BAKALÁŘSKÁ PRÁCE

Návrh procesu automatizování testů v rámci vývoje softwaru pro pojišťovnu

Designing a Software Test Automation Process for Insurance Company

STUDIJNÍ PROGRAM

Ekonomika a management

STUDIJNÍ OBOR

Řízení a ekonomika průmyslového podniku

VEDOUCÍ PRÁCE

doc. Ing. Martin Zralý, CSc.

PAVELKA

DAVID

2020

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Pavelka Jméno: David Osobní číslo: 475141

Fakulta/ústav: Masarykův ústav vyšších studií (MÚVS)

Zadávající katedra/ústav: Oddělení ekonomických studií

Studijní program: Ekonomika a management

Studijní obor: Řízení a ekonomika průmyslového podniku

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh procesu automatizování testů v rámci vývoje softwaru pro pojišťovnu

Název bakalářské práce anglicky:

Designing a Software Test Automation Process for Insurance Company

Pokyny pro vypracování:

Cíl: Cílem této práce je navrhnout proces pro řízení tvorby automatizovaných testů v IT oddělení na pobočce nadnárodní pojišťovny, tak aby odpovídal zásadám procesního řízení a byl co nejefektivnější. Východiskem návrhu bude analýza současného přístupu k tvorbě automatizovaných testů v pojišťovně a rešerše relevantních pramenů.
Přínos: Návrh automatizovaných testů včetně doporučení pro jeho implementaci.
Osnova: 1. Úvod, 2. Relevantní teorie, 3. Charakteristika řešené společnosti, 4. Analýza současné situace, 5. Návrh řešení 6. Doporučení k implementaci, 7. Shrnutí a zhodnocení výsledků, 8. Závěr.

Seznam doporučené literatury:

BUREŠ, M., RENDA, M., DOLEŽAL, M. A KOL. Efektivní testování softwaru. Praha: Grada, 2016.
ŘEPA, V., Podnikové procesy. Praha: Grada, 2007, druhé aktualizované vydání
ŘEPA, V, Procesně řízená organizace. Praha: Grada, 2012
HAMMER, M., HERSHMAN L., Rychleji, levněji, lépe,. Praha: Management Press, 2013

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Martin Zralý, CSc., MÚVS ČVUT v Praze, oddělení ekonomických studií

Jméno a pracoviště konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: 30.11. 2019 Termín odevzdání bakalářské práce: 30.4. 2020

Platnost zadání bakalářské práce: 30. 9. 2021

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

27. 03. 2020

Datum převzetí zadání

Podpis studenta(ky)

PAVELKA, David Návrh procesu automatizování testů v rámci vývoje softwaru pro pojišťovnu. Praha: ČVUT 2020. Bakalářská práce. České vysoké učení technické v Praze, Masarykův ústav vyšších studií.

MASARYKŮV ÚSTAV
VYŠŠÍCH STUDIÍ
ČVUT V PRAZE

Prohlášení

Prohlašuji, že jsem svou bakalářskou 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: 15 5. 2020 Podpis:

Poděkování

Mé díky patří vedoucímu této práce panu doc. Ing. Martinovi Zralému, CSc. za jeho odborné vedení, cenné rady a vstřícnost při konzultacích.

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

V neposlední řadě bych rád poděkoval své rodině a blízkým za jejich podporu, pomoc a trpělivost.

Abstrakt

Tato bakalářská práce předkládá návrh procesu řízení tvorby automatizovaných testů pro IT sekci pobočky nadnárodní pojišťovny. Vlastní návrh procesu automatizace testování je obohacen o rozhodovací schéma, které slouží jako opora při rozhodování o výhodnosti automatizace.

Návrh vychází z relevantní teorie a z podrobné analýzy současné situace v IT, se zaměřením především na její testingové oddělení. Navržené schéma a procesy jsou graficky znázorněny pomocí diagramů odpovídající standardu BPMN. Práci uzavírají doporučení k implementaci vytvořeného procesu a pokyny pro práci s automatizovanými testy.

Klíčová slova

Automatizace testování; testování softwaru; BPMN standard; modelování procesů; životní cyklus vývoje a provozu systému

Abstract

This bachelor's thesis proposes the process of a software test automation for the IT section of an insurance company. The process proposal is extended by a decision tree, serving as a supportive tool for deciding on a profitability of test automation for a given project.

The proposal is based on a relevant theory and on a detailed analysis of the current situation in IT section, mainly focused on its testing department. The proposed process and decision tree are graphically portrayed by diagrams in BPMN standard. The thesis is concluded by a list of recommendations regarding the implementation of a proposed process and by the guide on the execution of automated tests.

Key words

Test automation; software testing; BPMN standard; process modelling; software development life cycle

Obsah

1 Úvod

Každý softwarový produkt musí být před tím, než může být nasazen do provozu, řádně otestován. V testování, stejně jako v každé další firemní činnosti, jde především o rychlost a cenu. Testování může proběhnout buď manuálně, nebo automatizovaně. Rozhodování o tom, zda se automatizace testování v daném případě vyplatí a jak by případná automatizace měla probíhat, je komplikované. Tato bakalářská práce je motivována přáním pomoci společnosti, v této práci řešené, výše uvedené činnosti co nejvíce usnadnit.

1.1 Cíl práce

Cílem práce je navrhnout proces pro řízení tvorby automatizovaných testů v rámci IT sekce české pobočky nadnárodní pojišťovny a následně vytvořit doporučení pro jeho implementaci.

Pro získání kontextu je nejdříve nutné představit řešenou společnost (dále v textu „Pojišťovna"), popsat, jakou roli v ní hraje sekce IT a jak je v současnosti řízen testing. Tyto informace poslouží jako vstupy pro návrh procesu.

Před vlastním návrhem procesu je nutné analyzovat současný přístup Pojišťovny k řízení automatizace testů a k zasazení testingu do rámce životního cyklu vývoje softwaru. Na základě současného stavu a s využitím relevantní teorie je vytvořeno rozhodovací schéma určující, kdy se automatizace testování vyplatí a kdy je naopak efektivnější pouhé manuální testování.

Samotný proces je navržen v souladu s postupy správného procesního řízení, představenými v rámci rešerše relevantních pramenů. Teoretická část práce obsahuje také úvod do teorie testování softwaru a jeho automatizace. Zabývá se rovněž představením procesu vývoje softwaru jako takového.

Pro naplnění cíle práce je nezbytné porozumět současnému stavu v Pojišťovně a s použitím příslušné odborné literatury vytvořit vlastní návrh nového procesu a jeho vhodné implementace. Návrh vychází z osvědčených testovacích metodik, jako je životní cyklus vývoje a provozu systému, či W-model. Proces je následně graficky namodelován pomocí standardu BPMN.

1.2 Úkoly práce

Výše uvedený cíl práce lze rozvést do následujících úkolů práce:

1. Představení Pojišťovny
2. Charakteristika IT sekce Pojišťovny
3. Relevantní teorie
4. Analýza současné situace v testingu a přístupu Pojišťovny k řízení automatizace testů
    a. Popis současného stavu v testingovém oddělení
    b. Analýza současného stavu automatizace testů
5. Tvorba rozhodovacího schématu
6. Návrh procesu automatizace testů pro Pojišťovnu
7. Doporučení pro implementaci navrženého procesu
8. Shrnutí a vyhodnocení výsledků práce

1.3 Osnova práce

Úvod
V rámci první kapitoly je představen cíl bakalářské práce (BP), naznačen postup jejího zpracování a jsou vymezeny technické a manažerské nástroje, které jsou v práci využity. Dále jsou vytyčeny úkoly, jejichž splnění je třeba pro dosažení cíle práce. Kapitola také obsahuje stručný obsah jednotlivých částí BP, včetně uvedení jejich smyslu.

Charakteristika řešené společnosti
Druhá kapitola[1] se zabývá představením Pojišťovny, pro kterou je návrh procesu pro řízení tvorby automatizovaných testů vypracován. Detailněji je pak specifikována IT sekce Pojišťovny.

Relevantní teorie
Třetí kapitola se věnuje identifikaci relevantních zdrojů a představuje základní teorii, ze které vychází návrh práce. Teorie použitá v této BP je rozdělena do 4 oblastí, klíčových pro dosažení cíle práce:

– Základní teorie vývoje softwaru

[1] Pořadí kapitol Relevantní teorie a Charakteristika řešené společnosti bylo v práci oproti zadání BP prohozeno z důvodu lepší návaznosti mezi jednotlivými kapitolami.

- Teorie testování softwaru
- Automatizace testování
- Tvorba podnikových procesů

Analýza stávající situace
Čtvrtá kapitola analyzuje současnou situaci v testingovém oddělení Pojišťovny. Především se zaměřuje na jeho přístup k automatizaci testování a popisuje zasazení testovacích fází do životního cyklu vývoje a provozu systému.

Návrh řešení
Pátá kapitola obsahuje vlastní návrh procesu automatizace testování v rámci Pojišťovny. Vychází z analýzy stávající situace a využívá relevantní teorii. Vlastní návrh se skládá ze dvou hlavních částí:

- Tvorba rozhodovacího schématu
- Návrh procesu automatizace testování

Návrh procesu automatizace testování je následně zasazen do kontextu současných procesů testingového oddělení.

Doporučení k implementaci
Šestá kapitola se soustředí na doporučení pro co nejefektivnější implementaci navrženého procesu do současné struktury Pojišťovny.

Shrnutí a vyhodnocení výsledků práce
V této kapitole jsou shrnuty výsledky práce a je vyhodnoceno, zda bylo dosaženo stanovených dílčích úkolů práce.

Závěr
Poslední kapitolou je závěr. Obsahuje vyhodnocení cíle a dílčích úkolů práce a uvádí možnosti dalšího rozšíření BP.

2 Charakteristika řešené společnosti

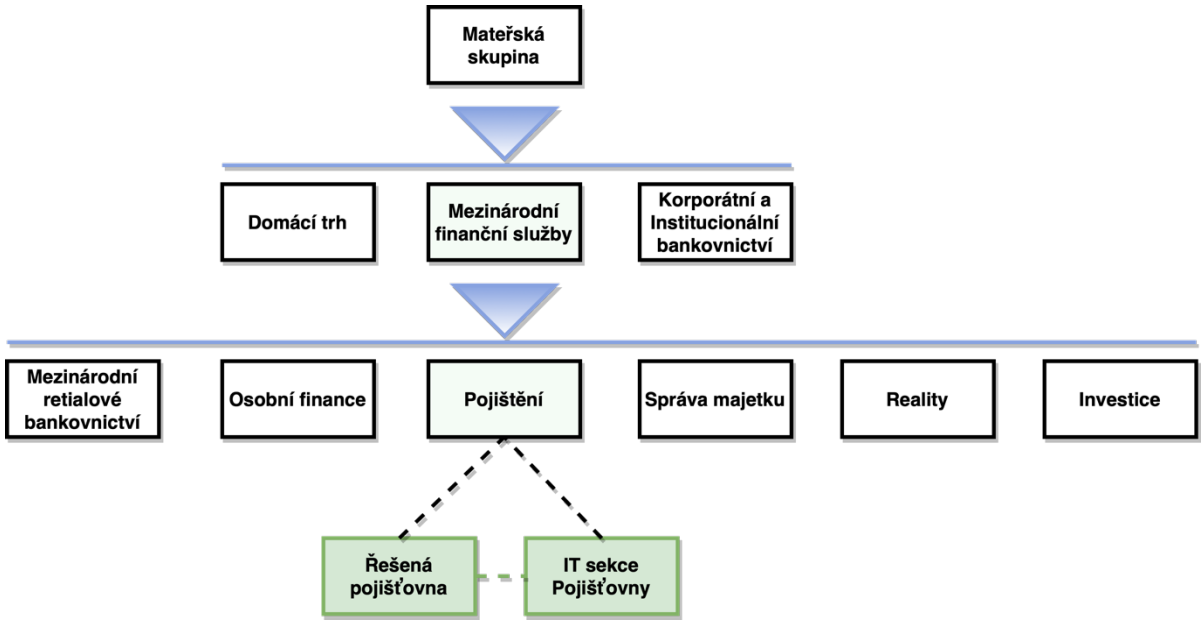
Bakalářská práce je zaměřena na návrh procesu automatizování testů v rámci vývoje softwaru na české pobočce nadnárodní pojišťovny. V této kapitole se nejdříve stručně seznámíme s Pojišťovnou a poté s její IT sekcí. Na žádost vedení firmy bude Pojišťovna v rámci této práce anonymizovaná.

