minimalizovali relativní cenu na defekt (12).

Agilní přístup k vývoji minimalizuje relativní cenu na defekt tím, že dodává zákazníkovi produkt průběžně po menších částech. Graf 1 znázorňuje, jak roste relativní cena na chybu s jednotlivými fázemi SDLC.



Graf 1: Relativní cena na defekt v průběhu SDLC (vlastní zpracování dle (12))

3.2.2 Základní artefakty testingu

Testovací artefakty jsou nedílnou součástí testingu. Pojem artefakt zastřešuje všechny dokumenty, které jsou vytvářeny během testování softwaru. Jejich cílem je transparentně evidovat a komunikovat plán a průběh aktivit testování. Generované artefakty jsou pak sdíleny v rámci týmu, s manažery, se zákazníky a případně dalšími zúčastněnými stranami.

Následuje výčet základních testovacích artefaktů. Definice použité v této práci odpovídají terminologii používané v řešené Pojišťovně.

- *Testovací strategie* – dokument popisující celkový přístup k testování na daném projektu. Představuje testovací požadavky, test scope, použité typy testů, projektové milníky, harmonogram všech testingových aktivit, definuje prostředí pro provedení testů a specifikuje zapojené role a jejich odpovědnosti.
- *Testovací požadavky* – soupis požadavků na analýzu a pokrytí testy, definuje obsah uživatelských testů.
- *Test scope* – definuje veškeré součásti testování na daném projektu/požadavku.
- *Test analýza* – dokument shromažďující požadavky na danou funkcionalitu. Určuje, pomocí jakých testovacích scénářů bude funkcionalita otestována a definuje jejich očekávané výsledky.
- *Testovací scénář* – sada vstupů/akcí v testovaném systému a jejich očekávané výsledky. Nutnou součástí testovacího scénáře je popis výchozího a cílového stavu systému.
- *Testovací data* – specifikace dat, která jsou potřebná pro úspěšné provedení testovacího scénáře.
- *Bug* – jakákoliv chyba, nedostatek, či defekt v testované aplikaci/systému, která způsobuje nesprávné nebo neočekávané chování a neodpovídá výsledku definovaném v souvisejícím testovacím scénáři.
- *Testovací sada* – sada souvisejících (mnohdy i přímo navazujících) testovacích scénářů zabývající se stejnou oblastí testovaného systému. Používá se pro lepší přehlednost.
- *Test plán* – kolekce všech testovacích sad a testovacích scénářů, které budou provedeny v rámci daného projektu.
- *Test dashboard* – grafický report přehledně znázorňující aktuální stav a dosavadní průběh testování na projektu. Zobrazuje různé metriky, grafy a statistiky relevantní pro projekt. Například počet a seznam všech nalezených bugů, počet provedených testovacích scénářů, jejich výsledky a podobně.
- *Testovací prostředí* – specifikace prostředí (soubor hardwaru, softwaru a databází, na kterém běží řešené systémy) na kterém budou prováděny testy a jaké podmínky musí prostředí splňovat (napojení na SMS bránu, anonymizovaná data, propojení s dalšími systémy a další).

3.2.3 Typy testů

Tato kapitola představuje základní typy testů. Použité definice vycházejí z glosáře ISTQB (11) a jsou upraveny tak, aby reflektovaly realitu testingového oddělení IT sekce Pojišťovny. ISTQB glosář (International Software Testing Qualification Board Glossary) je slovník pojmů spojených s testováním softwaru, který vydává a pravidelně aktualizuje ISTQB (mezinárodní certifikační organizace v oboru softwarového testování).

- *Statický test* – testování, které nezahrnuje spouštění kódu softwaru. Například kontrola syntaxe kódu, revize analytických dokumentů atd.
- *Dynamický test* – testování nad spuštěným kódem softwaru, opak statických testů.
- *Front-end test* – test uživatelského rozhraní aplikace (to co vidíme a můžeme vykonávat v samotné aplikaci nebo na webové stránce).
- *Back-end test* – protiklad front-end testování. Zaměřuje se na testování komponent skrytých za GUI[1]. Například servery, databáze, výpočty atd.
- *End-to-end test* – test simulující určitou funkcionalitu systému od začátku do konce procesu. V případě Pojišťovny by to byl například test začínající nahlášením pojistné události a končící vyplacení pojistného plnění klientovi.
- *FAT (Factory Acceptance Testing)* – ověřuje, jestli nově vyvinutá funkcionalita splňuje všechny stanovené požadavky zadání.
- *Sanity Check* – testovací sada složená ze základních testovacích scénářů, jejichž cílem je ověřit stav řešeného systému a jeho připravenosti na další testování. V literatuře se často Sanity Check uvádí i pod názvem Smoke Testing.
- *UAT (User Acceptance testing)* – testy vykonávané zadavateli požadavků na software. Podobně jako u FAT je cílem UAT porovnat dodaný software se zadáním. Rozdíl je v tom, že FAT jsou prováděny interně testingem a UAT provádí zákazník. Na základě výsledku User Acceptance Testing je následně rozhodnuto, jestli je nově vyvinutý software připraven k nasazení na produkčního prostředí, nebo je ještě potřeba úprav.
- *SIT* (System Integration Testing) – má za cíl ověřit komunikaci mezi propojenými systémy (např. účetní systém na zadávání faktur a internetové bankovnictví).
- *Regresní testy* – testy používané v případě, kdy dochází k integraci nové funkcionality do již existujícího softwaru. Jejich cílem je ověřit, jestli nově přidané funkcionality neovlivnily dosavadní stav systému a nezpůsobily nové bugy.

[1] GUI je zkratkou Graphic User Interface, jedná se o tu část aplikace, která umožňuje uživatelům interakci se softwarem prostřednictvím grafický ikon a vizuálních indikátorů (11).

Regresní testy jsou prováděny pro všechny systémy integrované se změněným softwarem.

- *RAT* (Release Acceptance Testing) – ekvivalent regresního testování prováděný koncovými uživateli softwaru. Uživatelé testují všechny potencionálně zasažené systémy a ověřují, jestli fungují tak jak mají. V případě že jsou uživatelé spokojeni s výsledky RAT, dojde k podepsání akceptačního protokolu a software může být nasazen na produkční prostředí.
- *Automatické testy* – testy, které nejsou prováděny manuálně testery, ale jsou spouštěny automatizačními nástroji. Tyto nástroje jsou schopny automaticky vykonávat testy podle jasně nadefinovaných kroků a očekávaných výsledků. Používají se z důvodů úspory času a financí.
- *Jednotkové testy* – testování nejmenších možných spustitelných celků kódu. Testy jsou obvykle automatizovány. Jejich údržbu a vyhodnocování má na starost vývoj.
- *Explorativní testy* – testování, bez formálních testovacích scénářů, zaměřené na „outside the box" přístup. Tester prochází software na základě svých zkušeností a hledá chyby, které by nemusely být nalezeny při pouhém postupování dle jasně nadefinovaných testovacích scénářů.
- *Zátěžové (Performance) testy* – hodnotí, jak software funguje z hlediska odezvy a stability při určitém zatížení. Provádějí se obvykle za účelem prověření rychlosti, stability a spolehlivosti systému.

3.2.4 Další pojmy spojené s testingem a vývojem

- *Pull request* – v okamžiku, kdy je developerem nově napsaný kód připraven na zahájení procesu sloučení s hlavní větví kódu, je založen pull request, neboli žádost o sloučení kódu.
- *Build* – proces převedení nově napsaného kódu do spustitelného softwaru. Každá změna či oprava kódu potřebuje svůj vlastní build.
- *Deployment* – proces spuštění (nasazení) nového software buildu na konkrétním prostředí (testovací, produkční). Součásti procesu deploymentu jsou instalace, konfigurace a otestování nasazení.
- *Release* – proces implementace nové verze softwaru a jejího zpřístupnění koncovým uživatelům do provozu.

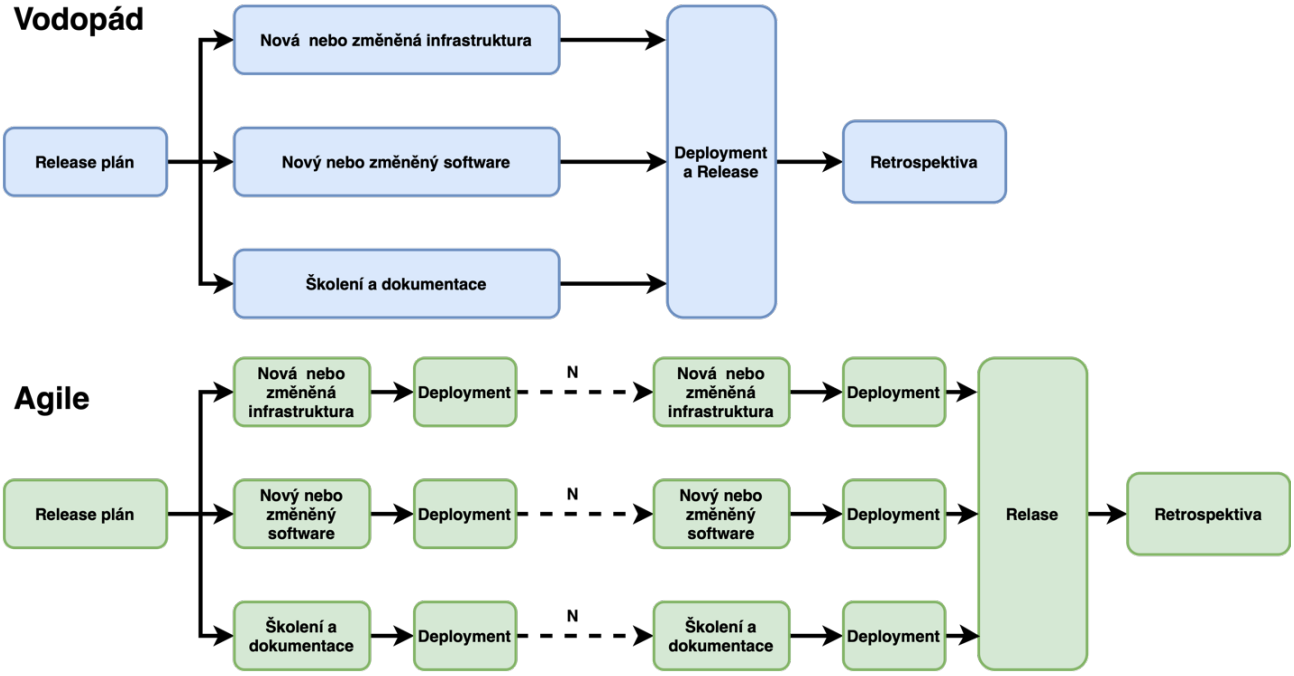
3.3 Implementace softwaru

Proces implementace softwaru, v IT často nazývaný anglickým názvem Release management, je část SDLC zodpovídající za zpřístupnění nových či upravených funkcionalit a aplikací jejich koncovým uživatelům (11).

Release v sobě může kromě samotného softwaru zahrnovat také dokumentaci, školení (pro uživatele, nebo IT), aktualizované procesy, nové nástroje a jakékoliv další požadované komponenty.

Jednotlivé implementace se od sebe mohou lišit velikostí, od releasů obsahující pouze drobnou změnu jedné funkcionality, až po velmi rozsáhlé releasy zasahující do celého aplikačního portfolia firmy. Obsah releasu je definován v takzvaném Release plánu, dokumentu definujícím obsah a časový harmonogram implementace. Komponenty releasu mohou být vyvíjeny a dodávány jak interně, tak i externě (11).