2.1 Pojišťovna

Řešená Pojišťovna je akciovou společností se sídlem v Praze. Společnost je součástí jedné z největších světových finančních skupin. Skupina napomáhá svým klientům (jednotlivcům, podnikatelům, malým a středním podnikům a korporacím) realizovat jejich projekty pomocí svého produktového portfolia zahrnující financování, investice, spoření a pojištění. Historie bank, které později vytvořily skupinu se datuje od 19. století a v současné době působí ve více jak 70 zemích světa na pěti kontinentech a zaměstnává téměř 200 tisíc lidí (1).

Na českém trhu Pojišťovna působí již od devadesátých let, kdy jako jedna z prvních spustila produkt Pojištění schopnosti splácet finanční závazek, který klienty chrání v případě nepředvídatelných životních situací, které mohou ohrozit jejich schopnost splácet úvěr, hypotéku nebo například leasing. Postavení, které mají v rámci společnosti Pojišťovna a IT sekce k ní přidružená, je zobrazeno na obrázku 1.

V současnosti nabízí Pojišťovna produkty ve spolupráci s řadou finančních institucí – domácích i nadnárodních partnerů – nejčastěji s bankami, leasingovými společnostmi a společnostmi poskytujícími nákupy na splátky, osobní půjčky nebo úvěrové karty. Dále také spolupracuje s předními společnostmi napříč obory – například retailu (elektro), telekomunikací nebo dodavatelů energií (1).



Obrázek 1: Postavení Pojišťovny v rámci skupiny

2.2 IT sekce

IT sekce přidružená k české pobočce představuje klíčovou součást Pojišťovny. Funguje totiž jako sdílené servisní centrum pro region centrální a východní Evropy. V rámci IT tak dochází k vývoji a správě většiny aplikací používaných ve zmíněném regionu.

Sekce IT má právní formu společnost s ručením omezeným s jediným společníkem, a to Pojišťovnou. Obě tyto společnosti sídlí v Praze. Vzhledem k tomu, že IT dodává své služby pouze dalším společnostem ve skupině, jsou skupina spolu s Pojišťovnou jeho primárním zdrojem příjmů.

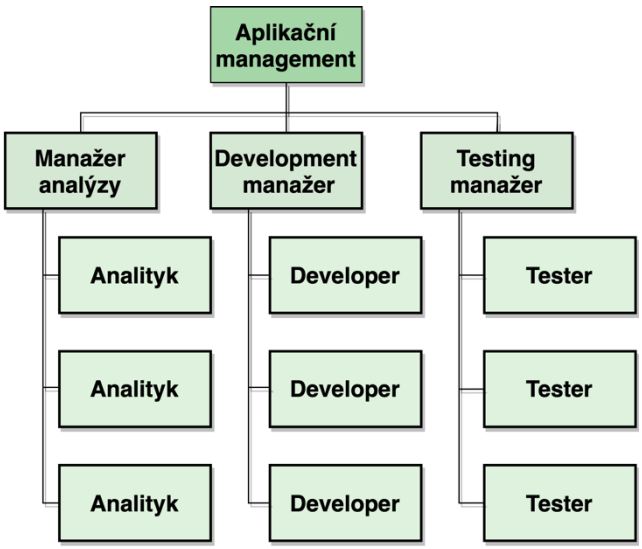
2.2.1 Organizační struktura

Směřování IT sekce udávají jednatelé. Ti také vyhodnocují finanční analýzy, rozhodují o firemní politice a řídí vztahy s Pojišťovnou a mateřskou firmou. Dodávku produktů pro sdílené servisní centrum řídí tým aplikačního managementu.

Aplikační management je odpovědný za řízení vyvíjených aplikací v rámci celého jejich životního cyklu a za včasnost dodávky produktů. Pod aplikační management spadají tři podsekce, v nichž se realizují klíčové činnosti tvorby aplikací – viz obrázek 2. Každá z těchto sekcí je řízená vlastním manažerem:

- *Analýza* – analytici zpracovávají požadavky ostatních oddělení Pojišťovny (likvidace, obchodní oddělení, call centrum, ...) a regulátorů (ČNB, GDPR, EU atd.) na nové aplikace, či funkcionality již zavedených systémů. Tyto požadavky pak zahrnou do analýzy, která obsahuje technické provedení, funkční a legislativní požadavky. Výstupem práce analytiků jsou technické analýzy a schématické diagramy softwarových aplikací i jejich celků (2).
- *Development* – developeři, neboli vývojáři, mají na starost vývoj softwarových řešení dle specifikací a dokumentace od analytiků (3).
- *Testing* – hlavní pracovní náplní testerů je testování výstupů (aplikace, programové komponenty atp.) developerů a zpracovávání výsledků testů. Testeři jsou zapojeni nejen na konci, ale i v průběhu celého životního cyklu. Například připravují testovací data, ověřují specifikace analýzy, či testují jednotlivé fáze návrhu (4).

Všechny pozice spadající pod manažery sekcí jsou pro zjednodušení schématu nazvána stejně. Ve skutečnosti jsou zaměstnanci v každé ze tří sekcí ještě rozděleni podle svých rolí. Podrobnému popisu rolí v rámci testingového oddělení a náplni jejich práce se věnuje kapitola 4.1.1.



Obrázek 2: Organizační struktura IT oddělení

2.2.2 Vyvíjený software

Dle požadavků poboček Pojišťovny v České republice a v ostatních zemích regionu centrální a východní Evropy vyvíjí a dodává IT oddělení řadu aplikací:

- Správa likvidace – software zajišťující komplexní správu agendy likvidátorů; aplikace umožňuje vysokou variabilitu prostředí, aby si ji mohla každá země nastavit dle svých požadavků
- Správa importů – program pro zpracování dat od partnerů
- Portál pro partnery – online portál pro příjem pojistných událostí od partnerů
- Portál pro klienty – online portál umožňující zákazníkům hlásit pojistné události
- Bankovní reporty – program pro generování bankovních reportů
- Bankovní příkazy – software pro automatickou správu bankovních příkazů

2.3 Popis problému

Společnosti v současnosti chybí jasně definovaná a formulovaná strategie pro automatizaci testování v rámci vývoje softwaru. Pojišťovna nemá nastavený žádný proces pro tvorbu a údržbu automatizovaných testů. Dále také chybí kritéria pro výběr oblastí vhodných k automatizování.

Právě návrhem a nastavením procesu, včetně souvisejících rozhodovacích, respektive výběrových kritérií, se bude návrh této práce zabývat.

3 Relevantní teorie

Úkolem této kapitoly je popsat teoretická východiska použitých metod, postupů a pojmů, které jsou v práci využity. Kapitola je rozdělena do čtyř částí. V první jsou stručně představeny teoretické základy spojené s vývojem softwaru. Druhá předkládá základní teorii testování softwaru. Na ní přímo navazuje další část, ve které je představena teorie automatizace testování. Čtvrtá, poslední část, se věnuje zásadám procesního řízení a tvorbě efektivních firemních procesů.

3.1 Základní teorie vývoje softwaru

Jak je popsáno v (5) (kapitola 12) výběr metod a postupů pro vývoj a rozvoj podnikového softwaru je ovlivněn požadovanou funkčností vyvíjené aplikace a také charakterem podniku a jeho produktů. Doporučené postupy řešení aplikací, nebo rovnou celých informačních systémů jsou označovány jako metodiky. Většina firem využívá vlastní metodiky, nebo disponuje standardizovanými metodikami, které si upravují podle vlastních potřeb.

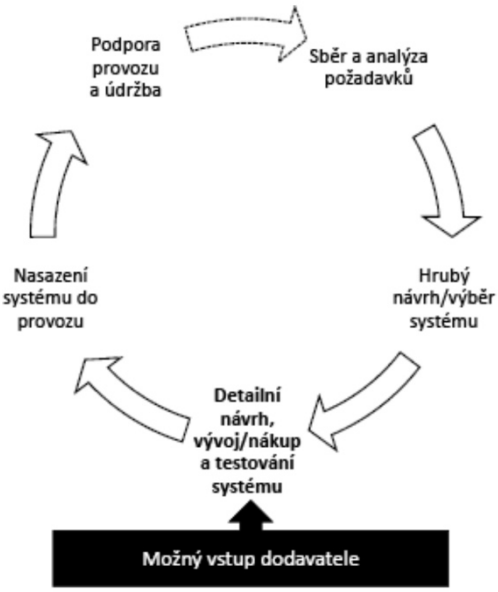
3.1.1 Životní cyklus vývoje a provozu systému

Tato kapitola se zaměřuje se na objasnění průběhu vývoje systému v rámci jeho jednotlivých fází. Celý proces vývoje a řízení systému označujeme jako jeho životní cyklus (anglicky Software Development Life Cycle – zkratka SDLC). Označení systém zde zastupuje celek složený z počítačového hardwaru a souvisejícího softwaru. SDLC zahrnuje činnosti, které lze rozdělit do jednotlivých fází procesu. Fáze životního cyklu jsou obvykle členěny takto:

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

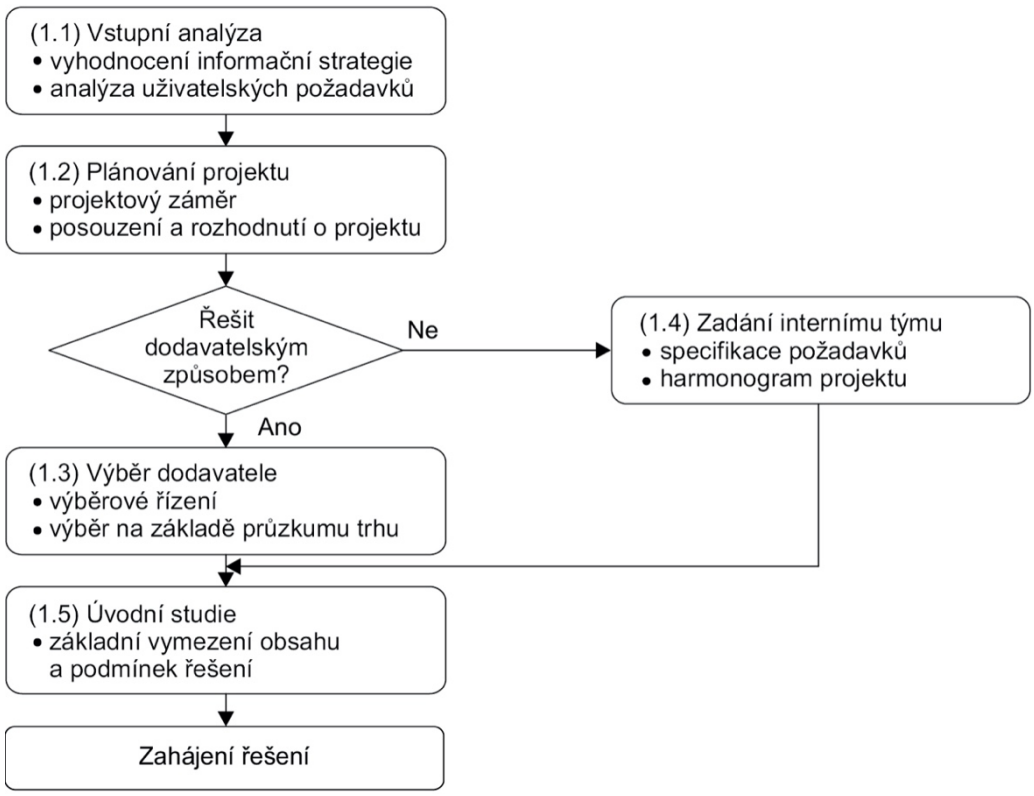
Termín životní cyklus systému byl zaveden proto, že od začátku prochází systém jednotlivými vývojovými fázemi a po určité době svého provozu a rozvoje se vrací zpět na začátek cyklu, kdy je potřeba vyvinout další, novou, úroveň aplikace. Tento požadavek je založen na nových potřebách a požadavcích uživatelů aplikace, anebo na aktuální dostupnosti pokročilejších technologií (5). Životní cyklus vyjadřuje dlouhodobou perspektivu a končí až finálním vyřazením systému z provozu.

Obrázek 3 dokumentuje průběh životního cyklu vývoje a provozu softwarového systému v rámci organizace. Základní obsah a principy jednotlivých fází jsou uvedeny níže.



Obrázek 3: Životní cyklus vývoje a provozu systému (8)

– Sběr a analýza požadavků – tato fáze začíná záměrem tvorby aplikace a je ukončena jasnou odpovědí na to, zda bude aplikace realizována, nebo ne. Během této fáze je vytvořena vstupní analýza, plán projektu a je rozhodnuto, jestli bude aplikace řešena dodavatelským způsobem. Průběh fáze znázorňuje obrázek 4.



Obrázek 4: Úlohy plánovací fáze (5)

– Hrubý návrh/výběr systému – zde dochází k tvorbě konceptuálního návrhu systému. Ten obsahuje informaci o tom, jakými pravidly se bude systém řídit (tyto pravidla jsou také označována jako Business logika).

– Detailní návrh, vývoj/nákup a testování systému – během této fáze již dochází k samotnému vývoji (develpomentu) sytému. Vývoj může být zajištěn interně, firemními developery, nebo může být zakoupen od externího dodavatele. V rámci této fáze přichází na řadu také testování systému.

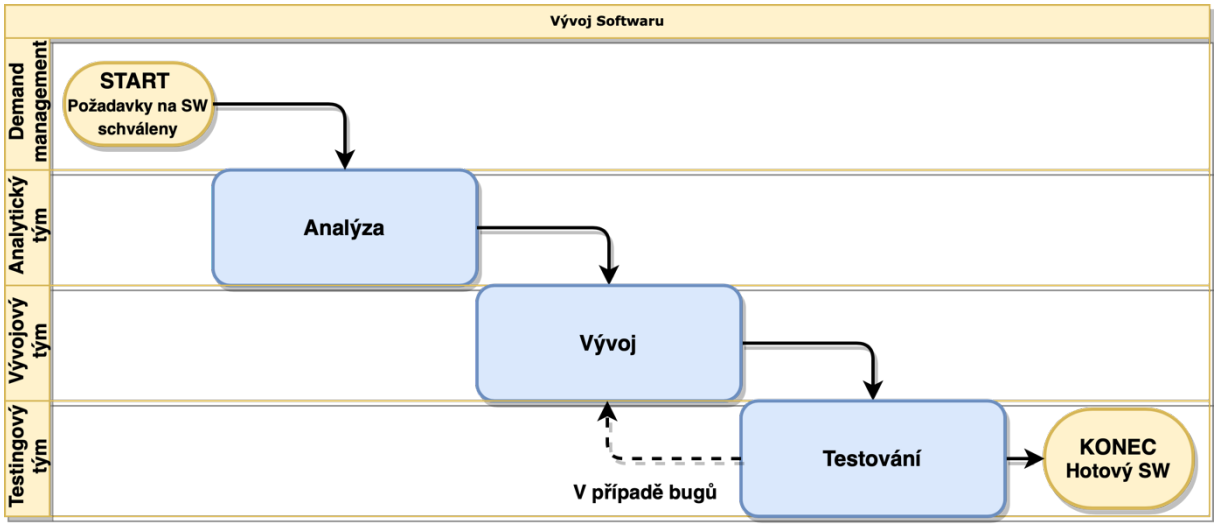
– Nasazení systému do provozu – jakmile je systém naprogramován a prošel akceptačním testováním, je nasazen do provozu, aby sloužil svému účelu.

– Podpora provozu a údržba – v průběhu používání softwaru je poskytována podpora a údržba jeho uživatelům. V případě, že uživatelé přijdou s požadavkem na novou funkcionalitu, nebo s žádostí o opravu systému, celý cyklus začíná nanovo.

3.1.2 Vodopádový model

Druhá a třetí fáze cyklu, tedy návrh a vývoj systému, pokud jsou prováděny interně jsou často prováděny pomocí „Vodopádového principu dodávky softwaru". Tento způsob dodávky funguje na principu postupného předávání práce – nejprve analytici zpracují detailní analýzu, tu předají vývojářům, kteří podle ní naprogramují systém. Jakmile je systém provozuschopný, přechází na testování. Jestliže v něm testing nalezne nějaké chyby (bugy), vrátí systém zpět vývojářům na opravu.

Celý proces vodopádového principu je zobrazen diagramem na obrázku 5.



Obrázek 5: proces vývoje softwaru podle vodopádového principu

3.2 Teorie testování softwaru

Testing neboli testování, je název pro proces, kdy se snažíme ověřit, zda testovaný subjekt funguje tak, jak má, splňuje dané specifikace a drží se předem stanovených pravidel (6). Všechny produkty, které uvádíme na trh, by měly nejdříve projít testováním. Mnohdy je to dokonce i zákonnou podmínkou – léky, potraviny, ale například ani dětské hračky nemohou být prodávány, aniž by byly podrobené testování a musí splňovat dané nároky. Testing tedy slouží jako kontrolní proces odpovídající za výslednou kvalitu produktu. Můžeme ho chápat jako podmnožinu procesu ověřování a plánování kvality.

Obdobnou roli hraje testování i v rámci vývoje softwaru. S tím rozdílem, že na většinu softwaru nejsou kladeny legislativní požadavky na kvalitu, a tak se mohou testovací postupy a metody napříč firmami výrazně lišit. Výjimkou je pouze finanční sektor společně s jakýmkoliv softwarem pracujícím s osobními, či citlivými údaji[2]. Avšak jako ve všech odvětvích, tak i v software testingu existuje Best Practice – osvědčené postupy pomocí kterých se ve více organizacích dosáhlo dobrých výsledků, a proto se používají jako doporučení ostatním (7). Testingové Best Practice bude sloužit jako základ i pro řešení navrhované touto prací.

3.2.1 Testovací proces

Samotný proces testování systému lze rozdělit do několika úrovní. Dle (8) (kapitola 2.3.1) je úroveň testování definována jako „skupina testovacích aktivit, které jsou společně organizovány a řízeny“. Jednotlivé úrovně by měly mít samostatně odlišené cíle a odpovědnosti, aby byl proces detekce chyb co nejefektivnější (8).

S myšlenkou úrovní testování pracuje W-model popsaný v následující kapitole.

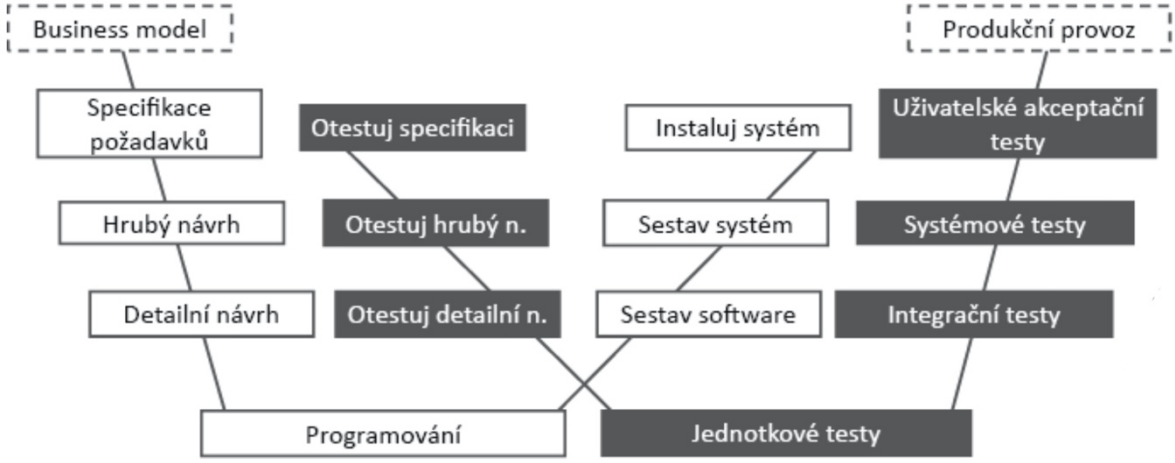
3.2.1.1 W-model

Jedná se o aplikaci životního cyklu vývoje systému (popsáno v kapitole 3.1) rozšiřující všeobecně známý vodopádový model se zvýšeným důrazem na zapojení testingu již v raných fázích vývoje. Model dává do přímé souvislosti vývojové a testovací aktivity. Model zapojuje testování do průběhu celého projektu – od původních business

2 Osobní údaje (9) jsou dle stávající legislativy a GDPR definovány jako veškeré informace vztahující se k identifikované či identifikovatelné fyzické osobě. Konkrétně se jedná například o jméno, pohlaví, věk a datum narození, osobní stav, ale také IP adresu a fotografický záznam. Mezi osobní údaje řadíme i tzv. organizační údaje, kterými jsou například e-mailová adresa, telefonní číslo či různé identifikační údaje vydané státem. Citlivé osobní údaje (10) jsou speciální kategorií podle GDPR, která zahrnuje údaje o rasovém či etnickém původu, politických názorech, náboženském nebo filozofickém vyznání, členství v odborech, o zdravotním stavu, sexuální orientaci a trestních deliktech či pravomocném odsouzení osob. Tento typ údajů může svůj subjekt vážně poškodit ve společnosti. Citlivé údaje zahrnují také genetické a biometrické údaje. Zpracování citlivých osobních údajů podléhá ještě přísnějšímu režimu, než je tomu u obecných osobních údajů.

požadavků, přes kontrolu jednotlivých fází návrhu až po kontrolu průběhu jednotlivých úrovní testování samotného systému.