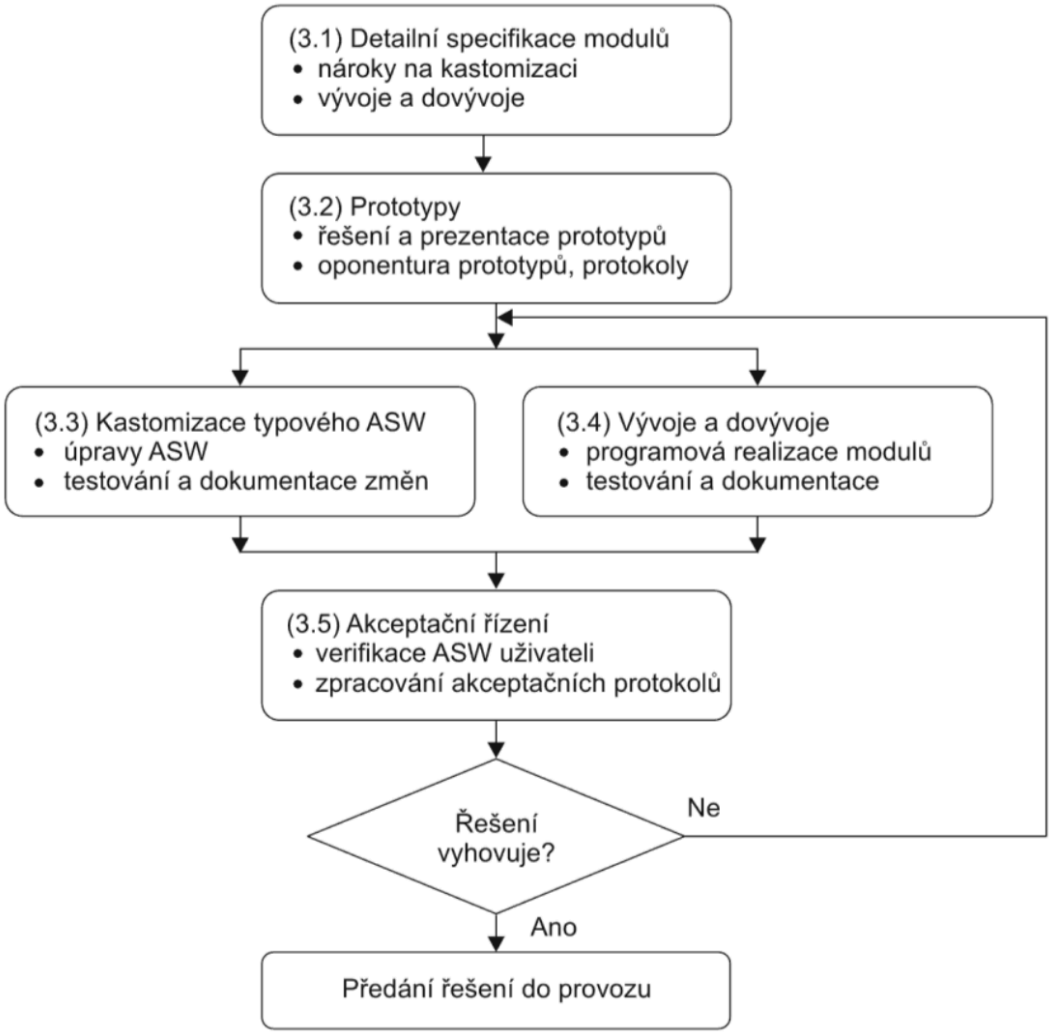
Rozdíl průběhu procesu implementace softwaru ve vodopádovém a agilním modelu zobrazuje následující obrázek:



Obrázek 10: Implementace softwaru při použití vodopádového modelu a Agilu (vlastní zpracování)

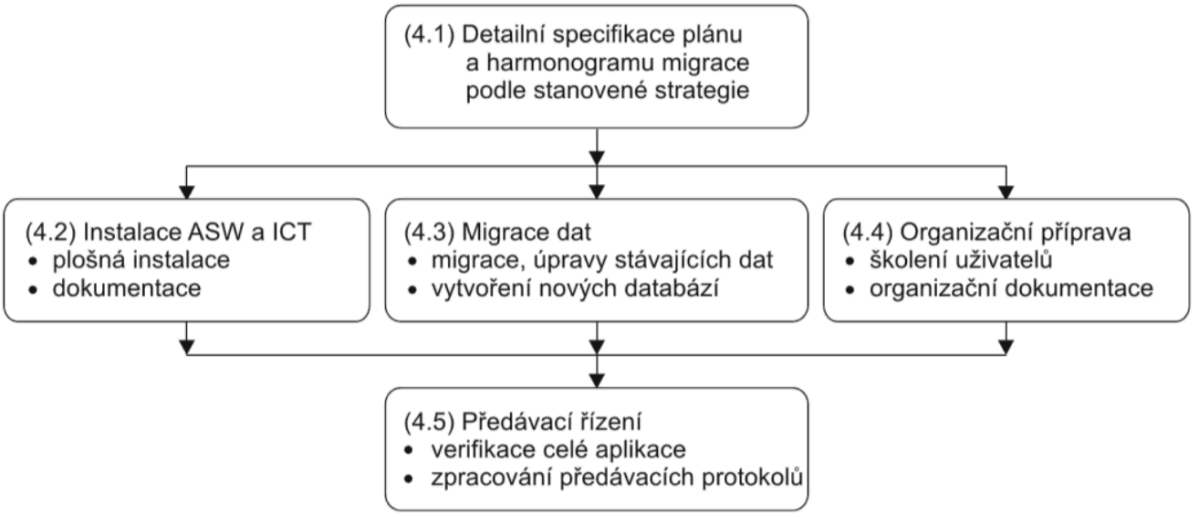
První (modrý) diagram zobrazuje klasický vodopádový model. Druhý (zelený) diagram ukazuje průběh implementace při použití agilních metodik. Agile odděluje deployment od releasu, může tak dojít k vyvinutí a nasazení softwaru v několika vlnách před samotným releasem. Finální činnost implementačního procesu, retrospektiva, slouží jako ohlédnutí se za releasem, vyhodnocení jeho průběhu a sepsání změn a poučení pro další releasy.

Nedílnou součástí releasu je testování prováděné testingovým týmem a release akceptační testy (RAT) prováděné uživateli. Úspěšné RAT jsou zakončené podepsáním akceptačního protokolu zástupcem businessu, stvrzující nasazení na produkci. Jednotlivé činnosti procesu implementace zobrazuje diagram na obrázku 11:



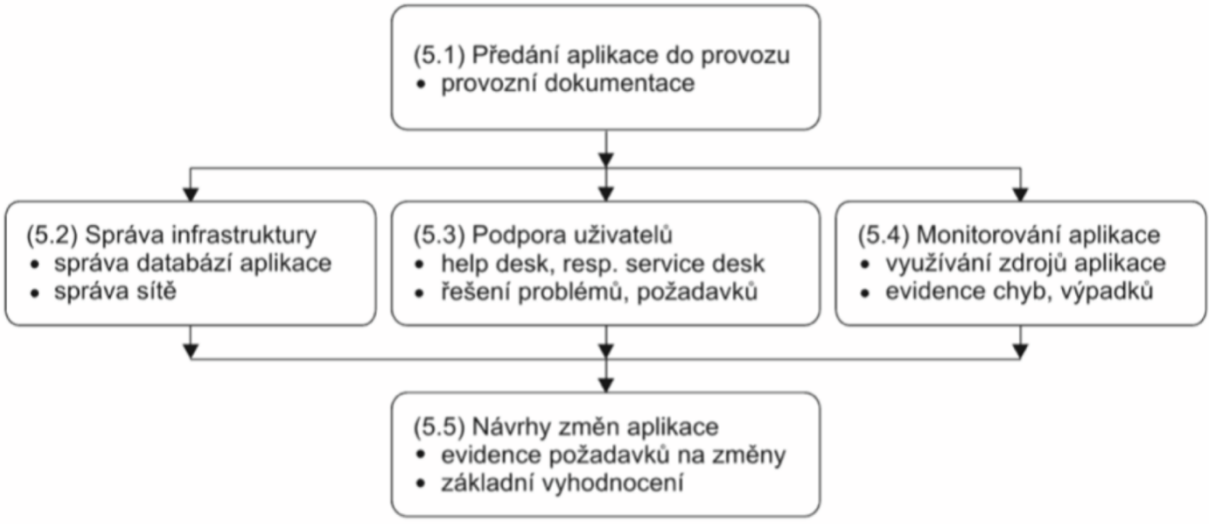
Obrázek 11: Činnosti v rámci procesu implementace softwaru (13)

Zkratka ASW použitá v diagramu se používá pro aplikační software. Poslední činnost diagramu – Předání řešení do provozu, je rozdělena do dvou sub-procesů. Prvním je Příprava na zavedení do provozu a migrace dat, druhým je Zavedení do provozu, provoz a údržba (13). Sub-procesy jsou zobrazeny na obrázcích 12 a 13:



Obrázek 12: Příprava na zavedení do provozu a migrace dat (13)

Ihned po nasazení releasu na produkci, nebo ještě v průběhu samotného releasu dochází k plánování následujícího releasu a vývoji jeho obsahu.



Obrázek 13: Zavedení do provozu, provoz a údržba (13)

3.4 Tvorba podnikových procesů

Podnikové procesy jsou definovány jako soubor činností, jejichž pomocí je postupně přetvořen souhrn vstupů na souhrn výstupů. Tento proces je umožněn díky zapojení finančních, hmotných, nehmotných a lidských zdrojů. Koncovým uživatelem procesu může být jiné oddělení v organizaci, zákazník, či další proces (14).

Podstata řízení podniku pomocí procesů tkví v tom, že podnik se správně nastavenými procesy se takzvaně „řídí sám" – každý pracovník v podniku zná své místo v procesu, návaznosti v činnostech jsou jasně dané a všechny možné variantní stavy jsou přesně popsané. Při správně nastavených procesech je tak zásah manažera nutný jen v mimořádných situacích (15).

Podnikové procesy je nutné řídit, průběžně aktualizovat a zlepšovat. Za tímto účelem je nutné v organizaci používat procesní analýzu, kterou se zabývá následující kapitola.

3.4.1 Procesní analýza

Procesní analýza se používá k odhalení neefektivity v procesech a nalezení jejich možného vylepšení. Předpokladem pro použití této analýzy je, že řešené procesy jsou srozumitelné a odpovídají skutečnému stavu v organizaci. Základním principem procesní analýzy je analýza jednotlivých procesů a jejich vnitřní logiky (16).

Cílem procesní analýzy použité pro vyhodnocení současných procesů je nalézt problémy (například nejednoznačně definované odpovědnosti, nedostatek informací nebo organizační bariéry) a identifikovat všechny činnosti, které nepřispívají k výsledku procesu a jsou tudíž redundantní (16).

Výsledek procesní analýzy se nazývá konceptuální procesní model. Ten slouží jako podklad pro zavádění změn ve stávajících procesech a pro vytvoření potřebných technických a organizačních podmínek pro implementaci upravených procesů a jejich správné fungování v kontextu řešené organizace (14). Všechny podnikové procesy by před svou implementací měly být zbaveny všech neproduktivních a neefektivních činností, zbytečných nákladů a nekvalitních výstupů (17).

3.4.2 Tvorba procesů

Předtím, než se začne s tvorbou, či úpravou procesů, je nejdříve nutné stanovit vlastníka procesu, který bude zodpovídat za jeho návrh, implementaci a dodržování (18). Samotná tvorba nových procesů se dělí do čtyř základních fází:

1. *Mobilizace* – fáze, která se soustředí na sestavení pěti až devíti členného týmu, který bude mít tvorbu a implementaci procesu na starost. Počet 5 až 9 členů je zdůvodněn tím, že nabízí dostatečnou rozmanitost, ale stále se dá ještě efektivně řídit. Jednotliví členové by měli pocházet z různých oddělení napříč firmou, a to nejen z těch, jichž se proces týká.

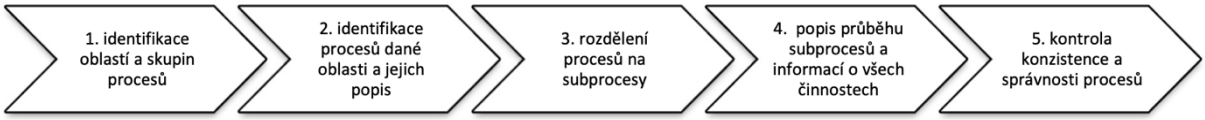
2. *Diagnostika* – fáze během které nově sestavený tým provádí procesní analýzu současných procesů s cílem identifikovat neefektivní a zbytné činnosti. Výstupem diagnostiky často bývají diagramy zakreslené v BPMN[2] standardu. Tomuto standardu se věnuje samostatná kapitola 3.4.3.

3. *Nové rozvržení* – po zhodnocení procesní analýzy z předchozí diagnostické fáze přechází tým k modelování nových procesů, které jsou již očištěny o veškeré neefektivnosti a řídí se zásadami procesního managementu. Důležitou součástí této fáze je simulace průběhu nového procesu. Během simulace je třeba zapojit zástupce všech funkčních útvarů, kterých se proces týká, aby mohli poskytnout zpětnou vazbu a případně identifikovat nedostatky návrhu. Pokud je třeba větších zásahů do nového procesu, je vhodné provést novou simulaci.

4. *Změna* – poslední fází je změna neboli implementace finální podoby procesu. Mnohdy probíhá prvotní implementace jen v pilotním režimu, a to například jen pro jednoho partnera, nebo na jednom produktu. Všechny zainteresované strany musí být s předstihem obeznámeny s podstatou procesní změny. V případě, že v pilotním provozu nebyly identifikovány žádné závažné nedostatky, je nový proces implementován do operací celého podniku.

[2] BPMN je zkratkou pro Business Process Modelling Notation (19).