W-model je zobrazen na obrázku 6. Bílými obdélníky tu jsou označeny vývojové aktivity. Aktivity testingu, neboli jednotlivé typy použitých testů, jsou znázorněny obdélníky tmavými. Testy na levém rameni tmavého V jsou testy statické, ty na pravém rameni jsou dynamické. Jako statické testy jsou označeny ty testy, během kterých nedochází ke spuštění kódu vyvíjeného systému (jedná se například o kontrolu a validaci specifikací a návrhů systému). Dynamické jsou naopak ty testy, během nichž je kód systému spouštěn (8). Na co se jednotlivé testy zaměřují a co obnáší jejich exekuce je popsáno v kapitole 3.2.3 Typy testů.



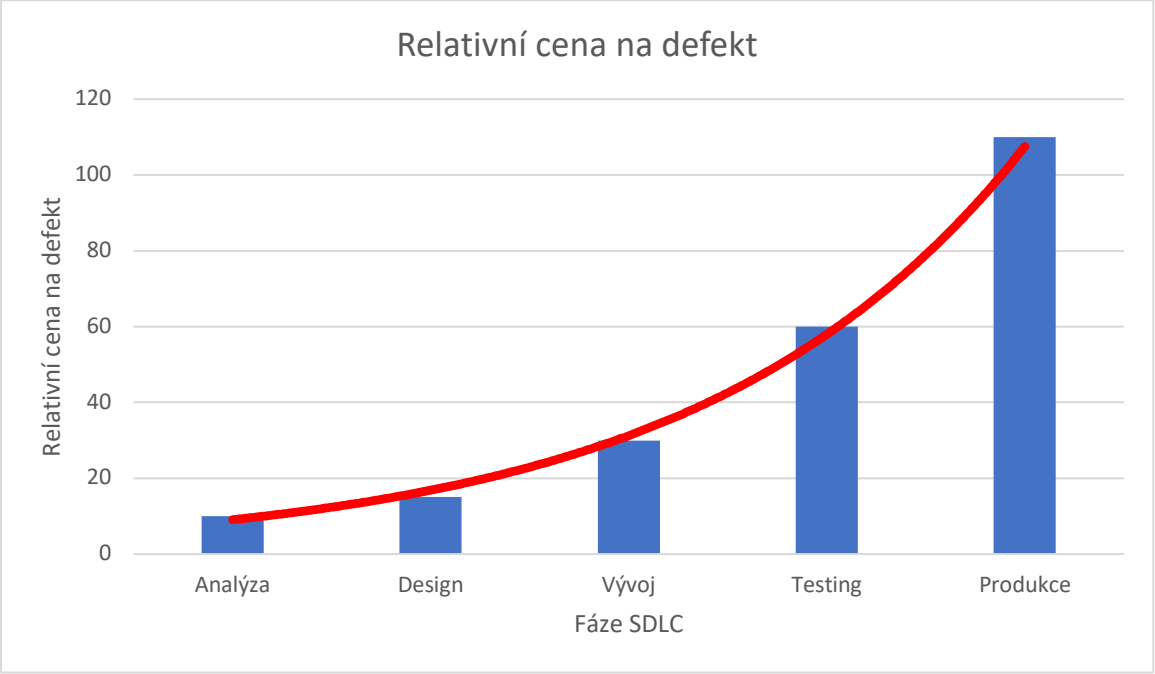
Obrázek 6: W-model (8)

3.2.1.2 Cena testingu v průběhu životního cyklu

Jako je uvedeno v (8) (kapitola 12) za obecně uznávaný model poměru jednotlivých testů v rámci životního cyklu systému se udává takzvaná „Testovací pyramida“, která je ve zjednodušené formě zobrazena níže na obrázku 7. Základní myšlenkou tohoto modelu je, věnovat větší podíl práce vytvoření a údržbě testů v nižších úrovních. Pyramida kopíruje typy testů z W-modelu – ty co jsou představeny jako první ve W-modelu, leží na spodu pyramidy. Odhalením a opravením defektů v nižších vrstvách zabráníme jejich přechodu do vyšších vrstev.

Obrázek 7: Testovací pyramida (8)

V souvislosti s tím se také objevuje pojem „Relativní cena na defekt“, ta znázorňuje, kolik stojí oprava zjištěného defektu v jednotlivých fázích SDLC. V čím vyšší fázi jsme, tím větší počet různých zaměstnanců by byl zapojen do vývoje systému a bylo by na něm odvedeno více práce. Tudíž, pokud je chyba již v samotném návrhu a vyvíjený systém by šel bez řádného testování až do poslední fáze (produkce), znamená to, že všechny činnosti mezi návrhem a uvedením do produkce vycházely z chybného návrhu. Konečné řešení tak bude špatné a po nalezení bugu bude muset celý proces začít znovu od nového návrhu. V případě, že by byla chyba odhalena již v některé nižší fázi, je do vývoje dáno méně úsilí, a tudíž i financí. Oprava tedy bude stát méně. Aby se předešlo defektům co nejdříve, je doporučováno testovat již od prvního návrhu systému.



Graf 1: Relativní cena na defekt v průběhu SDLC

Průběh růstu relativní ceny na defekt zobrazuje výše uvedený graf číslo 1. Jedná se o graf vytvořený přímo pro tuto práci pomocí dat dostupných z článku (9).

3.2.2 Základní artefakty testingu

Pojmem „Artefakt“ jsou v testingu označovány různé dokumenty, plány, reporty atd., které v průběhu vývoje softwaru vytvořili developeři a testeři. Tyto výstupy jsou sdíleny se zúčastněnými stranami projektu – například s členy testovacího i vývojového týmu, klienty, test manažerem a dalšími zapojenými jedinci. Artefakty jsou nedílnou součástí celého vývojového cyklu a transparentně zaznamenávají postup testování v rámci projektu. Základními artefakty testingu jsou:

- *Test scope* – oblast (funkce systému, aplikace, …), kterou se bude testování v rámci projektu zabývat.

- *Testovací strategie* – stručně popisuje, jaký přístup k testování bude u projektu zvolen, co vše bude součástí test scope. Obsahuje detailní časový a kapacitní plán, definuje cíle testingu na projektu a testingové role (jednotlivé role v rámci týmu testingu a jejich náplň práce jsou detailně popsané v kapitole 4.1.1), které budou do projektu zapojeny, obsahuje typy testů a seznam prostředí, na kterých budou testy vykonávány.

- *Testovací požadavky* – soupis požadavků na testování – jakou bude potřeba udělat analýzu, co vše musí pokrýt testy, co bude součástí UAT testu.

- *Test dashboard* – manažerský nástroj, sloužící k zobrazování a analyzování nejdůležitějších dat a metrik z projektu. Je to v podstatě virtuální nástěnka. Test dashboard zobrazuje statistiky a grafy týkající se protestovanosti prostředí, počtu aktivních bugů, nebo třeba aktivity testerů. Odpovědnost za nastavení dashboardu a jeho průběžnou aktualizaci tak, aby vyhovoval projektu, má většinou test lead, případně test manažer, nebo projektový manažer. Náplň těchto funkcí je uvedená v kapitole 4.1.1.

- *Test analýza* – podrobně definuje, co vše a do jaké hloubky má být otestováno, jaké typy testů budou na jednotlivé části použity a jaké jsou očekávané výsledky testů.

- *Test case (testovací scénář)* - specifikace vstupů, podmínek provedení, postupu testování a očekávaného výstupu definující jeden test. Test vždy simuluje určitou část logiky aplikace, nebo business procesu. Business proces je soubor navazujících činností, jejichž cílem je poskytnutí služby, nebo produktu klientovi.

- *Testovací sada* – slouží jako nástroj pro spojování souvisejících, nebo přímo navazujících testovacích scénářů. Napomáhá větší přehlednosti organizace testování na projektu.

- *Test plán* – sdružuje dohromady testovací sady a jednotlivé testovací scénáře, které mají být otestovány v rámci projektu. Dále v něm jsou popsány cíle testování na projektu a harmonogram průběhu testování.

- *Testovací data* – jsou definována v rámci testovací analýzy a jsou specifikovaná tak, aby jejich použitím během exekuce testovacího scénáře bylo dosaženo požadovaného výsledku testu.

- *Bug* – chyba, nebo defekt v programu či systému, která způsobí nesprávný nebo neočekávaný výsledek testu.

- *Checklist připravenosti* – kontrolní seznam požadavků, používaný k ověření, zda jsou splněny všechny předpoklady pro přechod do další fáze testingu.

- *Testovací prostředí* (test environment) – jako prostředí je v IT označován soubor operačního systému, databází, softwarových nástrojů a dalších podpůrných

elementů potřebných ke spuštění vyvíjeného systému. Je zažitým standardem, že firma provozuje vždy několik prostředí najednou. Na jednom prostředí běží produkční verze softwaru, na dalším probíhá vývoj nových funkcionalit. Toto prostředí také může sloužit jako testovací prostředí (prostředí, na kterém jsou prováděny testy), lepší je ovšem mít vývojářské a testovací prostředí oddělené, aby obě oddělení mohla pracovat nezávisle na sobě. Dále je také rozumné mít oddělené prostředí s kopií aktuální produkční verze pro případ, že by bylo potřeba obnovit ztracená data, nebo by došlo k výpadku produkčního prostředí. Neoddělitelnou součástí prostředí jsou i testovací data, bez nich by na prostředí nešlo řádně testovat. Mezi tato potřebná data patří například data sloužící pro přístup uživatelů k aplikaci, nebo přímo data, která jsou daným systémem zpracovávána.

Definice jednotlivých artefaktů jsou psány v souladu se zažitou terminologií v rámci řešené Pojišťovny. V prostředí jiné firmy proto mohou být chápana lehce odlišně.

3.2.3 Typy testů

- *Front-end testy* – testy zaměřující se na front-end systému. Front-end je ta část systému, kterou vidí koncový uživatel (například návštěvník webových stránek).
- *Back-end testy* – opak front-endových testů. Tyto testy se zaměřují na část systému, kterou běžný koncový uživatel nevidí (například kód aplikace).
- *End-to-end testování* – typ testů, během kterého je otestován celý business proces, a to od začátku do konce za podmínek simulujících provoz na produkci.
- *Funkční testy* – hodnotí, zda systém splňuje stanovené funkční požadavky. Funkční požadavky specifikují, jaké funkce má být řešený systém schopen vykonávat.
- *Test nové funkcionality* – testuje, jestli nově přidaná funkcionalita funguje v souladu s návrhem.
- *Regresní testy* – cílem regresního testování je zjistit, zda nově naimplementované funkcionality neplánovaně neovlivnily ty dosavadní.
- *SIT* – neboli systémové integrační testy se soustředí na ověření interakcí mezi propojenými systémy.
- *UAT* – je zkratkou pro „User Acceptance testing“, neboli Akceptační testy prováděné koncovými uživateli softwaru. Podle výsledku UAT se uživatel/klient rozhoduje, jestli vyvinutý produkt přijme, nebo ne. Úspěšné UAT je završené podepsáním *Akceptačního protokolu* – dokumentu, který stvrzuje, že je klient s produktem i jeho protestováním spokojen a přebírá si ho.
- *RAT* – celými slovy Release Acceptance Testing navazuje na UAT testy. Účelem těchto testů je ověřit funkčnost napříč všemi aplikacemi v celém releasu. Všechny

aplikace, které jsou aktuálním releasem ovlivněny, musí být otestovány. RAT je závěrečným testem po dokončení funkčního, systémového a regresního testování.

- *Check prostředí* – série rychlých, jednoduchých testů, která slouží jako kontrola prostředí, na kterém se chystá tester pracovat. Výsledek checku mu řekne, zdali je prostředí v testovatelném stavu, nebo ne.

- *Explorativní testy* – přístup k testování, při kterém tester nepostupuje podle předem daného scénáře, ale využívá svých znalostí, schopností a kreativity. Tester simultánně poznává systém, navrhuje test a zároveň ho i provádí.

- *Automatické testy* –testy, které jsou prováděny pomocí automatizačních nástrojů (více v kapitole 3.3.4) podle předem sepsaných scénářů. Exekuce testovacích scénářů probíhá automaticky na rozdíl od manuálních testů, kde musí scénáře provádět tester ručně.

- *Zátěžové testy* – zátěžové neboli performance testy ověřují jak systém, či jeho komponenty fungují pod různými úrovněmi zatížení. Testy se obvykle zaměřují na rychlost, stabilitu a responzivnost systému při zatížení. Zátěžové testy jsou ideální kandidátem na automatizaci (viz kapitola 3.3.2).

Definice jednotlivých typů testů byly převzaty z ISTQB glosáře (9) a poupraveny tak, aby odpovídaly firemní zvyklostem. ISTQB (International Software Testing Qualification Board) Glossary, neboli glosář mezinárodní rady pro testování softwaru, je světově uznávaný standardizovaný slovník pojmů používaný v testování softwaru. Glosář je pravidelně aktualizován a poskytuje konzistentní definice pojmů z oboru.

3.2.4 Další pojmy spojené s testingem a vývojem

- *Plán kapacity* – plán používaný ke správě zdrojů potřebných k realizaci řešeného projektu. Plán by měl obsahovat vícero scénářů pro různé obchodní předpovědi a možnosti s odhady potřebných nákladů, lidských a dalších zdrojů pro dosažení stanovené úrovně služeb (10).

- *Deployment* – neboli nasazení je proces nasazení kódu na konkrétní (například testovací nebo produkční) prostředí.

- *Build* – název pro zprovozněný systém, který vývojáři předávají testingu ke kontrole. Jakmile je build prohlášen za úspěšně otestovaný, je předán koncovým uživatelům jako release.

- *Release* – proces, kdy je koncovým uživatelům/zákazníkům umožněno začít používat novou verzi, či funkcionalitu vyvíjeného systému.

- *Pull request* – požadavek vývojáře na kontrolu a integraci nově napsaného kódu do systému. Jakmile vývojář napíše svůj kód, podá pull request, ten je pak automaticky vyhodnocen systémem a developerův kód projde testováním (více

v kapitolách 4.2.1.2 až 4.2.1.4). V případě, že kód projde testy bez problému, je zaintegrován, pokud ne, vrací se na opravu vývojáři.

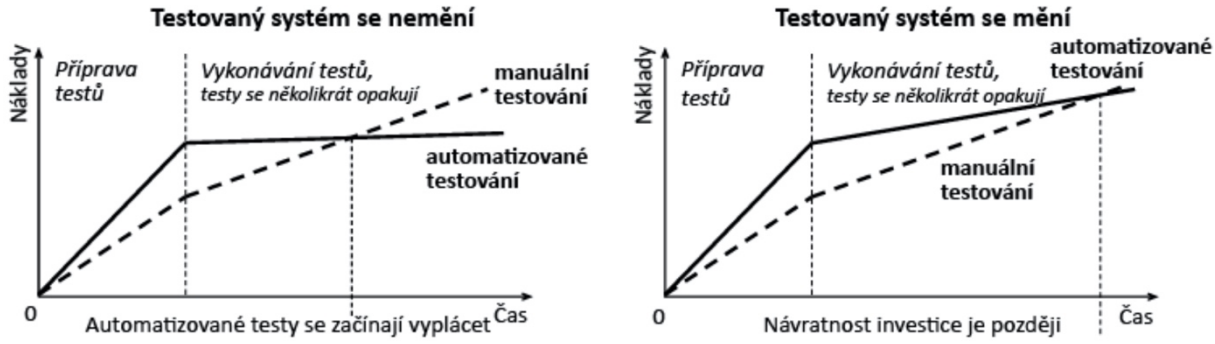
– *Uživatelské rozhraní* – součásti sytému, které uživateli poskytují informace a ovládací prvky umožňující interakci se systémem.

3.3 Automatizace testování

Podle toho, zda jsou testy prováděny člověkem nebo softwarem, rozlišujeme testování na manuální a automatizované. Pokud test vyžaduje lidské ohodnocení a úsudek nebo rozličné přístupy, které není třeba zaznamenat a pravidelně opakovat, je vhodnější manuální testování – test je vykonán přímo samotným testerem. Pro opakované spouštění velkého množství testů nebo v případě testu, kdy musíme generovat velké množství syntetických dat, stejně tak jako pro zátěžové testování je výhodnější použít automatizované testy.

3.3.1 Kritéria použití automatizace testů

Protože automatizovaná exekuce testu stojí mnohem méně úsilí než manuálně prováděné testování, se od určitého počtu opakování vyplatí test automatizovat. Úspora času potřebného pro alternativní manuální testování po jistém počtu opakování převáží náklady na vytvoření automatizovaného testu (8). Pokud se testovaný systém v průběhu času nemění, dojde k návratnosti investice o poznání dříve než v případě, kdy průběžně dochází ke změnám systému. Porovnání těchto dvou případů nabízí obrázek 8:



Obrázek 8: Návratnost investice na automatizaci v čase (8)

Nejdůležitějším hlediskem pro, či proti automatizaci proto většinou bývá návratnost investice – hodnocení toho, zdali bude tvorba a provoz automatických testů levnější, či dražší než manuální testy vykonávané testery. Mezi další významná kritéria při rozhodování patří (8):

– *Jednoznačnost výsledku* – jestliže není výsledek testu jednoznačný a snadno interpretovatelný, program si s ním jen obtížně poradí. Proto je potřeba zvážit, jestli je automatizace testů v takových případech vhodná.

- *Četnost opakování testovacího scénáře* – vyplatí se automatizovat jen ty testy, které se nespouští jen jednou, ale běží opakovaně – například regresní testy nebo check prostředí.

- *Stabilita systému* – čím méně se sytém mění, tím menší je i potřeba aktualizovat testy. Pokud tedy u systému nejsou v budoucnu plánovány žádné výraznější změny, nebo alespoň nejsou pravidelné, vyplatí se automatizovat.

- *Závažnost testu pro business proces* – jestliže je testovaná funkcionalita klíčová pro hlavní produkční činnosti, její selhání by vedlo k vážnému poškození podnikání. Proto je důležité tyto klíčové oblasti pravidelně testovat.

- *Počet platforem* – jestliže potřebujeme aplikaci otestovat na vícero operačních systémech, či ve vícero webových prohlížečích, je vhodné testy automatizovat.

- *Množství použitých dat* – automatizováním testů, pro které je potřeba vygenerovat velké množství dat (například vyplňování dlouhých formulářů) se zásadně usnadní práce testingového týmu. To samé platí pro skupiny testů, které mají stejný průběh a liší se jen v zadávaných datech. Díky automatizovaným testům lze otestovat mnohem větší množství kombinací vstupních dat, než by bylo reálné pouze manuálními testy.

- *Délka, kterou test zabere* – automatizování testů, které zaberou více času, nebo musí být pouštěny přes noc, opět usnadníme testerům práci.

- *Náročnost automatizace* – u méně důležitých a málo opakovaných testů je potřeba také zvážit, jak obtížné bude případně testovací scénář automatizovat.

- *Počet přerušení procesu* – jestliže je proces často přerušován, nebo je potřeba přepnout mezi jednotlivými systémy, je automatizace obtížná z důvodu potřeby návaznosti dat napříč systémy. A vzhledem k zapojení více systémů bude proces také více náchylný ke změnám.

Zatímco se výše uvedený výčet zaobíral spíše případy, kdy se automatizace testů doporučuje, lze stanovit i seznam situací, kdy to vhodné není:

- Test je potřeba spustit jen jednou – výjimkou může být případ, kdy je třeba provést test s velkým množstvím testovacích dat. I kdyby byl takovýto test spuštěn pouze jednou, může mít smysl ho automatizovat.

- Testování uživatelského prostředí – pouze reálný uživatel dokáže posoudit, jak snadné a intuitivní používání softwaru je.

- Testy, které je potřeba provést co nejdříve – v případech (například vyvinutí nové funkce do aplikace), kdy je třeba rychlé zpětné vazby by tvorba automatizovaných testů jen zdržovala.

- Explorativní testy – testy plně spoléhající na testerovu intuici a znalosti nelze automatizovat.

- Testovací scénáře, které nelze zautomatizovat stoprocentně, by měly být automatizovány jen v případě, že i pouhá částečná automatizace značně ušetří čas.

Jak se píše v (8) (kapitola 12.2), již během procesu samotné tvorby automatizovaného testu dochází vlastně k testování. Na případné defekty v systému se tudíž přijde již během vytváření testu. Hlavní oblastí, která je nejčastěji automatizována jsou proto regresní testy. Ty se pravidelně spouští, aby ověřily, zda se v dané oblasti, dříve bez defektů, neobjevil nějaký bug.

3.3.2 Typy automatických testů

- *Unit testy* – česky označované jako jednotkové testy, jsou procesem podrobného testování co nejmenších možných částí (proto unit/jednotkové testy) kódu. Každý z unit testů musí být samostatně spustitelný a testy musí být na sobě nezávislé. Jejich cílem je zjistit, jestli kód dělá to, co programátor zamýšlel a jestli funguje správně. Unit testy si píší a udržují vývojáři přímo pro svůj kód (proto jsou někdy také označovány jako vývojářské testy). Vyhodnocování výsledků obvykle provádí buď sami vývojáři nebo případně pověřený člen testingu. Význam unit testů tkví v tom, že odhalí chyby hned na začátku vývojového procesu, takže jejich oprava je v tomto okamžiku nejméně nákladná (8).

- *Smoke testy* – nebo také „Build verification testy". Jejich primárním smyslem není hledat defekty, ale ověřit, že klíčové funkcionality testovaného systému fungují, systém je stabilní a vhodný pro detailnější testy, či zpřístupnění uživatelům. Smoke testy se používají především po kompilaci nového kódu, po nasazení systému do produkce, nebo po přesunu na nové prostředí. Po svém spuštění Smoke testy rychle odhalí, zdali nedošlo k chybám v konfiguraci softwaru (8).