Postup, vedoucí k návrhu a modelování procesu ve třetí fázi tvorby procesů, je podrobně rozveden do následujících pěti kroků (16):



Obrázek 14: Postup procesního modelování (16)

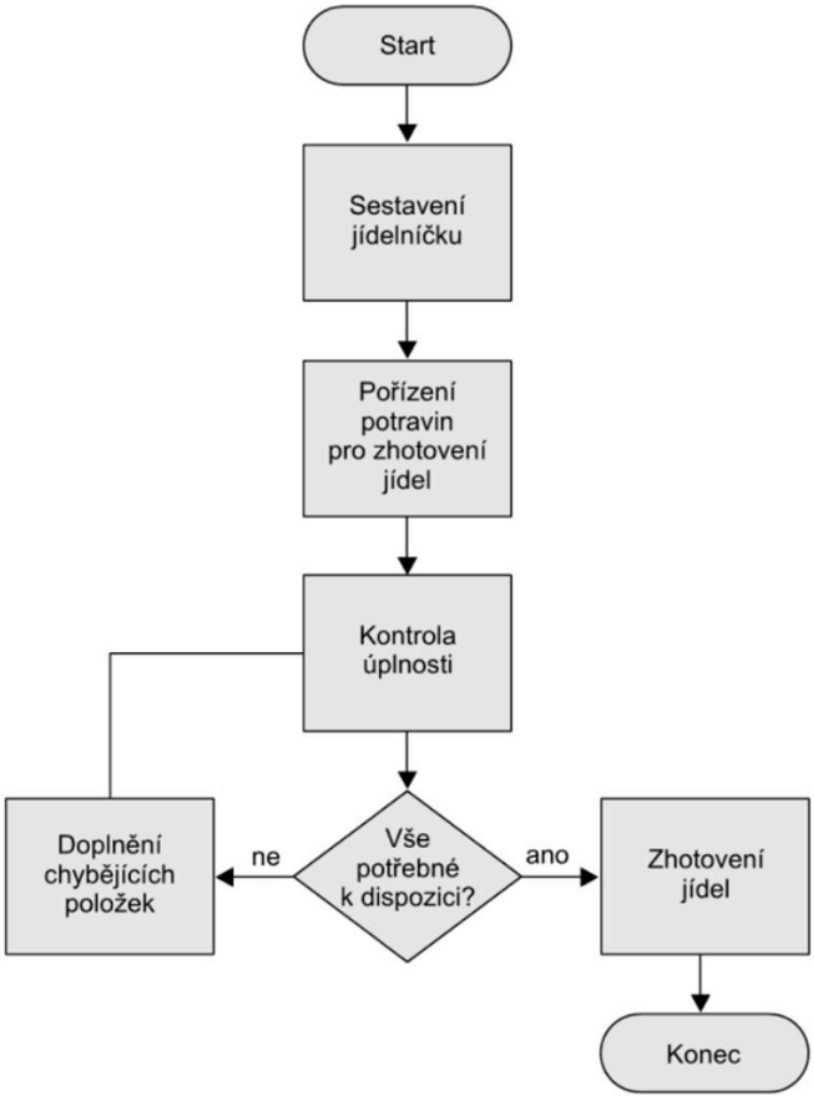
1. *Identifikace oblastí a skupin procesů* – vychází z procesní analýzy a rozděluje podnikové procesy do skupin.
2. *Identifikace procesů dané oblasti a jejich popis* – popis základních charakteristik procesů v rámci jednotlivých skupin.
3. *Rozdělení procesů na subprocesy* – rozdělení procesů do menších logický celků
4. *Popis průběhu subprocesů a informací o všech činnostech* – definování toho, z jakých činností se subprocesy skládají a jaké jsou jejich vstupy a výstupy.
5. *Kontrola konzistence a správnosti procesů* – ověření, zda je vytvořený procesní model funkční a kompatibilní s prostředím v organizaci.

3.4.3 Modelování procesů

Aby byly podnikové procesy snadno pochopitelné a přehledné, jsou graficky modelovány za pomoci diagramů. Nejčastějším zobrazení procesů bývají procesní mapy a BPMN dráhové diagramy (19).

Procesní mapa je přehledový diagram, který má za cíl přehledně vyjádřit souvislosti mezi jednotlivými procesy, či skupinami procesů. Procesní mapy jsou stěžejní pro poskytnutí kontextu všech procesů v organizaci (15).

Diagramy procesních map obvykle neobsahují detailní informace, soustředí se na zobrazení vztahů mezi jednotlivými procesy, vazby subprocesů na základní procesy, a možné smyčky či větvení procesů. Při tvorbě procesních map se postupuje shora dolů, nebo zleva doprava s hlavním tokem zarovnaným v jedné přímce (17). Obrázek 15 zobrazuje příklad procesní mapy:

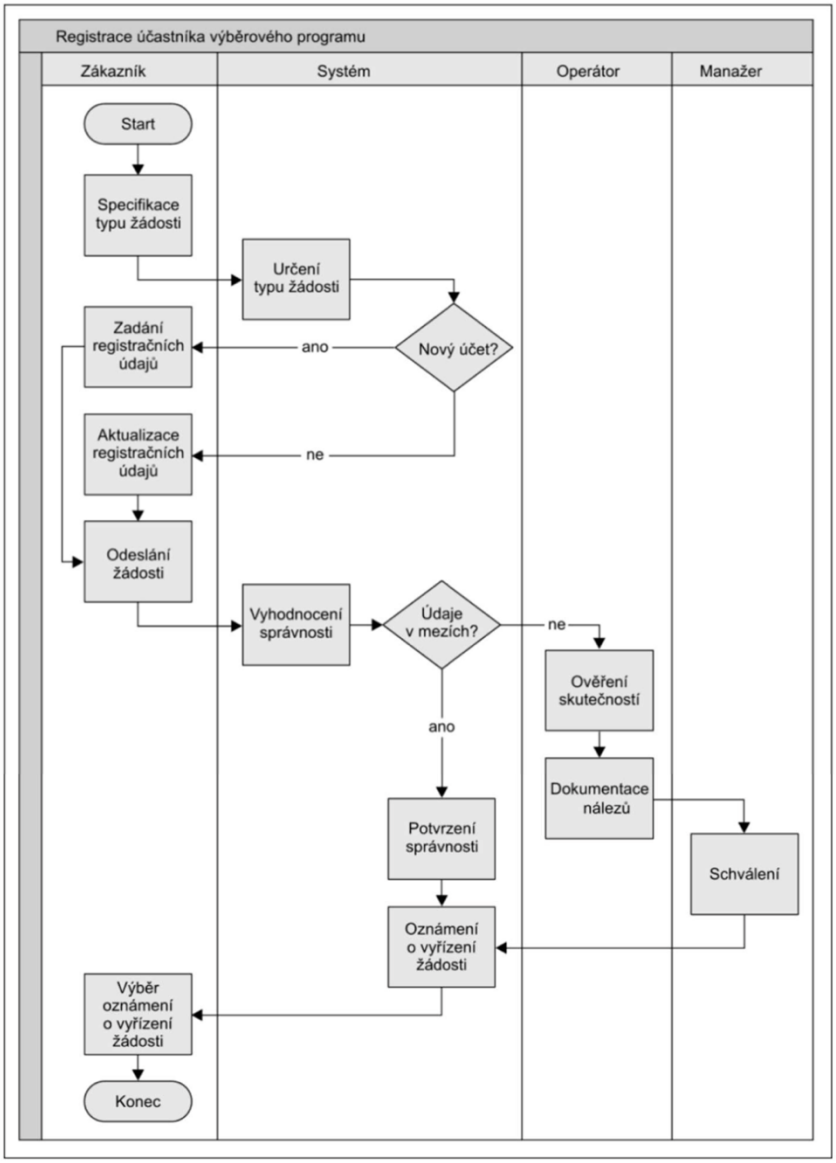


Obrázek 15: Příklad procesní mapy (17)

Pro detailní znázornění procesů, všech jejich činností a návazností se používají dráhové diagramy[3]. Používají se pro svou přehlednost a to, že odpovídají na základní otázky spojené s procesním managementem – *Kdo? Co? Kdy?* Stejně jako procesní mapy i dráhové diagramy jsou tvořeny postupně buď zleva doprava, nebo shora dolů (17).

Hlavním přínosem dráhových diagramů je jejich schopnost znázornit procesy a jejich činnosti i se všemi souvisejícími objekty, jako jsou například zdroje, či dokumenty potřebné pro provedení daných činností, nebo vstupy a výstupy procesu. Obrázek 16 znázorňuje příklad dráhového diagramu.

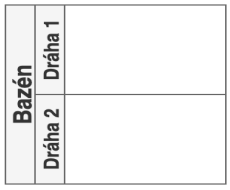
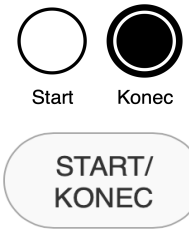
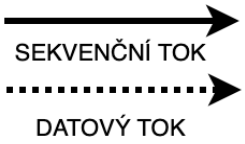
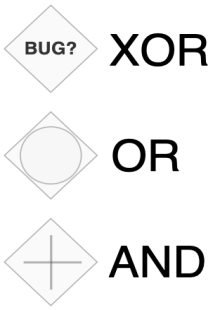
[3] Díky svému vzhledu jsou také často označovány jako plavecké dráhy.



Obrázek 16: Ukázka procesní mapy (16)

Jedním z nejrozšířenějších grafických jazyků pro modelování dráhových diagramů je notace BPMN. Diagram modelovaný pomocí BPMN se skládá z jednotlivých elementů, definovaných grafickými symboly (14). Tabulka 1 zobrazuje základní přehled symbolů používaných v rámci BPMN standardu.

Dráhové diagramy navržené v rámci této diplomové práce jsou modelovány dle notace BPMN a uzpůsobeny zvyklostem procesního modelování v řešené Pojišťovně. Cílem je, aby se nově vypracované diagramy formálně co nejvíce podobaly těm již existujícím a bylo je tak možné snadno implementovat.

Název	Ukázka symbolu	Popis
Bazény a dráhy		Bazén v sobě zahrnuje všechny činnosti procesu a dělí se na jednotlivé dráhy. Každá dráha vyjadřuje účastníka procesu. Samotný proces je následně vyjádřen jako posloupnost činností mezi dráhami.
Události		Událost značí změnu stavu v procesu. Nejčastější jsou počáteční a koncové události. Počáteční událost zahajuje řetězec úloh v rámci procesu na základě konkrétní počáteční události a koncová událost popisuje stav, který ukončením procesu nastal.
Činnosti		Atomické (dále již nedělitelné) aktivity vykonávané v rámci procesu. Graficky jsou znázorněné pomocí obdélníku se zaoblenými rohy a popiskem uvnitř.
Subproces		Subproces je součástí jiného procesu a je graficky oddělen od hlavního procesu.
Toky		Vyjádření posloupnosti pořadí činností v rámci procesu. Sekvenční toky značí šipka směřující od zdrojového objektu k cílovému. Datový tok značí přenos zprávy či dat od jednoho účastníka procesu k druhému.
Brány		Brány jsou používány k řízení interakce sekvenčních toků za účelem větvení a sbíhání procesů. Základními branami jsou: *XOR* – lze vybrat jen jednu větev procesu *OR* – lze vybrat jednu, nebo více větví procesu *AND* – všechny větve procesu od brány probíhají souběžně
Datové objekty		Představují informaci o tom, jaká data slouží jako vstup, či výstup jednotlivých činností. Vazba mezi datovým objektem a činností je vyjádřena pomocí datového toku.

Tabulka 1: Symboly BPMN (vlastní zpracování)

4 Analýza současné situace

Úkolem této kapitoly je představit a zanalyzovat současný stav testingu a procesu softwarové implementace v prostředí IT sekce Pojišťovny. Popis jednotlivých činností a procesů v rámci životního cyklu vývoje softwaru (zahrnující testingové aktivity) je doplněn o grafické znázornění diagramy.

Analýza byla provedena na základě rozhovorů se zaměstnanci Pojišťovny a informací získaných z interní dokumentace týkající se procesu implementace softwaru a testingu.

4.1 Rozsah a obsah pravidelných releasů

Většina softwarové implementace v rámci IT sekce pojišťovny probíhá v rámci takzvaných pravidelných releasů. Pravidelné releasy probíhají třikrát až čtyřikrát do roka a jejich součástí je jak implementaci nových aplikací, tak i implementaci oprav bugů a nových funkčností do již používaných aplikací. Pravidelné releasy v sobě zahrnují implementaci pro všechny země a aplikace podporované IT sekcí. Aktuálně je podporováno necelých deset aplikací v rámci 7 evropských zemí.

Cílem softwarové implementace v rámci pravidelných releasů je poskytnout business části Pojišťovny požadované úpravy, opravy a nové funkčnosti do podporovaných aplikací, a to co nejrychleji, za nejnižší možné náklady a s minimálním rizikem. Spojení implementace změn pro jednotlivé země v rámci jednoho releasu umožňuje spojení testovacích, vývojových, releasových a akceptačních činností, za účelem snížení časové a zdrojové náročnosti implementace softwaru.