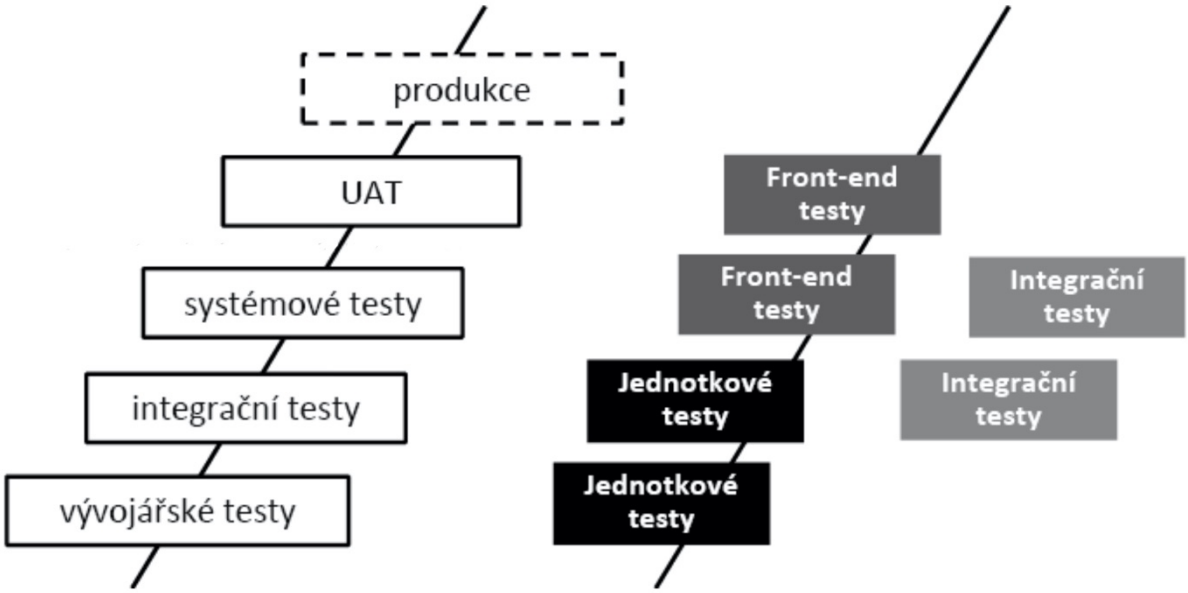
- *Integrační testy* – velká část dnešních aplikací se skládá z modulů nebo většího počtu spolupracujících systémů, které jsou vzájemně propojené integračním rozhraním. Konkrétní části business procesu jsou obvykle implementovány těmito jednotlivými moduly, či systémy. Pokud přecházíme z jedné části procesu do druhé, přes integrační rozhraní se přenášejí odpovídající data. Právě tuto komunikaci pak prověřují integrační testy (8). Integrační testy mohou být jak jednoduché a automatizované, tak i komplexní a manuální, záleží na situaci a testovaném systému.

- *Performance testy* – jak již bylo napsáno v kapitole 3.2.3, performance testy zkouší, jak daný systém obstojí pod různými úrovněmi zátěže. Performance testy jsou většinou automatizované především z důvodů snazší simulace zátěže systému oproti manuálnímu testerovi. Například pro test stability aplikace při velkém počtu

uživatelů přihlášených najednou, by se v případě manuálního testování muselo opravdu sehnat tolik lidí, kolik přihlášených test požaduje, aby vystupovali jako uživatelé. Oproti tomu použití automatizovaného testu umožní simulaci libovolného počtu přihlášených uživatelů, aniž by bylo třeba zapojovat reálné lidi. Další výhodou automatizovaných performance testů je také to, že je lze pustit kdykoliv je třeba bez nutnosti čekání na reálné uživatele.

3.3.3 Automatizované testy ve W-modelu

Pro lepší chápání souvislostí je dobré si uvést základní typy automatizovaných testů v relaci k W-modelu životního cyklu vývoje systému, představeném v kapitole 3.2.1.1. Vztah mezi W-modelem a automatizovanými testy je naznačen na obrázku 9. Bílé obdélníky na levé straně obrázku 9 reprezentují pravé rameno „V" testovacích aktivit z W-modelu na obrázku 6. Tmavé obdélníky v pravé části pak zastupují základní typy automatizovaných testů, které lze pro jednotlivé úrovně testování využít. Vyšší pozice v modelu znamená, že je k dispozici testovaný systém v ucelenějším stavu. Jednotkové testy jsou prováděny vývojáři na nejnižší úrovni nad nově psaným kódem. Automatizované integrační testy jsou používány v situaci, kdy je už nový kód zapojený do systému a je potřeba otestovat integraci s okolními systémy. Pro spuštění automatizovaných front-end testů je potřeba, aby byl celý systém již zprovozněný a měl fungující uživatelské rozhraní.



Obrázek 9: Automatizované testy v rámci W-modelu (8)

3.3.4 Tvorba automatizovaných testů

Testy na téměř všech úrovních W-modelu lze automatizovat. V závislosti na tom, o jaký test se jedná, jsou k dispozici různé automatizační nástroje, které lze použít. Automatizační nástroje (anglicky automation tools) jsou samostatným softwarem, pomocí něhož lze tvořit a později i spouštět automatizované testy (12). Samotné testy jsou tvořeny tak, že pověřený developer nebo tester přepíše požadovaný testovací scénář do automatizačního nástroje pomocí syntaxe (pravidla pro zápis výrazů daného programovacího softwaru), který používá automatizační nástroj. Následuje stručný výčet nejpoužívanějších automatizačních nástrojů (12):

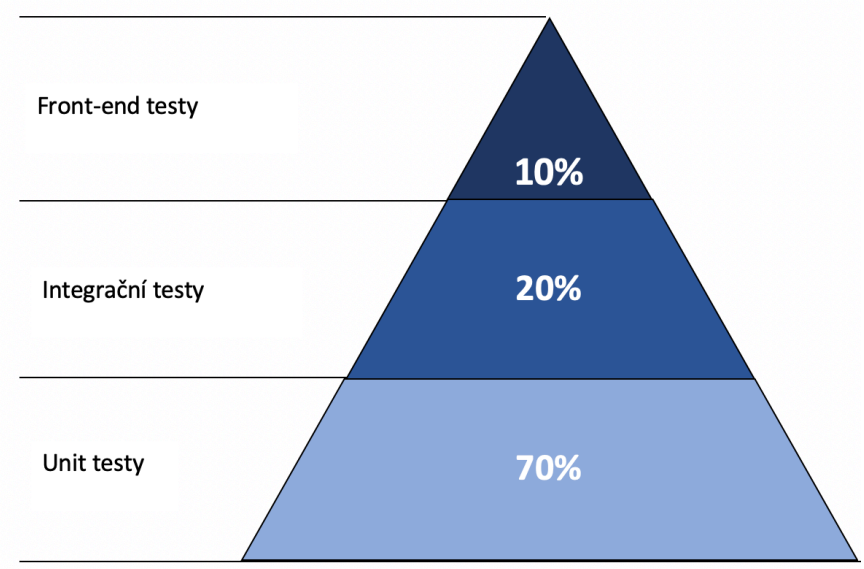
- *Selenium* – multiplatformní (Windows, Mac, Linux, …) nástroj umožňující tvorbu testů webových aplikací ve vícero programovacích jazycích (Java, Python, Perl, Ruby a další). Selenium také nabízí funkci nahrání manuálního testu provedeného testerem. Selenium pak tento test zautomatizuje.

- *SpecFlow* – umožňuje psát testy prostou angličtinou za použití jednoduché syntaxe. Díky tomu je tvorba automatizovaných testů přístupná i pro uživatele, kteří neovládají programovací jazyky. SpecFlow se nejvíce používá pro testování business logiky aplikace.

- *SoapUI* – open-source nástroj pro testování webových rozhraní a automatizaci testů. Software existuje v základní volné a komerční „Pro" verzi. Pro verze umožňuje kompletní automatizaci end-to-end testů a psaní testů bez nutnosti kódování – testy jsou zadávány pomocí předem nadefinovaných výrazů a podmínek.

- *JUnit* – nástroj pro tvorbu automatizovaných unit testů pro kód psaný v programovacím jazyce Java. JUnit umí automaticky spouštět série navazujících testů.

Co se rozložení množství automatizovaných testů týče, tak i zde je doporučováno (8) držet se modelu testovací pyramidy s přibližným rozdělením 70 ku 20 ku 10 procentům podílu jednotlivých typů testů na celku. Největší důraz je kladen na unit testy, které testují samotný kód, hned po tom, co ho developer napíše. Pokud je defekt odhalen již na začátku procesu je jeho oprava rychlá a nezatěžuje další části vývojového cyklu (testing a analýzu). Tyto testy odhalí nejvíce chyb. Uprostřed jsou integrační testy a nejmenší množství by mělo být automatizovaných front-end testů. Testovací pyramida i s doporučeným procentuálním rozložením objemu testů je znázorněna na obrázku 10.



Obrázek 10: rozložení automatizovaných testů

3.4 Tvorba podnikových procesů

Řepa (13) definuje podnikový proces jako souhrn činností, které postupně přetvoří vstupy na výstupy. Koncovým uživatelem procesu může být například zákazník, jiné oddělení v podniku, nebo další proces. Pro to, aby byl proces co nejefektivnější, je vhodné do jeho průběhu koncové uživatele zapojit. Ti jsou totiž nejlepším zdrojem zpětné vazby. Každý proces potřebuje k svému průběhu zdroje, a to jak lidské, tak hmotné, nehmotné, i finanční.

3.4.1 Návrh procesů

Podle Hammera (14) je prvním krokem tvorby efektivních firemních procesů volba *vlastníka procesu*. Vlastník procesu je osobou odpovědnou za návrh, implementaci a dodržování nového procesu. Dále také odpovídá za jeho kontinuální vývoj. Samotný průběh návrhu nových procesů pak lze podle Hammera rozdělit do čtyř etap:

1. *Mobilizace* – během této etapy je sestaven tým, který bude nové procesy navrhovat. Ideální tým by měl mít mezi 5 až 9 členy, aby byla zajištěna rozmanitost, ale aby se zároveň dal tým snadno řídit. Členové týmu by měli být mixem členů z různých oddělení, která jsou již v současném procesu zapojena a těch, která stojí mimo něj.

2. *Diagnostika* – neboli podrobná analýza současných procesů. Během analýzy by mělo rovnou dojít i k identifikaci problémových míst současných procesů. Pro snadno čitelné grafické znázornění firemních procesů se nejčastěji používá standard BPMN (Business Process Modelling Notation). Tento standard bude podrobněji popsán v kapitole 3.4.2 Modelování procesů.

3. *Nové rozvržení* – v rámci této etapy dochází již k samotnému návrhu nových procesů. Nový proces je vhodné opět graficky znázornit. Doporučuje se na konec této etapy zařadit simulaci průběhu nového procesu. Ta obnáší společnou diskuzi nově navrženého procesu se zástupci všech oddělení, jichž se změny v procesu dotknou.

4. *Změna* – poslední etapa obnáší implementaci navržené podoby nového procesu. Především během této etapy je důležitá otevřená komunikace se zbytkem společnosti, aby všichni věděli, jak je změna v procesech zasáhne.

3.4.2 Modelování procesů

Procesy jsou modelovány pomocí diagramů. Jako standard notace[3] pro grafické zobrazení diagramů se v současnosti užívá notace BPMN (Business Process Modelling Notation). BPMN diagramy popisují procesy pomocí souboru symbolů (15). Symboly lze rozdělit do několika kategorií zobrazených v tabulce 1.

Název	Ukázka symbolu	Popis
Událost		Vnější podnět, který vzniká mimo proces, nebo proces ukončuje. Bývá zobrazen kolečkem, nebo elipsou s popiskem.
Činnost		Základní element procesu. Popisuje, co za aktivitu musí být vykonáno, aby proces mohl pokročit dál. Symbol "obráceného domečku" reprezentuje činnost, obsahující v sobě další proces.
Výstup		Tento symbol reprezentuje výstup procesu. Výstupem bývá obvykle dokument, nebo produkt.
Brána		Brána indikuje místa, kde dochází k rozdělení, nebo spojení cest. Cesta, kterou se proces vydá záleží na podmínce zobrazené na bráně.
Spojení		Ukazuje návaznosti mezi jednotlivými objekty diagramu.
Plavecké dráhy		Tento objekt slouží k zobrazení aktivit patřící do jednoho proudu - například činnosti vykonávané jedním pracovníkem, nebo oddělením. Více drah se združuje do "bazénu".

Tabulka 1: Symboly BPMN

Diagramy vypracované pro tuto práci vycházejí ze standartu BPMN a jsou upraveny dle zvyklostí v řešené Pojišťovně. A to proto, aby co nejvíce odpovídaly diagramům, které se v Pojišťovně již používají.