4.1.1 Aplikace v releasu

Nejvýznamnější aplikace, používané napříč zeměmi a spravované centrem sdíleného servisu, jsou hlavní součástí pravidelných releasů. Konkrétně se jedná o:

- *Správu likvidace*
- *Správu importů*
- *Portál pro klienty*
- *Bankovní reporting*
- *Bankovní příkazy*

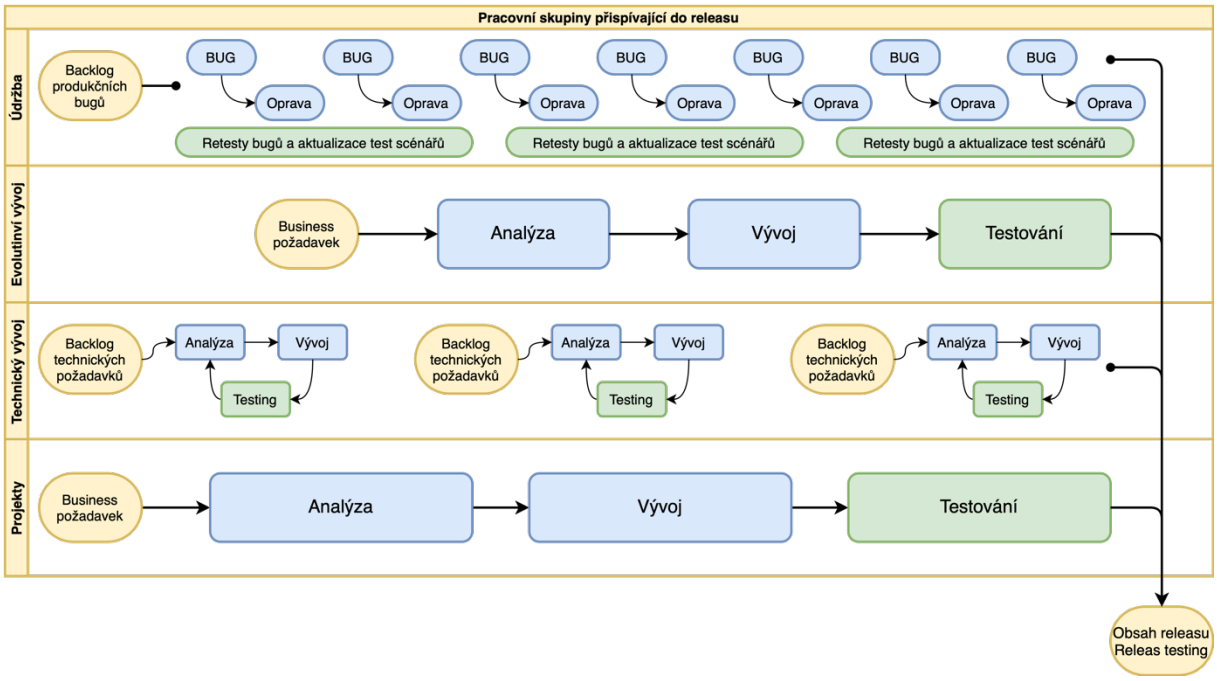
Popis aplikací a jejich významu pro Pojišťovnu je poskytnut v kapitole 2.2.2.

4.1.2 Pracovní skupiny přispívající do releasu

Obsah jednotlivých releasů se skládá z uživatelských požadavků, které jsou následně vyvíjeny v rámci specializovaných pracovních skupin IT sekce. Každá pracovní skupina je řízena samostatně a její výstupy se dostávají do hlavní verze systémů až v okamžiku implementace v releasu. IT sekce pojišťovny se dělí do následujících pracovních skupin:

- *Údržba* – opravy bugů a drobné úpravy v současných funkčnostech aplikací.
- *Evolutivní vývoj* – změny současných, nebo vývoj nových funkčností na základě uživatelských požadavků.
- *Technický vývoj* – kontinuální zlepšování technické stránky softwaru.
- *Projekty* – velké změny funkčních nebo technických parametrů softwaru.
- *Zaváděcí projekty* – projekty spojené se zaváděním současných aplikací a funkčností do nových, dříve nepodporovaných, zemí.

Způsob přístupu k vývoji se napříč pracovními skupinami liší. Některé fungují agilně a některé pracují dle vodopádového modelu. Proces dodávky oprav a nových funkčností do releasů znázorňuje obrázek 17:



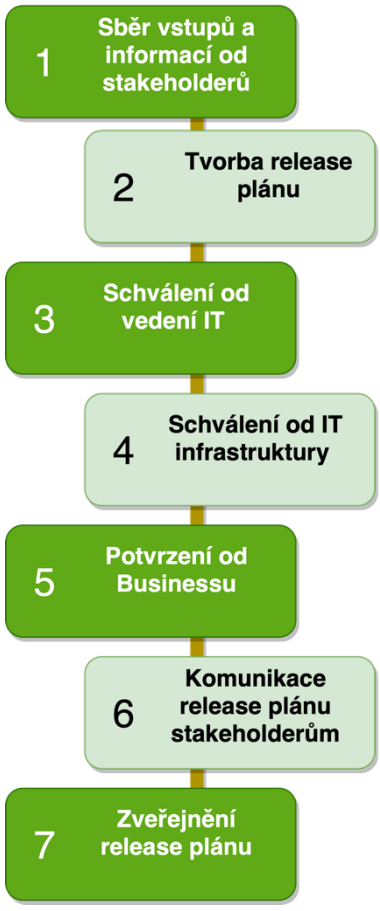
Obrázek 17: Obsah releasu (vlastní zpracování)

Zeleně jsou zvýrazněny aktivity prováděny testingovým oddělením, modře aktivity analýzy a vývoje.

4.2 Plánování releasu

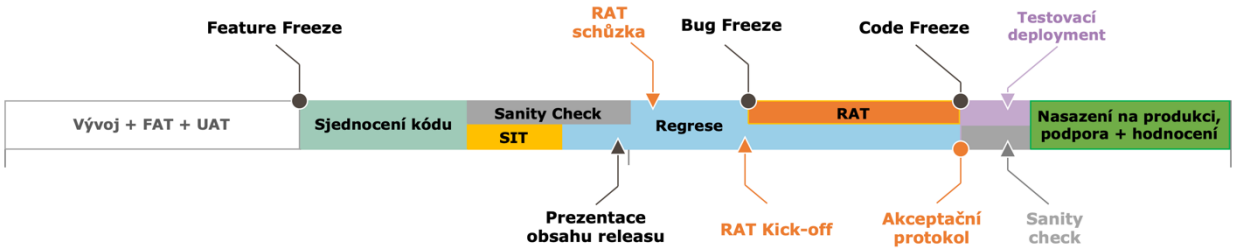
Release plán neboli harmonogram releasu je diagram určující časovou posloupnost základních činností spojených s releasem. Tvorbu plánu řídí release manažer, jehož úkolem je zkoordinovat kapacitu IT a businessu a najít vhodný termín pro implementaci změněného softwaru do produkčního prostředí.

Možnými limitujícími faktory pro plánování implementace jsou například účetní závěrky, audity, kapacita businessu na uživatelské testování, ale také souběh s dalšími projekty, kapacita IT sekce či kapacita IT infrastruktury. Release plán je tvořen na půl roku až rok dopředu a v případě jakýchkoliv změn ze strany IT či business je aktualizován. Proces tvorby release plánu je zobrazen následujícím diagramem:



Obrázek 18: Průběh tvorby release plánu (vlastní zpracování)

Release se jako každý proces skládá z jednotlivých činností. Grafické znázornění release plánu na obrázku 19 zachycuje návaznosti jednotlivých činností, včetně významných milníků procesu. Návaznosti, podmínky a relativní časové zařazení činností vůči termínu deploymentu na produkci zobrazuje tabulka 2.



Obrázek 19: Harmonogram releasu (vlastní zpracování)

Role zapojené do releasu jsou popsány v kapitole 4.2.1 a jednotlivé činnosti jsou detailně rozvedeny v kapitole 4.2.2.

ID	Činnost	Závislost	Podmínky výskytu	Do releasu
1	Vývoj			-
2	FAT	1	Dokončený vývoj	-
3	UAT	1;2	Dokončený vývoj a FAT	-
4	Feature Freeze	1;2;3	Dokončený vývoj	11 týdnů
5	Sjednocení kódu	4	Dokončený vývoj	10 týdnů
6	Sanity Check	5	Ihned po spojení větví kódu	8 týdnů
7	SIT	6	Úspěšný Sanity Check	8 týdnů
8	Regrese	6;7	Úspěšný Sanity Check	8 týdnů
9	Prezentace obsahu releasu od analytiků testingu a businessu	4	Zafixovaný obsah releasu	7 týdnů
10	RAT schůzka	9	Co nejdříve po prezentaci	6 týdnů
11	RAT Kick-off	10	Poslední pátek před RAT	5 týdnů
12	Bug Freeze	6;7;8	V půlce regrese	4 týdny
13	RAT	8;11	V půlce regrese	4 týdny
14	Code Freeze	8;13	Po skončení RAT a spravení kritických bugů	1 týden
15	Akceptační protokol	14	Úspěšné RAT	1 týden
16	Testovací deployment	15	Podepsaný akceptační protokol	1 týden
17	Sanity Check	16	Ihned po testovacím deploymentu	1 týden
18	Nasazení na produkci	16;17	Úspěšný Sanity Check	0
19	Podpora po releasu	18	Nasazení na produkci	2-4 týdny po
20	Vyhodnocení releasu	19	Nasazení na produkci	2 týdny po

Tabulka 2: Činnosti v rámci releasu (vlastní zpracování)

Release plán je v praxi vypracován vždy s konkrétními daty. Pro účely této diplomové práce byla provedena analýza releasů v období 2020 až 2021 a z ní stanoveny relativní doby výskytu jednotlivých činností a milníků v závislosti k začátku nasazování na produkční prostředí.

4.2.1 Role zapojené v průběhu releasu

Tato kapitola se zabývá popisem rolí zapojených do implementace softwaru a jejich úloze v rámci procesu. Následuje výčet rolí.

Release manažer

- Monitoruje a koordinuje release napříč všemi fázemi procesu implementace softwaru
- Zodpovídá za tvorbu a aktualizaci release plánu
- Kontroluje dodržování Feature Freeze a urguje zapojené pracovní skupiny ohledně dodávek funkcionalit do releasu
- Zajišťuje proces akceptace implementovaného releasu od zákazníků
- Organizuje a vyhodnocuje retrospektivu po releasu
- Komunikuje se zúčastněnými stranami ohledně stavu releasu

Test leader

- Tvoří test strategii releasu/projektu a test plán odpovídající test scope
- Definuje požadavky na testovací prostředí a alokaci zdrojů pro testování releasu
- Řídí přípravu testovacího prostředí a testovacích dat pro exekuci testování
- Zodpovídá za tvorbu, aktualizaci a vyhodnocování test dashboardu s výsledky testování. Výsledy testování komunikuje všem relevantním osobám.
- Vyhodnocuje závažnost chyb nalezených testery a případně urguje jejich opravu
- Organizuje a vyhodnocuje retrospektivu testování uplynulých releasů, řídí aktualizaci testovací metodiky a artefaktů

RAT koordinátor

- Společně s business analytiky definuje test scope uživatelských releasových akceptačních testů
- Tvoří a udržuje test plán pro fázi releasových akceptačních testů
- Koordinuje přípravu testovacího prostředí a testovacích dat pro RAT
- Slouží jako kontaktní a podpůrná osoba pro uživatele zapojené do RAT
- Monitoruje, vyhodnocuje a reportuje stav RAT
- Sbírá, vyhodnocuje a komunikuje výsledky zpětné vazby k implementaci releasu a průběhu RAT od uživatelů

Business analytik

- Sbírá a formalizuje požadavky od businessu na nové funkcionality do tzv. business požadavků
- Provádí UAT testování nových funkcionalit
- Organizuje koncové uživatele během RAT a podílí se na testování

IT analytik

- Analytik na straně IT, hledá řešení pro business požadavky
- Sepisuje analýzu, poskytuje odhady pracnosti pro nové funkcionality
- Podílí se na SIT

Release inženýr

- Spravuje produkční a testovací prostředí
- Nasazuje nový kód/funkcionality na dané prostředí
- Zajišťuje technickou stránku releasu ve spolupráci s lokální IT podporou

Vývojář

- Vyvíjí nové funkcionality
- Opravuje nalezené bugy

Tester

- Provádí test analýzu nových funkcionalit
- Podílí se na přípravě testovacích scénářů pro nové funkcionality a údržbě regresních testovacích scénářů
- Identifikuje a připravuje testovací data nutné pro exekuci testovacích scénářů
- Provádí testování Sanity Check, FAT, SIT a regresí
- Zaznamenává nalezené bugy a re-testuje jejich opravy
- Někteří testeři se specializují na automatizaci testů, jejich exekuci a údržbu

Role Test leader a RAT koordinátora bývá často spojena v jedné osobě, která řídí jak testování z IT strany, taky i ze strany uživatelů.

4.2.2 Popis činností v rámci procesu softwarové implementace

Release plán na obrázku 19 a tabulce 2 představuje posloupnosti a závislosti v rámci procesu implementace softwaru. Tato podkapitola detailně rozepisuje obsah jednotlivých činností a podstatu releasových milníků.

- *Vývoj + FAT + UAT* – první část procesu implementace, kdy jednotlivé pracovní skupiny vyvíjejí nové aplikace/funkčnosti na základě business požadavků. Nově vyvinuté funkcionality jsou nejdříve otestovány testerem v rámci FAT. Součástí FAT je tvorba test analýzy. Po dokončení FAT je nová funkcionalita předána business analytikům k uživatelskému akceptačnímu testování. UAT provádí vždy analytik z té země, odkud požadavek vzešel. UAT je prováděno formou explorativního testování.

- *Feature Freeze* – předem stanovený termín, ke kterému musí všechny pracovní skupiny dodat vyvinutou, otestovanou a business analytikem schválenou funkcionalitu. Jestliže není jedna z těchto podmínek splněna, funkcionalita je odstraněna z aktuálního releasu a posunuta do dalšího.

- *Sjednocení kódu* – jakmile je po Feature Freeze zafixován obsah releasu, je třeba spojit kód všech změn pro každou aplikaci do jediné verze. Tento proces má na starosti zástupce vývoje a release inženýr. Jakmile je kód spojen, release inženýr ho nasadí na testovací prostředí.

- *Sanity Check* – v okamžiku, kdy release inženýr nasadí na testovací prostředí verzi aplikací se všemi novými změnami, testing začíná se sanity checky, aby ověřil, zda je nová verze použitelná pro další testování. V případě nasazení nové verze s opravami nalezených chyb je potřeba provést sanity check znovu.

- *SIT* – jestliže v rámci sanity check nebyly identifikovány žádné závažné bugy, přechází testing k systémově integračním testům.

- *Regrese* – po SIT následuje regresní testování, které provádí také testing.

- *Prezentace obsahu releasu od analytiků testingu a businessu* – jakmile je znám obsah releasu, analytici zorganizují prezentaci o všech novinkách a změnách v systémech. Změny jsou prezentovány jak testerům, aby věděli, jaké zásahy do současného stavu aplikací implementace způsobí, tak i businessu, aby business zástupci jednotlivých zemí zjistili, jaké změny žádané ostatními zeměmi půjdou do releasu.

- *RAT schůzka* – schůzka RAT koordinátora s business analytiky zemí zasažených releasem. Cílem schůzky je definovat test scope RAT a vyjasnit případné otázky ze strany businessu.

- *RAT Kick-Off* – schůzka RAT koordinátora s business analytiky těsně před začátkem RAT. RAT koordinátor zde prezentuje informace o aktuální stavu testovacího prostředí, jednotlivých aplikacích a známých aktivních chybách.
- *Bug Freeze* – okamžik, od kterého se již nespravují nekritické bugy v současné verzi softwaru, ale jejich oprava se odsouvá do dalšího releasu.
- *RAT* – testy prováděné koncovými uživateli a business analytiky.
- *Code Freeze* – okamžik zafixování kódu implementovaného softwaru. Vzniká verze, kandidát na nasazení na produkci. Od tohoto okamžiku by už nemělo docházet k žádným změnám v kódu.
- *Akceptační protokol* – v závislosti na výsledcích RAT, podepisují zástupci jednotlivých zemí akceptační protokol. Protokol potvrzuje, že IT může v daném termínu nasadit testovaný software na produkční prostředí. V případě známých chyb může dojít k tvorbě SLA[4].
- *Testovací deployment* – schválená finální verze kódu je nejdříve nasazena na takzvané před-produkční prostředí. Jedná se o prostředí s konfigurací co nejpodobnější té na produkci. Testovací prostředí má většinou anonymizovaná klientská data, před-produkční nikoliv.
- *Sanity Check* – sanity check na před-produkčním prostředí. Finální kontrola systémů před samotným nasazením na produkci. Před-produkční prostředí svým nastavení a daty (neanonymizovaná data, fungující SMS a email brána, produkční konfigurace atd.) odpovídá, co nejvíce je možné, skutečné produkci.
- *Nasazení na produkci* – jestliže nejsou v sanity check na před-produkčním prostředí nalezeny žádné bugy, je možný deployment na produkci.
- *Podpora po releasu*– určitou dobu po deploymentu na produkci má IT navýšenou kapacitu pro podporu produkce, pro případ, že by se na produkčních systémech objevila chyba, kterou nezachytilo testování.
- *Vyhodnocení releasu* – zhodnocení uplynulého releasu a získávání zpětné vazby od businessu. Na základě připomínek a nálezů z retrospektivy se upravuje další release.

[4] SLA (Service Level Agreement) je závaznou dohodou mezi poskytovatelem služby a zákazníkem, která identifikuje požadované služby, úroveň těchto služeb a dobu jejich dodání (20). Například do kdy musí poskytovatel dodat opravu chyby.

4.2.3 Metriky softwarové implementace

V průběhu i po dokončení procesu implementace změn v softwaru je sledováno několik metrik vypovídajících o stavu testovacího prostředí, kvalitě změn v kódu, stavu aplikacích jako takových, a o výkonosti týmů. Následuje výčet releasových metrik používaných v Pojišťovně:

- *Průměrná doba implementace požadavku* – jak dlouho od sepsání původního business požadavku trvá, než je funkcionalita implementována?
- *Průměrná doba vyřešení bugu* – jaká je průměrná doba od založení bugu, po jeho spravení a nasazení opravy?
- *Průměrný počet produkčních bugů po nasazení* – kolik bugů bylo odhaleno v nové verzi softwaru po implementaci do produkčního prostředí?

Mimo metriky, které Pojišťovna používá byly pro tuto diplomovou práci stanoveny ještě následující doplňující metriky:

- *Průměrná doba vyřešení bugů severity 1* – jak dlouho trvá vyřešit závažné bugy se severitou 1? Škála severity používaná v Pojišťovně je uvedena na další straně.
- *Průměrný počet bugů od sjednocení kódu* – kolik bugů bylo hlášeno během releasového cyklu dané implementace? Bugy jsou počítány od okamžiku sjednocení všech větví kódu až po nasazení na produkci.
- *Průměrná severita produkčních bugů po nasazení* – jaká byla průměrná závažnost bugů nalezených po nasazení na produkčním prostředí?

V rámci analýzy současného stavu procesu softwarové implementace v Pojišťovně byly vyhodnoceny releasy posledních dvou let. Napříč tímto období nabývaly sledované metriky těchto hodnot:

Metrika	Průměrná hodnota	Jednotka
Průměrná doba implementace požadavku	218	Dny
Průměrná doba vyřešení bugu	63	Dny
Průměrný počet produkčních bugů po nasazení	41	Počet
Průměrná doba vyřešení bugů severity 1	8,5	Dny
Průměrný počet bugů od sjednocení kódu	115	Počet
Průměrná severita produkčních bugů po nasazení	1,56	N/A

Tabulka 3: Metriky softwarové implementace v Pojišťovně (vlastní zpracování)

Severita neboli závažnost bugů je v Pojišťovně měřena na číselné stupnici od 1 do 4. Jednotlivé stupně závažnosti mají následující význam:

1. *Kritická* – Nutno opravit co nejdříve. Chyba znemožňující fungování celého systému, nebo jeho součásti. Chybu nelze obejít a dosáhnout požadovaného výsledku.
2. *Vysoká – Vážná chyba v systému, nebo jeho součásti, ke které ovšem existuje možnost obejití k dosažení požadovaných výsledků.*
3. *Střední – Bug, který způsobuje nesprávné, neúplné nebo nekonzistentní výsledky systému.*
4. *Nízká – Drobná nebo kosmetická chyba, kterou lze snadno obejít.*

Kromě severity je u bugů zaznamenávána také priorita. Priorita může nabývat hodnot 1 až 4, přičemž 1 znamená nejvyšší prioritu. Kombinace hodnot priority a severity pomáhá analytikům a vývojářům určit pořadí v jakém se věnují opravě bugů.

4.3 Zapojení testingu v průběhu releasu

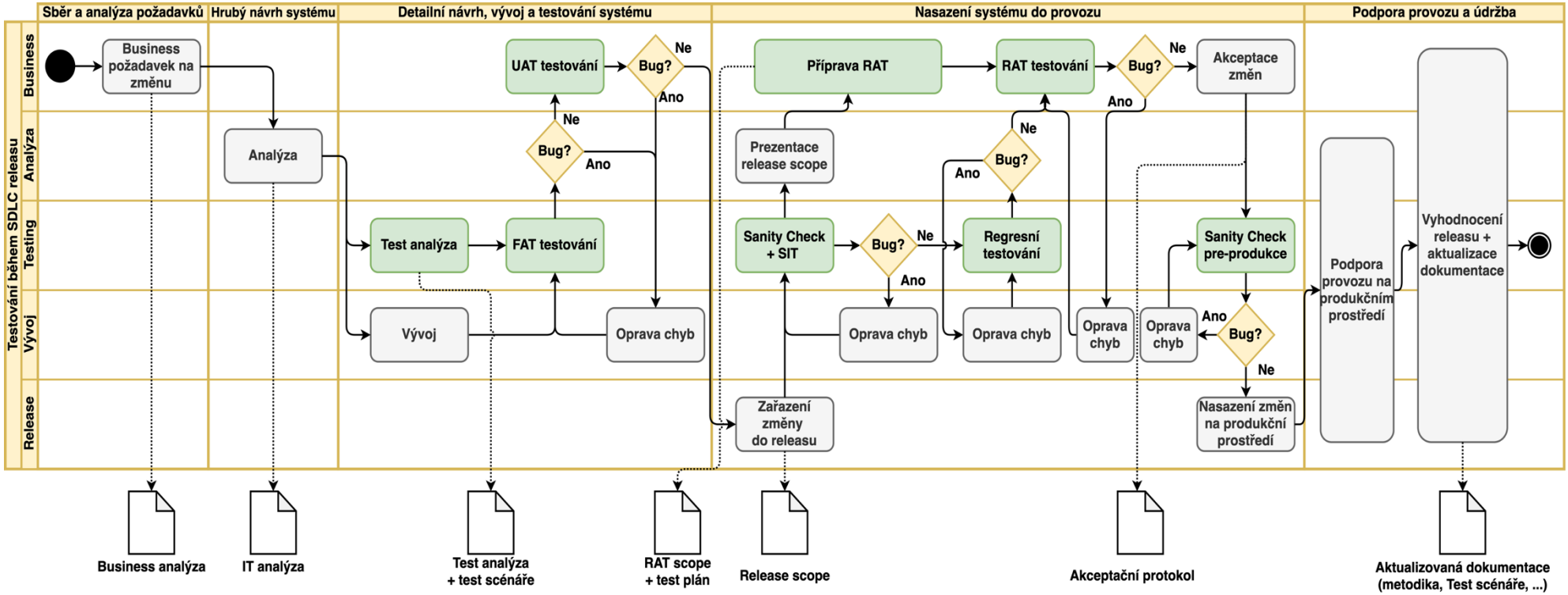
Testingové oddělení IT sekce Pojišťovny je zapojené do celého životního cyklu vývoje softwaru. Obrázek 20 na následující straně zobrazuje činnosti testingu spojené s releasem dané do souvislosti s pěti fázemi SDLC představené v kapitole 3.1.1.

Jednotlivé dráhy diagramu představují oddělení zapojené do procesu implementace softwaru. Business zde pro vyšší přehlednost diagramu představuje jak zadavatele požadavků, koncového uživatele, tak i business analytiky. Činnosti vykonávané přímo testingem, nebo za jeho spolupráce/koordinace (UAT a RAT testování) jsou označeny zeleně.

Součástí činnosti test analýza je také tvorba nových testovacích scénářů zaměřených na kontrolu nové funkcionality a jejích dopadů na současný systém. Nově vytvořené test scénáře jsou následně prováděny během FAT testování. Jestliže FAT neodhalí žádné chyby, funkcionalita je předána k uživatelskému UAT testování business analytikovi, který založil daný požadavek na změnu. V případě nalezení chyb během FAT, nebo UAT je funkcionalita vrácena vývoji k opravě. Následně jsou všechny funkcionality, které byly vyvinuty a otestovány do termínu Feature Freeze, zařazeny do aktuálního releasu a jejich kód je spojen do jedné větve.

Nově sjednocený a nasazený kód je zkontrolován testingem pomocí sanity check a následně SIT. Ve stejnou dobu jako sanity check probíhá i prezentace obsahu releasu od analytiků pro business. Na základě této prezentace pak business ve spolupráci s RAT koordinátorem stanoví rozsah a obsah uživatelského akceptačního testování RAT. RAT začíná obvykle v půlce regresního testování.

Po dokončení regresního i RAT testování a opravení všech zásadních bugů dochází k podepsání akceptačního protokolu. V rámci protokolu mohou být stanovena případná SLA. Následuje nasazení na před-produkční prostředí a sanity check. V případě, že sanity check neodhalí žádné chyby, je release nasazen na produkční prostředí a následuje fáze podpory provozu a údržby, během které dochází k řešení SLA, vyhodnocení průběhu releasu a aktualizaci dokumentace, testovacích scénářů a dalších releasových/ testingových artefaktů.