[3] Notace je soubor symbolů zastupující různé entity (například hudební noty) pro usnadnění komunikace v rámci daného oboru.

4 Analýza současné situace

Cílem této kapitoly je popsat současný stav v Pojišťovně. Níže uvedené podkapitoly shrnují nejzásadnější poznatky o současném stavu testingu, jeho úloze a zařazení v rámci životního cyklu vývoje softwaru. Dále je analyzován současný přístup k automatizaci testování a jeho používání.

4.1 Testingové oddělení

Hlavním úkolem oddělení testingu je ověřit, zda vyvíjený software, či jeho komponenty, odpovídají požadovaným charakteristikám. Testing reportuje svá zjištění relevantním zúčastněným stranám. Pracovníkům v rámci oddělení jsou přiřazeny následující role:

- Test manažer
- Test lead
- Test analytik
- Tester
- UAT koordinátor

Náplň práce jednotlivých rolí a jejich příslušné odpovědnost jsou detailně popsány v následující kapitole.

4.1.1 Role v týmu

Test manažer

- Je odpovědný za sestavení a vedení testovacího týmu
- Komunikuje s aplikačním managementem a vedením společnosti, relevantní informace dále šíří v týmu
- Definuje rozsah a strategii testování v rámci jednotlivých projektů/releasů,
- Zodpovídá za alokaci zdrojů (software, hardware) a plánování kapacit
- Monitoruje a vyhodnocuje průběh testování a jeho kvalitu podle předem definovaných metrik
- Ověřuje, zda jsou naplněna vstupní/výstupní kritéria projektu
- Hodnotí ukončené projekty a aktualizuje firemní testingovou metodologii
- Je odpovědný za zaučení nových členů týmu
- Je vlastníkem procesu automatizace testování a odpovídá za něj

Test lead (leader)

- Má na starost tvorbu detailního test plánu pro daný projekt
- Společně s test manažerem se podílí na tvorbě testovací strategie
- Plánuje regresní testování – tvoří odhady pracnosti a nákladů pro testování, získávání zdrojů, definování testovacích fází a správu nalezených defektů
- Definuje požadavky na testovací data a požadované HW a SW nástroje
- Podílí se na tvorbě test analýzy
- Spravuje defekty zaznamenané během testování systému (zkontroluje defekty před jejich přiřazením developerům na opravu)
- Průběžně reportuje o aktuálním stavu exekuce testování
- Ověřuje, zda jsou naplněna vstupní/výstupní kritéria projektu
- Určuje prioritu aktivních defektů

Test analytik

- Společně s test manažerem a test leaderem se podílí na tvorbě testovací strategie
- Hlavní část jeho pracovní náplně tvoří především tvorba test analýz a testovacích scénářů - ty vypracovává na základě vstupů od analytiků a businessu
- Během tvorby testovacích scénářů formuluje požadavky na potřebná testovací data a HW a SW nástroje – ve spolupráci s test leaderem
- Slouží jako podpora testerů a pomáhá při zácviku nových členů týmu
- Test analytik má také stejnou roli a stejné povinnosti jako tester

Tester

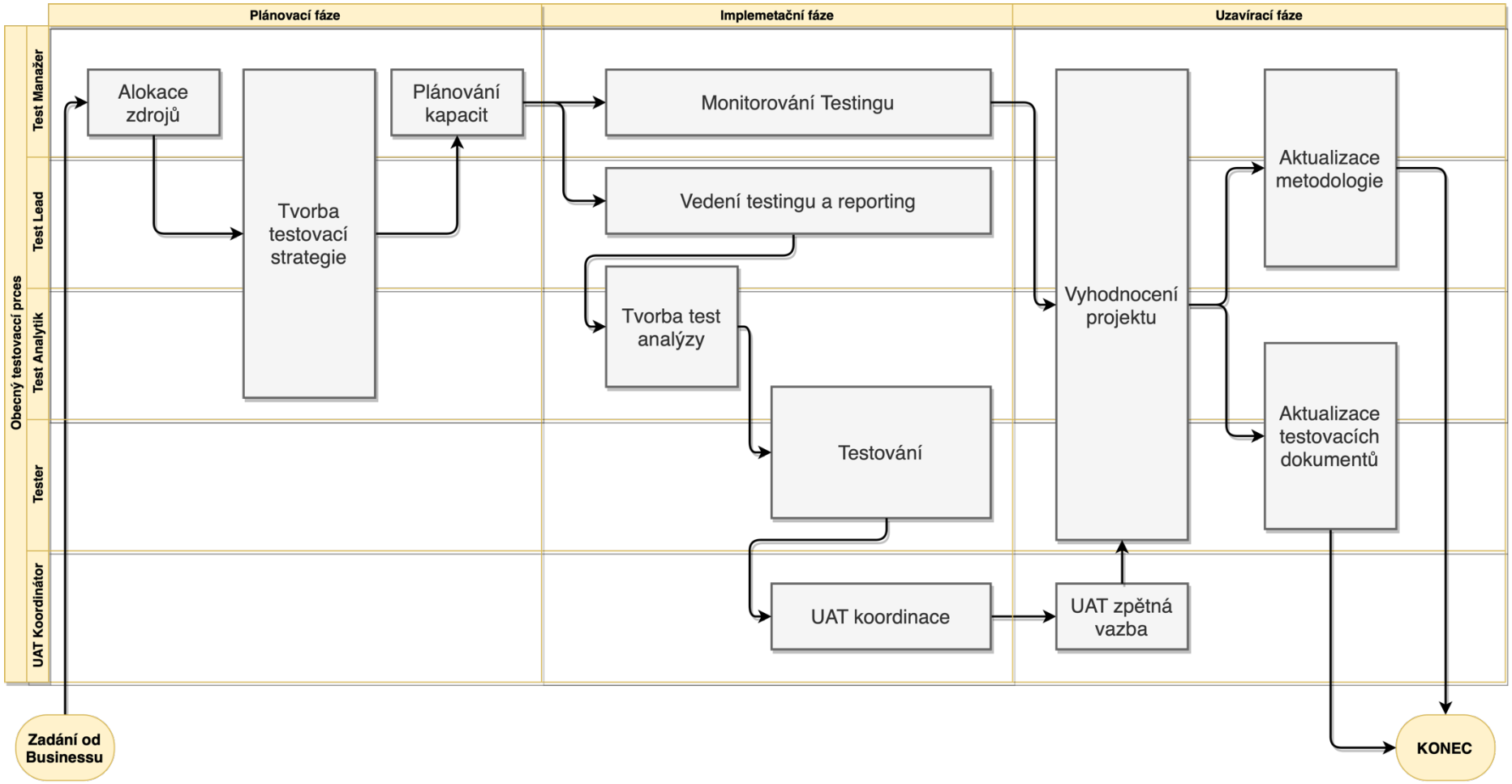
- Provádí detailní testovací analýzu pro danou oblast a fázi testování
- Identifikuje testovací data potřebná pro provedení testů
- Sepisuje testovací scénáře pro všechny testy, které jsou součástí testovací fáze projektu
- Provádí kontrolu a exekuci přidělených testovacích scénářů
- Poskytuje odhady pracnosti, rizik a dalších předpokladů jako jeden z podkladů pro tvorbu detailního test plánu
- Verifikuje a zaznamenává výsledky testování
- Zaznamenává a sleduje defekty objevené v průběhu testování
- Vykonává regresní testy pokaždé, když je nasazen nový release

UAT koordinátor

- Definuje ve spolupráci s obchodními analytiky rozsah UAT testů
- Vytváří a udržuje testovací plán pro UAT fázi
- Koordinuje přípravu testovacích dat a prostředí pro UAT
- Poskytuje podporu při vyhodnocení výsledků testů
- Monitoruje a hlásí pokrok UAT
- Podporuje akceptační fáze projektů / releasů
- Shromažďuje zpětnou vazbu UAT ze všech zemí, kde dochází k nasazení testovaných funkcionalit

Role test leadera se obvykle předává mezi test analytiky po jednotlivých releasech. To samé platí i pro roli UAT koordinátora. Testeři jsou ve společnosti vedeni ke zvyšování své kvalifikace, aby se postupem času mohli stát test analytiky.

Další rolí, která zasahuje do fungování testingového oddělení, je role projektového manažera. Projektový manažer má na starost plánování, exekuci a uzavření daného projektu. Testing přichází do styku s projektovými manažery v případech, kdy je vyvíjen nějaký úplně nový systém. Při práci na standardních aktualizacích již zaběhlých systémů není účast projektového manažera třeba.



Obrázek 11: činnosti v rámci testovacího procesu

Diagram na obrázku 11 zobrazuje vztah mezi fázemi testovacího procesu a zapojením jednotlivých rolí v rámci oddělení testingu. Role v rámci oddělení a jejich odpovědnosti jsou rozepsány nad diagramem. Popisu fází testovacího procesu a tomu, jaké činnosti obsahují, se věnuje následující kapitola.

4.1.2 Fáze testovacího procesu

Základní testovací proces, který aplikuje řešená společnost se skládá ze tří fází:

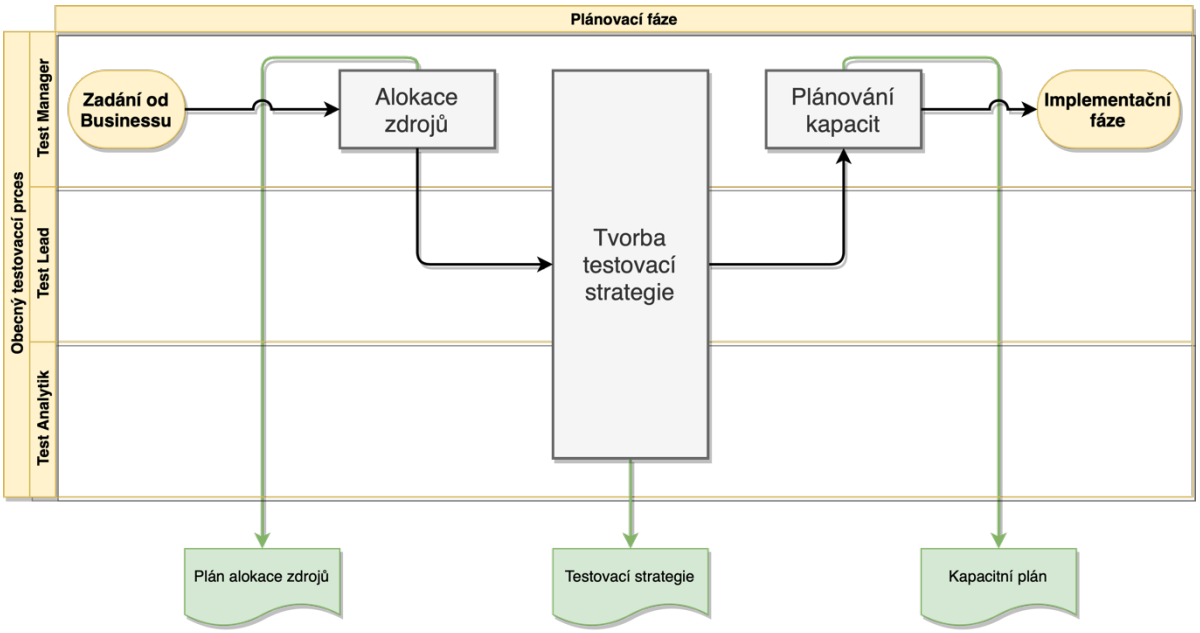
1. Plánovací fáze
2. Implementační fáze
3. Uzavírací fáze

Jednotlivé fáze jsou podrobně představeny a znázorněny pomocí diagramů v následujících podkapitolách.