Obrázek 20: Testovací aktivity v průběhu SDLC releasu (vlastní zpracování)

4.4 Shrnutí analýzy stávající situace

Po provedení analýzy současného stavu procesu implementace softwaru v Pojišťovně byly autorem práce identifikovány následující nedostatky:

- Obsah releasu je komunikován až těsně před zahájením RAT. Testing se dozvídá přesný obsah releasu až poté, co začíná se sanity checky a SIT testováním. To vede k dodatečným úpravám v releasovém test plánu během testování a nutnosti pořádání RAT plánovacích schůzek na poslední chvíli před zahájením samotného uživatelského testování.
- V release plánech se nepočítá s automatickým testováním, i když IT sekce vyvíjí značné úsilí automatizovat sanity check a regresní testovací scénáře.
- Dlouhá doba od požadavku po implementaci změny. Opravy bugů, a to i těch se severitou jedna, také trvají příliš dlouho.
- Neexistuje role UAT koordinátora, výsledky UAT se komunikují pouze na úrovni jednotlivých požadavků. UAT testy provádí vždy jen business analytik ze země, odkud požadavek vzešel, a to jen na testovacím prostředí dané země.
- Neexistují jasně stanovená kritéria pro přechody mezi jednotlivými fázemi testování.
- Samotný průběh releasu je řízen striktně vodopádovým přístupem k vývoji. Nebyl by proces implementace softwaru efektivnější při použití agilních metod?

5 Návrh řešení

Obsahem této kapitoly je vlastní návrh inovací procesu implementace softwaru v rámci IT sekce Pojišťovny. Inovace jsou vzhledem k tématu práce zaměřeny primárně na testovacích činností spojených s releasy. Návrh je založen na teoretických principech představených ve třetí kapitole této práce a vychází z analýzy stávajícího stavu z předchozí kapitoly.

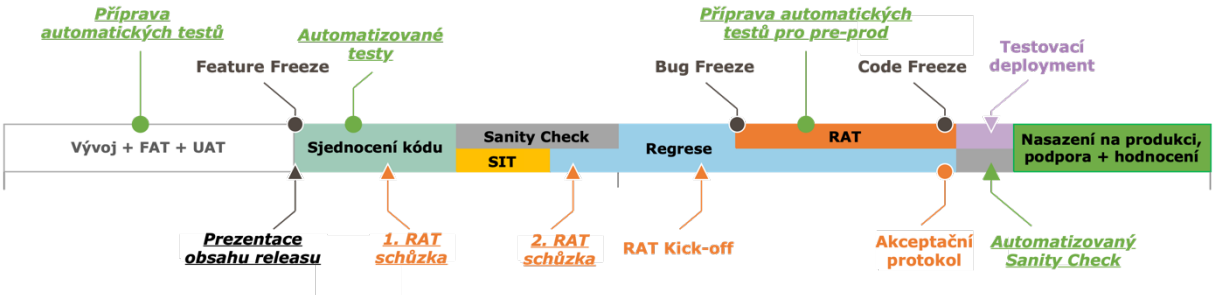
Návrh inovace procesu softwarové implementace se zaměřuje na následující 7 témat:

- Harmonogram releasu
- Použití automatizovaných testů v releasu
- Kvalita dodávaného softwaru
- Koordinace uživatelského testování
- Kritéria pro přechod mezi jednotlivými fázemi testování releasu
- Testovací aktivity v průběhu releasu
- Agilní release

Šestá kapitola následně předkládá doporučení pro zavádění navržených inovací do aktuálního procesu implementace softwaru v Pojišťovně.

5.1 Harmonogram releasu

První navrhovanou změnou současného procesu softwarové implementace v Pojišťovně je úprava harmonogramu releasu. Změny oproti původnímu harmonogramu na obrázku 19 jsou zobrazeny pomocí časové osy na obrázku 21 a v tabulce číslo 4, která poskytuje popis jednotlivých změn v současném release plánu. Odůvodnění těchto změn je poskytnuto pod tabulkou.



Obrázek 21: Upravený harmonogram releasu (vlastní zpracování)

Všechny změněné a nově přidané aktivity jsou v harmonogramu vyznačeny kurzívou a podtrženým textem.

ID	Činnost	Závislost	Podmínky výskytu	Do releasu
1	Vývoj			-
2	FAT	1	Dokončený vývoj	-
3	*UAT*	*1;2*	*Dokončený vývoj a FAT*	*-*
4	*Příprava automatických testů*		*Dostupné testovací prostředí*	
5	Feature Freeze	1;2;3	Dokončený vývoj	11 týdnů
6	*Prezentace obsahu releasu od analytiků testingu a businessu*	*5*	*Zafixovaný obsah releasu*	*10 týdnů*
7	Sjednocení kódu	5	Dokončený vývoj	10 týdnů
8	*Automatizované testy*	*4;5*	*Zafixovaný obsah releasu a připravené aut. testy*	*10 týdnů*
9	*1. RAT schůzka*	*6*	*Měsíc před zahájením RAT*	*9 týdnů*
10	Sanity Check	7	Ihned po spojení větví kódu	8 týdnů
11	SIT	10	Úspěšný Sanity Check	8 týdnů
12	*2. RAT schůzka*	*9*	*2 týdny po 1. RAT schůzce*	*7 týdnů*
13	Regrese	10;11	Úspěšný Sanity Check	7 týdnů
14	RAT Kick-off	12	Poslední pátek před RAT	5 týdnů
15	Bug Freeze	10;11;13	V půlce regrese	4 týdny
16	RAT	13;14	V půlce regrese	4 týdny
17	*Příprava automatických testů na před-produkčním prostředí*	*4;8;13*	*Automatické testy vyladěné v předchozích fázích testování*	*4 týdny*
18	Code Freeze	13;16	Po skončení RAT a spravení kritických bugů	1 týden
19	Akceptační protokol	16	Úspěšné RAT	1 týden
20	Testovací deployment	19	Podepsaný akceptační protokol	1 týden
21	*Automatizovaný Sanity Check*	*17;20*	*Ihned po testovacím deploymentu*	*1 týden*
22	Nasazení na produkci	19;21	Úspěšný Sanity Check	0
23	Podpora po releasu	22	Nasazení na produkci	2-4 týdny po
24	Vyhodnocení releasu	22	Nasazení na produkci	2 týdny po

Tabulka 4: Aktualizované činnosti v rámci releasu (vlastní zpracování)

Nově přidané, či změněné činnosti jsou v tabulce vyznačeny kurzívou a podtrženým textem. Ostatní činnosti zůstávají nezměněny.

Následuje popis změn a nových činností:

- *UAT – změnami* v průběhu UAT se zabývá samostatná kapitola 5.4.
- *Příprava automatických testů* – úpravy stávajících automatizovaných testovacích scénářů, tak aby fungovaly i pro nové funkcionality, které budou představeny v nadcházejícím releasu. Případně jsou také vytvořeny nové automatické testy. Zařazení automatizovaných testů v softwarové implementaci podrobně rozebírá kapitola 5.2.
- *Prezentace obsahu releasu od analytiků testingu a businessu* – prezentace obsahu releasu byla v harmonogramu přesunuta za feature freeze. Tento posun umožní delší a kvalitnější přípravy na RAT fázi a upřesní testerům rozsah jejich testování před zahájením samotného testování.
- *Automatizované testy* – jakmile je kód sjednocený, je potřeba začít spouštět automatizované testy. Ty je nejprve třeba do-upravit tak, aby správně fungovaly nad novou, sjednocenou verzí kódu. Poté co jsou testy funkční, jsou používány pro automatizovaný sanity check.
- *1. RAT schůzka* – RAT schůzka byla rozdělena na dvě navazující schůzky. Během první představí RAT koordinátor business analytikům rozsah testovacích aktivit na releasu a společně stanoví scope RAT.
- *2. RAT schůzka* – druhá schůzka RAT koordinátora s business analytiky, na které dochází k finalizaci RAT test plánu a vyjasnění případných otázek ze strany businessu. Rozdělení plánování RAT na dvě samostatné schůzky v dostatečném předstihu před samotným zahájením RAT dává více prostoru pro kvalitní přípravu.
- *Příprava automatických testů na před-produkčním prostředí* – konfigurace testovacího a před-produkčního prostředí je často odlišná. Proto je třeba zkontrolovat, jestli automatizované testy lze spustit na před-produkčním prostředí a případně je upravit.
- *Automatizovaný Sanity Check* – spuštění automatizovaných testovacích scénářů na před-produkčním prostředí. Delegování sanity check z manuálních testerů na automatické testy nejen že urychlí testování, ale také odstraní nutnost žádat v předstihu o výjimku přístupu testerů na prostředí s neanonymizovanými daty, kam běžně přístup nemají.

5.2 Automatizace testování

Přestože testing aktivně pracuje na tvorbě automatizovaných testovacích scénářů nejsou tyto testy ještě používány v rámci testování releasů. Aktuálně má testing automatizovaný běžný sanity check pro rychlou kontrolu stavu testovacího prostředí, a část regresní testovací sady. Použití automatických testů šetří práci manuálním testerům[5] a významně zkrátí dobu potřebnou na testování. Navíc lze naplánovat jejich spouštění i mimo pracovní dobu, například přes noc. Tester pak jen ráno zkontroluje výsledky testů a případně nahlásí bugy.

Při implementaci automatizace testování do release plánu je potřeba počítat i s časem na přípravu testů. Nové funkcionality představené v releasu mohou změnit chování systému a staré automatizované testy by tak už nemusely fungovat. Proto je důležité začít s přípravou na spouštění těchto testů ihned, jakmile je daná funkcionalita zařazena do cílového releasu.

Další velkou výhodou použití automatizovaných testů je zkrácení schvalovacího procesu pro přístup k testování na před-produkční prostředí, kde se nachází neanonymizovaná data. Vzhledem k tomu, že samotné testování provádí automat a tester jen kontroluje report o výsledku testu, stačí aby přístup k před-produkčnímu prostředí měl jen software vykonávající testy. Nalezené chyby se předávají na opravu analytikům a vývojářům, kteří mají přístup k neanonymizovaným datům z důvodu podpory produkce.

Činnosti spojené s přípravou a spouštění automatizovaných testovacích scénářů, jsou součástí návrhu inovovaného harmonogramu releasu v kapitole 5.1.

5.3 Kvalita dodávky

Výsledky metrik sledovaných v průběhu releasu (kapitola 4.2.3), poukazují na dvě oblasti, které potřebují inovaci. Cílem této kapitoly je přednést návrh této inovace.

První kritickou oblastí je průměrná doba řešení bugů. Průměr 8,5 dne na opravu chyby se severitou 1 je velkým rizikem pro celou implementaci. Severita 1 znamená, že bug blokuje určitou funkcionalitu, nebo rovnou celý systém. Dokud není chyba opravena, není možné pokračovat v testování zasažené oblasti. Jestliže běžné release testování trvá deset týdnů a oprava bugů se severitou 1 trvá 8,5 dne, znamená to, že určitá funkcionalita/aplikace je blokovaná 12 procent celkového trvání releasového testování.

Stejně tak celková průměrná doba řešení bugů nabývá vysokých hodnot. Průměr 63 dní znamená, že se při frekvenci tří až čtyř releasů ročně (release každých 120 až 90 dní), opravy některých bugů dostanou až do následujícího, nebo ještě dalšího releasu po tom, ve kterém byl bug nahlášen.

[5] Tester, který vykonává testovací scénáře sám, bez použití automatizačních softwarů, tedy manuálně.

Ke zlepšení současného stavu je třeba lepšího řízení priorit a kapacit. Možným řešením je navýšení počtu analytiků a vývojářů na opravování chyb. Nábor nových lidí ovšem souvisí se zvýšením výdajů na zaměstnance a vzhledem k času potřebnému pro nalezení, přijmutí a zácvik vhodných kandidátů, se zlepšení projeví až v horizontu několika měsíců. Rychlým řešením je změna v současném kapacitním plánu a přeřazení vývojářů a analytiků z projektů a dalších pracovních skupin na opravu bugů. Tato změna sice prodlouží dobu dodávky nových funkcionalit, ale prioritou číslo jedna by vždy měl být stav produkčního prostředí. Jak již bylo představeno v kapitole 3.2.1, čím později je chyba identifikována a řešena, tím nákladnější je pak její oprava (12).

Druhou kritickou oblastí, kterou indikují měřené metriky, je vysoký počet chyb nalezených v produkčním prostředí po nasazení nových verzí systémů. Chyby v produkci mohou pak v závislosti na své severitě ohrozit finanční výsledky Pojišťovny, či její reputaci u zákazníků. Z toho důvodu je třeba počet bugů po nasazení do produkce snížit na minimum.

Část řešení je již zmiňované zkrácení doby pro vyřešení nalezených bugů, zvláště těch s vysokou severitou. Čím dříve budou blokované oblasti systému opraveny, tím dříve může testing/uživatelé pokračovat v testování. Dalším krokem ke zlepšení je důsledné testování napříč celým procesem implementace. Posunutím termínu prezentace obsahu releasu hned po Feature Freeze a striktním dodržování samotného Feature Freeze docílíme transparentnosti obsahu releasu a prodloužíme čas na přípravu a exekuci testování. Návrhem zlepšení uživatelského testování se zabývá další kapitola.

Aby byla změna ve sledovaných metrikách měřitelná, je potřeba metriky pravidelně vyhodnocovat a stanovit si cílové hodnoty, ke kterým by se jejich výsledky měly v dalších releasech dostat. Cílové hodnoty metriky musí být stanoveny s ohledem na velikost plánovaného releasu.

5.4 Koordinace uživatelského testování

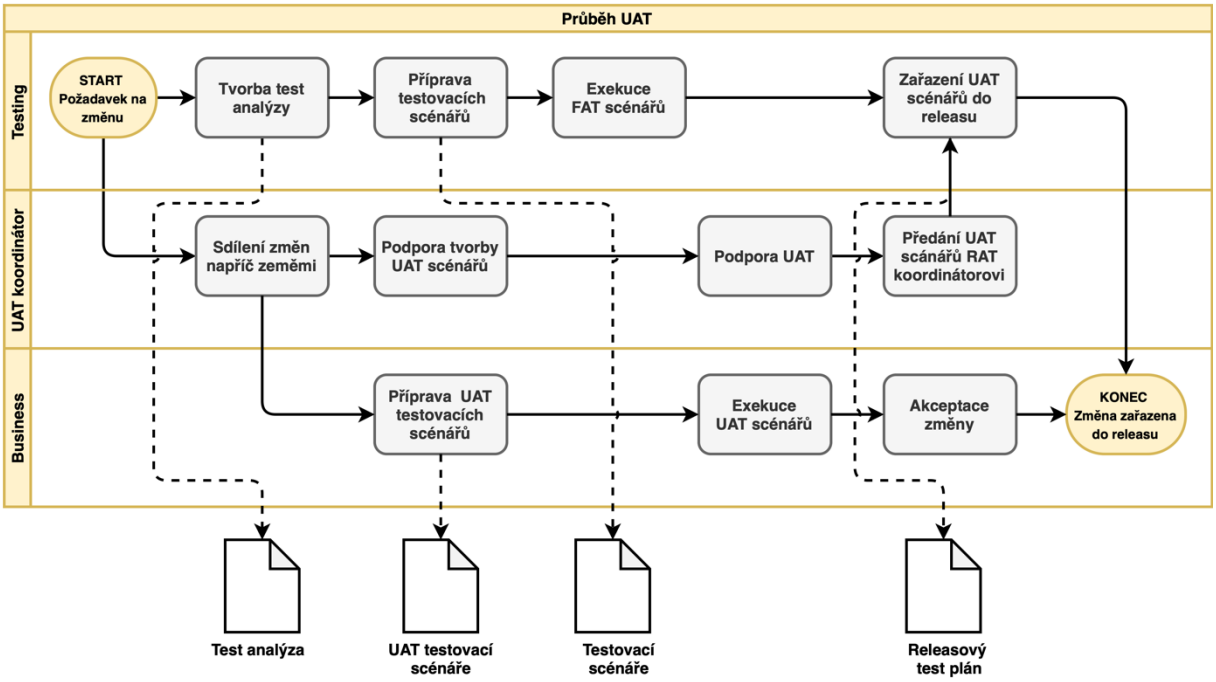
V současnosti neexistuje v Pojišťovně role UAT koordinátora. Nově vyvinuté funkcionality jsou po úspěšném FAT testování předány na test pouze business analytikovi ze země, která vznesla požadavek. Business ostatních zemí se o většině funkcionalit z ostatních zemí dozvídá až v okamžiku prezentace obsahu releasu. Tento přístup zpomaluje případnou implementaci a úpravy nové funkcionality pro ostatní země.

Samotné UAT testování probíhá formou explorativních testů bez formálního zápisu uživatelských testovacích scénářů. Neexistence uživatelských testovacích scénářů pak vede k tomu, že při sestavování RAT test plánu jsou používány scénáře psané testingovým oddělením. Velkým přínosem uživatelského testování je právě perspektiva, kterou business uživatel získá praxí na produkci. Jakmile mu jsou ovšem nabídnuty scénáře s jasně definovanými kroky, které jsou psány z pohledu IT, je přínos potlačen a dochází pak de facto ke dvojímu testování toho samého.

Z těchto důvodů je třeba vytvořit novou roli UAT koordinátora. Tato role bude zodpovídat za sdílení nově vyvíjených funkčností napříč všemi zeměmi centra sdíleného servisu, aby všechny země věděly, co je aktuálně ve vývoji. Další úlohou UAT koordinátora bude řízení

tvorby uživatelských testovacích scénářů k novým funkcionalitám. Samotné scénáře budou tvořeny a udržovány zástupci businessu.

V průběhu tvorby test plánu pro release, ve kterém bude funkcionalita představena, předá UAT koordinátor nové testovací scénáře RAT koordinátorovi. Ten scénáře projde a vybrané použije jako součást RAT test plánu. Je možné, že obě role UAT a RAT koordinátora budou vykonávány jedním zaměstnancem. Činnosti vykonávané UAT koordinátorem v průběhu releasu jsou znázorněny následujícím diagramem:



Obrázek 22: Průběh UAT (vlastní zpracování)

Činnosti spadající pod roli UAT koordinátora jsou zobrazeny v prostřední dráze. Oprava případných bugů nalezených během testování je zahrnuta do činnosti exekuce FAT/UAT scénářů.

5.5 Kritéria pro přechod mezi fázemi testování releasu

Aktuálně Pojišťovna nemá žádn pevně definovaná kritéria pro přechod mezi jednotlivými fázemi testování releasu. Nepřítomnost těchto kritérií může vést k opomenutí důležitých činností, zpoždění dodávek softwaru, nejasnosti ohledně odpovědnosti za jednotlivé činnosti, a hlavně k inkonsistenci mezi jednotlivými releasy.

Následující tabulka rozděluje proces softwarové implementace do šesti fází a pro každou stanovuje činnosti, které musí být splněny před postupem do další fáze. Každá činnost má definovaného vykonavatele, který odpovídá za vykonání dané činnosti, a svého příjemce, který přebírá a kontroluje výstupy činnosti.

Činnost	Vykonává	Přebírá
1) Před FAT a UAT		
Business analýza pro požadavky na změnu zařazené v releasu	Business Analytik	Analytik
Sdílení plánovaných změn všem zemím	UAT koordinátor	Business
IT analýza pro všechny požadavky v releasu	Analýza	Test leader
Alokace kapacit testingu	Test manažer	Test leader
Testovací prostředí FAT & UAT	Release inženýr	Test leader
Připravené test analýzy a testovací scénáře	Testing	Test leader
Připravené UAT testovací scénáře	Business	UAT koordinátor
Předání informací z projektů release týmu	Projektové týmy	Release tým
Příprava automatických testů pro release	Testing	Test leader
2) Před SIT		
Zafixovaný obsah releasu	Release manažer	Test leader
Dokončené FAT & UAT testování	Testing + Business	Test leader
Prezentace obsahu releasu	Analýza	Testing a Business
Připravená testovací dokumentace (test scénáře, test plán, dashboardy atd.)	Test leader	Release Manažer
Připravená testovací data	Testing	Test leader
Spojení kódu do jedné větve	Release inženýr	Release manažer
Alokace zdrojů na testování a opravu chyb	Linioví manažeři	Release manažer
Úspěšný sanity check	Testing	Test leader
3) Před regresí		
Funkční prostředí (úspěšný sanity check)	Testing	Test leader
Dokončené SIT	Testing	Test leader
Připravená testovací data	Testing	Test leader
4) Před RAT		
Report o výsledcích SIT a regresního testování	Test leader	RAT koordinátor
Předání UAT testovacích scénářů	UAT koordinátor	RAT koordinátor
Připravená RAT dokumentace (test scénáře, test plán, dashboardy pro reportování atd.)	RAT koordinátor	Test leader a Release Manažer
Připravená testovací data	RAT Koordinátor	Business Analytik

Alokace zdrojů – Business (testování) a podpora (oprava bugů, pomoc s testováním)	Business Analytik Release Manažer	RAT Koordinátor
5) Před nasazením na produkci		
Report o konečném stavu testování (SIT, Regrese a RAT testování)	Test leader	Release manažer
Příprava automatického sanity check na před-produkčním prostředí	Testing	Test leader
Nasazení kódu na před-produkční prostředí	Release inženýr	Release manažer
Úspěšný sanity check na před-produkci	Testing	Test leader
Příprava kódu pro nasazení na produkci	Release inženýr	Release manažer
Stanovení datum nasazení na produkci	Release manažer	Business analytici
Podepsání akceptačního protokolu	Business Analytici	Release manažer
6) Po nasazení na produkci		
Vyhodnocení releasu	Business analytik	Release manažer
Podpora po releasu	IT oddělení	Business
Aktualizace dokumentace	Zapojené strany	Odpovědné osoby

Tabulka 5: Kritéria pro přechod mezi jednotlivými fázemi testování releasu (vlastní zpracování)

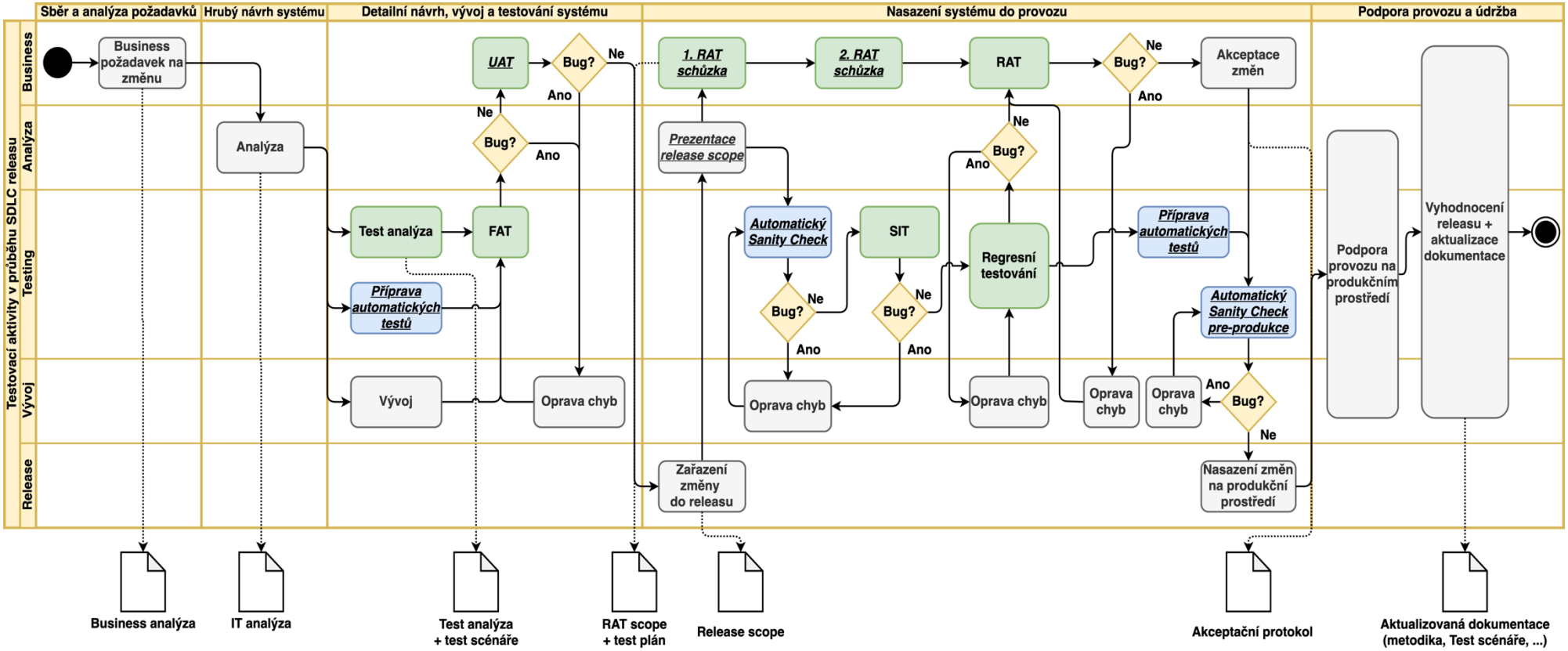
5.6 Testovací aktivity v průběhu releasu

Cílem této kapitoly je zobrazit všechny navržené inovace současného procesu implementace softwaru do jednotlivých fází životního cyklu vývoje softwaru. Grafické zobrazení procesu testování v průběhu releasu je na obrázku 23 na následující straně.

Diagram vychází ze současného procesu zobrazeném na obrázku 20 v kapitole 4.3. a doplňuje ho o nově přidané/upravené činnosti. Nové či změněné činnosti jsou v diagramu vyznačeny kurzívou a podtrženým textem. Změnou oproti původnímu diagramu z kapitoly 4.3 je zařazení automatizovaných testů, rozdělení plánování RAT do dvou schůzek a přesunutí prezentace release scope před sanity a SIT testy.

Aktivity související s automatizací testování jsou v diagramu znázorněny modře a ostatní aktivity vykonávané přímo testingem, nebo za jeho spolupráce/koordinace jsou znázorněny zeleně.

Změny v UAT procesu jsou z důvodu přehlednosti diagramu zahrnuty v činnosti "UAT". Diagram znázorňující detail nově navrženého sub-procesu je v kapitole 5.4.



Obrázek 23: Inovace testovacích aktivit v průběhu SDLC releasu (vlastní zpracování)

5.7 Agilní release

Dalším krokem k inovaci a zefektivnění procesu implementace softwaru v Pojišťovně je přechod na agilní metodiku. V současnosti sice některé pracovní skupiny (technický vývoj a některé z projektů) využívají agile během vývoje, ale do produkce se obvykle dostávají až standardizovaným releasem, který je řízen vodopádově.

Nevýhodou tohoto přístupu je dlouhá doba od zadání požadavku na změnu po nasazení na produkci. Navíc díky velikosti současných releasů, které jsou prováděny pro všechny země a vícero aplikací najednou, je obtížné najít termín releasu, který se hodí všem zúčastněným stranám. Díky tomu aktuálně probíhají pouze tři, maximálně čtyři releasy ročně. Nutnost stanovit přesný harmonogram releasu dopředu a tlak na dodržení data nasazení na produkci téměř znemožňují jakékoliv změny v harmonogramu (například kvůli změnám v požadavcích, nefunkčnímu prostředí či pomalému vývoji/testování).

Situaci, kdy novou verzi testovaného softwaru ještě nelze nasadit na produkci a současně nejsou možné změny harmonogramu, lze řešit dvěma způsoby. První je zrušení daného releasu s tím, že jeho obsah bude přidán do dalšího. Druhou možností je nasadit chybovou verzi na produkci a dodávat opravy během provozu.

Přechod na Agile by pro Pojišťovnu znamenal více menších releasů napříč celým rokem, rozdílné termíny releasů pro jednotlivé země kvůli větší flexibilitě a z toho plynoucí zkrácení doby od zadání požadavku po nasazení na produkci.

Hlavními předpoklady pro zavedení agile je přechod na agilní vývoj a důraz na automatizaci testování. Agile zkracuje celkovou dobu softwarové implementace a s tím i čas dostupný na testování, hlavně regresní testy. Z tohoto důvodu je potřeba automatizovat co nejvíce testů, aby mohly být pouštěny během každého sprintu a ušetřily tak čas, který by byl jinak potřeba na manuální testování.

6 Doporučení k zavedení inovace

Cílem této kapitoly je stanovit seznam doporučení pro úspěšné zavedení návrhové části této diplomové práce do současné situace v IT sekci Pojišťovny. Záměrem je, aby změny co nejméně narušily běžný chod společnosti a co nejlépe plnily svůj účel. Doporučení jsou postupně přiřazena k jednotlivým částem návrhu:

1. *Harmonogram releasu* – harmonogram včetně konkrétních dat je potřeba včas komunikovat všem zapojeným stranám, aby o plánovaných změnách věděly a počítaly s nimi. V případě jakýchkoliv změn, či problémů je třeba reagovat co nejrychleji. Je důležité mít na paměti, že odložení releasu je mnohdy menší zlo než nasadit defektní software na produkční prostředí. Odložení sice pro business znamená, že si bude muset déle počkat na nové funkcionality, ale na druhou stranu bude alespoň produkce fungovat tak jak předtím.

2. *Automatizace testování* – je důležité správně nastavit procesy pro práci s automatizovanými testy, aby bylo transparentní, kdy a jak jsou tvořeny, spouštěny a aktualizovány. Společná evidence manuálních a automatizovaných testů odhalí případné duplicity v testování a pomůže odhalit oblasti, které nejsou kryté žádnými testy.

3. *Kvalita dodávky* –metriky je důležité nejen sledovat a vyhodnocovat, ale také na jejich základě provádět případné úpravy do dalších releasů. S hodnocením průběhu implementace softwaru také souvisí průběžné reportování o stavu releasu. Tyto reporty by měly být standardizované napříč všemi pracovními skupinami a releasy, aby byly transparentní a snadno porovnatelné. Reportování musí probíhat v průběhu celého procesu, a to všem podílejícím se na releasu.

4. *Koordinace uživatelského testování* – koordinace uživatelského testování je hlavně o efektivní komunikaci s businessem. Pro pozvolné zavedení změny je možné začít s aplikací tvorby testovacích scénářů businessem postupně po jednotlivých zemích a proces průběžně, dle potřeby upravovat.

5. *Kritéria pro přechod mezi fázemi testování releasu* – aby byla kritéria řádně dodržována, je třeba stanovit osobu, která za ně bude odpovídat. Ideálním kandidátem je test leader. Kritéria a jejich stav pro konkrétní release by také měla být dostupná celému týmu podílejícímu se na implementaci softwaru. Tím bude dosaženo transparentnosti ohledně stavu aktuální fáze testování.

6. *Testovací aktivity v průběhu releasu* – je nutné dodržovat dané pořadí testovacích aktivit a přecházet do další testovací fáze až v okamžiku, kdy byla splněna všechna kritéria pro přechod.

7. *Agilní release* – před samotným přechodem k agilním metodikám je stěžejní připravit organizaci a jednotlivé zaměstnance na to, co změna přináší. Změna, její důsledky a hlavně přínosy, musí být srozumitelně vysvětlena všem, jichž se implementace agile dotkne. Případný odpor zaměstnanců ke změně může být rozhodujícím faktorem, jestli bude inovace implementována úspěšně anebo ne. Samotné zavedení agile pak může probíhat postupně a za pomoci externích odborníků.

Dalším doporučením je začít používat nové metriky definované v rámci kapitoly 4.2.3. Výsledky měřených metrik je důležité zhodnotit v rámci releasové retrospektivy a ponaučit se z nich do dalších releasů.

Závěr

Cílem této práce bylo navrhnout inovace procesu softwarové implementace v Pojišťovně z pohledu testingu. Tento cíl byl splněn.

Návrh inovací procesu vychází z relevantní teorie, je založen na analýze současného stavu a zaměřuje se na sedm oblastí, které byly v analýze identifikovány jako problémové. Těmito oblastmi jsou:

1. Harmonogram releasu
2. Automatizace testování
3. Kvalita dodávky
4. Koordinace uživatelského testování
5. Kritéria pro přechod mezi fázemi testování releasu
6. Testovací aktivity v průběhu releasu
7. Agilní release

Vlastní návrh byl nadále doplněn o doporučení k zavedení navržených inovací do současného procesu softwarové implementace.

Všechny dílčí úkoly diplomové práce byly tedy splněny.

Prostor pro rozšíření diplomové práce tkví především ve vypracování detailního strategického plánu pro přechod IT sekce pojišťovny od vodopádového modelu životního cyklu vývoje a provozu softwaru k agilním metodám. Výhody této transformace byly popsány již v návrhové části této práce, ale je zde prostor pro podrobné zpracování možných dopadů a důsledků implementace agilní metodiky v Pojišťovně. Dalším tématem hodným rozvinutí je vyčíslení finančních dopadů navržených inovací procesu implementace softwaru. Autorovi práce nebyl přístup k finanční dokumentaci IT sekce Pojišťovny umožněn, proto nejsou finanční dopady součástí práce.

Seznam použitých odborných pojmů

- *SDLC (Software Development Life Cycle)* – zkratka z anglického názvu pro životní cyklus vývoje a provozu systému.
- *Bug* – chyba v softwaru, kterou tester odhalil, zahlásil a předal k opravě.
- *Business* – část Pojišťovny zabývající se vývojem a prodejem pojišťovacích produktů. V kontextu této práce se jedná o zákazníky IT sekce.
- *Release* – proces implementace nové verze softwaru a jejího předání koncovým uživatelům do provozu.
- *UAT (User Acceptance Testing)* – testy prováděné uživateli softwaru. Cílem je zjistit, jestli dodaný produkt odpovídá požadavkům.
- *Sanity check* – ověřuje fungování klíčových funkcionalit systému. Výsledek určuje, jestli je daný systém testovatelný, nebo ne.
- Produkční prostředí – souhrn systémů které používá Business při své práci.
- Automatizovaný test – testovací scénář, který není prováděn testerem, ale strojově pomocí automatizačního programu. Použití automatizovaných testů snižuje náklady na testing a zrychluje testování.
- *Sprint* – jasně ohraničený časový úsek během kterého probíhají všechny aktivity agilního vývoje.
- *Product Backlog* – pojem spojený s agilními metodikami. Jedná se o seznam plánovaných funkcionalit řešené aplikace, které jsou vyvíjeny napříč jednotlivými sprinty.
- BPMN – jeden z nejrozšířenějších notačních standardů pro modelování podnikových procesů pomocí diagramů.
- Stakeholder – osoba, firma, či jakýkoliv další subjekt, který je ovlivněn řešenou problematikou.

Seznam použité literatury a zdrojů

1. webové stránky řešené společnosti. [Online] [Citace: 20. 1 2022.]

2. Národní soustava povolání. [Online] [Citace: 29. 1 2022.] https://www.nsp.cz/jednotka-prace/analytik-it.

3. Národní soustava povolání. [Online] [Citace: 29. 1 2022.] https://www.nsp.cz/jednotka-prace/samostatny-programator.

4. Národní soustava povolání. [Online] [Citace: 29. 1 2022.] https://www.nsp.cz/jednotka-prace/softwarovy-tester.

5. Pour, Jan, Gála, Libor a Šedivá, Zuzana. *Podniková informatika.* Praha : Grada, 2015. ISBN 978-80-247-5457-4.

6. Filipova, Olga a Vilão, Rui. *Software Development From A to Z: A Deep Dive into all the Roles Involved in the Creation of Software.* Berkeley, California : Apress, New York, Ny, 2018. 9781484239445.

7. BUREŠ, M., RENDA, M. DOLEŽAL, M. A KOLEKTIV. *Efektivní testování softwaru.* Praha : Grada, 2016. ISBN 978-80-247-5594-6.

8. Jorgensen, Paul C. *Software Testing: A Craftsman's Approach, Fourth Edition.* Boca Raton : CRC Press, 2013. 978-1466560680.

9. Beck, Kent. Agile Manifesto. *Manifesto for Agile Software Development.* [Online] 2001. [Citace: 27. 02 2022.] https://agilemanifesto.org/iso/en/manifesto.html.

10. Efisco. Efisco. *Efisco.* [Online] [Citace: 22. 03 2022.] https://efisco.net/waterfall/.

11. ISTQB. ISTQB Glossary. [Online] 30. 6 2021. [Citace: 7. 3 2022.] https://glossary.istqb.org/en/.

12. Jones, Capers. A SHORT HISTORY OF THE COST PER DEFECT METRIC. *softwarevalue.* [Online] 29. 12 2012. [Citace: 04. 03 2022.] https://www.softwarevalue.com/media/389295/cost-per-defect-2013.pdf.

13. Gála, libor, Pour, Jan a Šedivá, Zuzana. *Podniková informatika Počítačové aplikace v podnikové a mezipodnikové praxi - 3., aktualizované vydání.* Praha : Grada, 2015. 978-80-247-5457-4.

14. Řepa, Václav. *Podnikové procesy: procesní řízení a modelování.* Praha : Grada, 2006. ISBN 80-247-1281-4.

15. Šimonová, Stanislava. *Procesní řízení.* Pardubice : Univerzirta Pardubice, 2015. ISBN 978-80-7395-766-7.

16. Grasseová, Monika. *Procesní řízení ve veřejném sektoru: Teoretická východiska a praktické příklady.* Brno : Computer Press, 2008. ISBN 978-80-251-1987-7.

17. Svozilová, Alena. *Zlepšování podnikových procesů.* Praha : Grada Publishing, Praha. ISBN 978-90-247-3938-0.

18. HAMMER, M., HERSHMAN, L. *Rychleji, levněji, lépe. Devět faktorů účinné transformace podnikových procesů.* Praha : Management press, 2013. ISBN 978-80-7261-253-6.

19. Řepa, Václav. *Procesně řízená organizace.* Praha : Grada, 2012. ISBN 978-80-247-4128-4.

20. AXELOS. *ITIL® Foundation, ITIL 4 edition.* London : TSO, 20119. ISBN 9780113316076.

Seznam obrázků

Seznam grafů

Seznam tabulek

Všechny obrázky, grafy a tabulky, bez uvedeného zdroje, byly vytvořeny autorem pro účely této diplomové práce.

Evidence výpůjček

Prohlášení:
Dávám svolení k půjčování této bakalářské práce. Uživatel potvrzuje svým podpisem, že bude tuto práci řádně citovat v seznamu použité literatury.

Jméno a příjmení: David Pavelka
V Praze dne: 27. 04. 2022 Podpis:

Jméno	Oddělení/Pracoviště	Datum	Podpis