4.1.2.1 Plánovací fáze

Cílem plánovací fáze je definovat přístup k jednotlivým projektovým činnostem týkajících se testování a přidělit vhodné zdroje zapojeným testovacím rolím v souladu s plánovanými aktivitami. Během této fáze obsadí Test manažer potřebné role na projektu z řad členů testingového oddělení. Poté společně s Test leaderem a Test analytiky vytvoří testovací strategii.

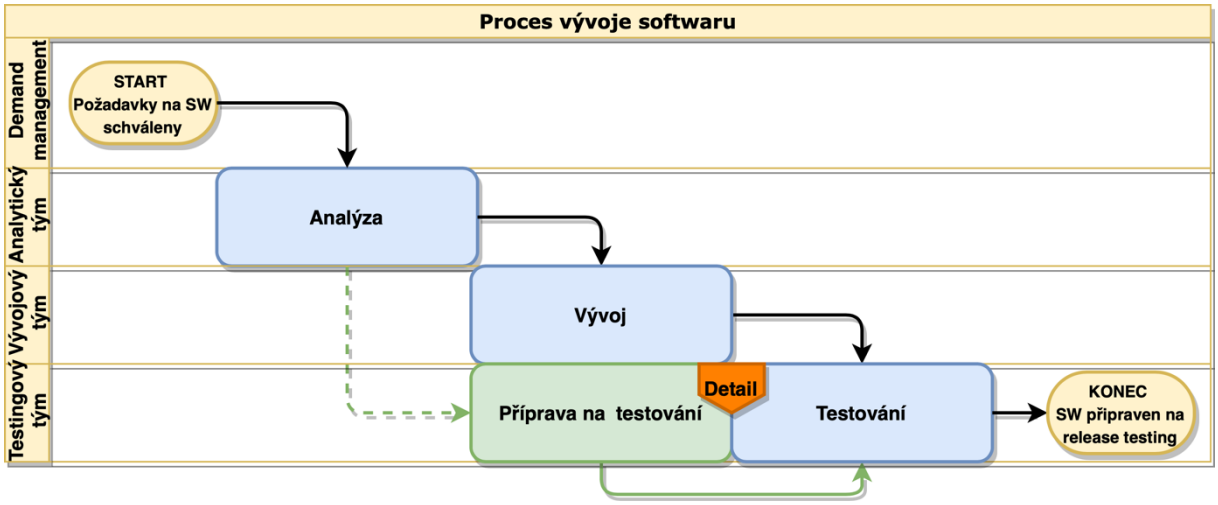
Jak je zřejmé z obrázku 12, celá fáze začíná zadáním požadavku na nový software/novou funkcionalitu pro již existující software od Businessu. V kontextu společnosti je Business chápán jako ta část pojišťovny, která je zodpovědná za návrh, vývoj a provoz pojišťovacích produktů, které přinášejí firmě zisky. Požadavek tedy může vzniknout například v obchodním oddělení, v oddělení likvidace, na účtárně, nebo třeba na call centru. Obrázek 12 zachycuje detail plánovací fáze z Obrázku 11. Zeleně jsou na něm zobrazeny výstupy procesu – dokumentace vznikající v souvislosti s novým projektem.



Obrázek12: Diagram průběhu plánovací fáze

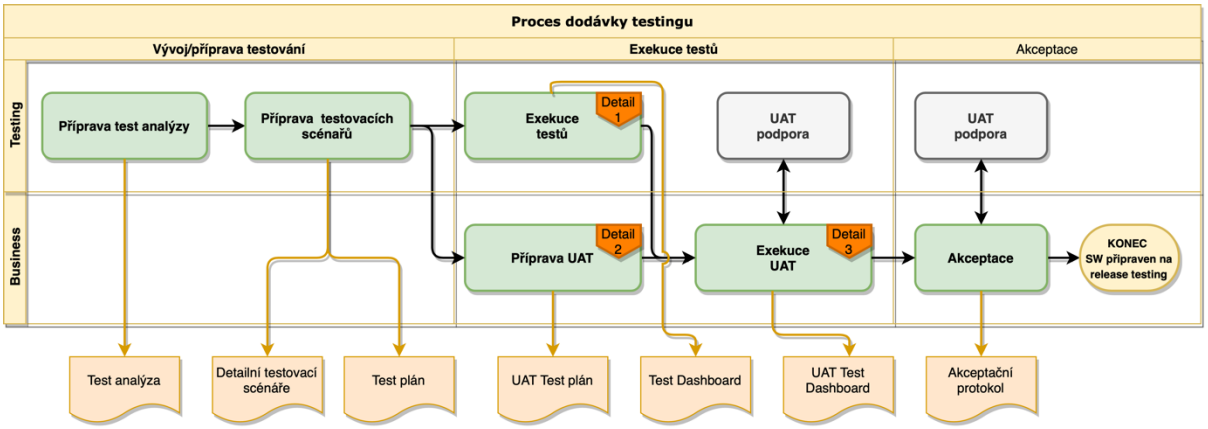
4.1.2.2 Implementační fáze

Během implementační fáze se odehrává většina testovacích aktivit. Probíhá zde detailní analýza testovaného rozsahu a identifikace požadavků na testování, jsou připravovány podrobné testovací scénáře, potřebná testovací data a prostředí. V implementační fázi také dochází k samotné exekuci a vyhodnocení připravených scénářů. Případné nalezené bugy jsou opraveny a znovu testovány. Fáze končí v okamžiku, kdy je software připraven na nasazení do produkce.



Obrázek 13: Diagram průběhu procesu vývoje softwaru

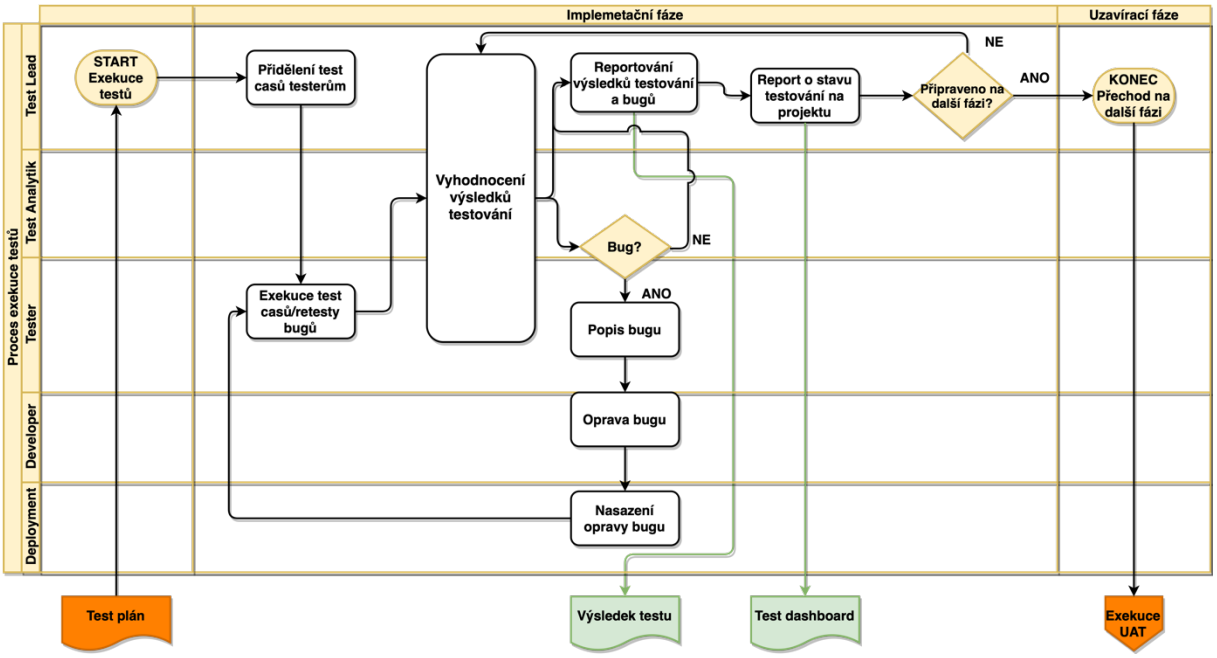
Výše uvedený diagram (Obrázek 13) znázorňuje, jakou pozici zaujímají činnosti Příprava na testování a Testování v rámci procesu vývoje softwaru. Činnosti znázorněné na diagramu probíhají v rámci implementační fáze zobrazené na Obrázku 11. Detail průběhu Přípravy na testování a Testování je podrobněji rozveden na následujícím diagramu – Obrázku 14.



Obrázek 14: Detail procesu dodávky testingu v rámci vývoje softwaru

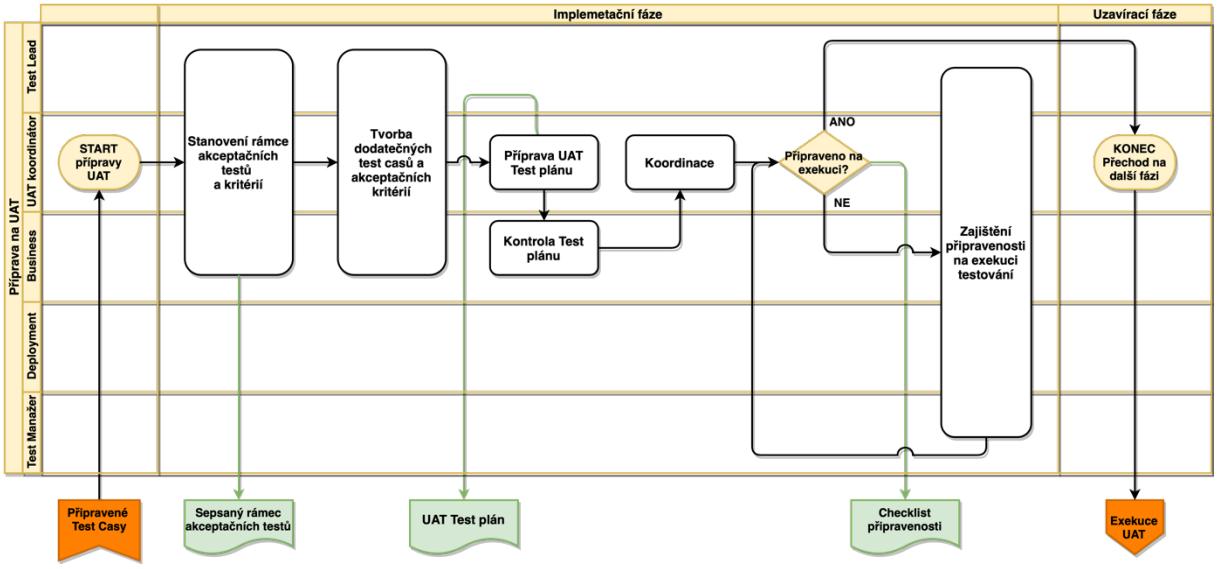
Proces dodávky testingu je rozdělen do tří částí. V první, vývoji a přípravě testování, jsou vytvořeny dokumenty test analýzy, test plán a detailní testovací scénáře. Jakmile jsou dokumenty připraveny, přechází proces do druhé části, kde se větví do dvou činností, které probíhají paralelně. Testing provádí exekuci testů podle připravených testovacích

scénářů (podrobně popsáno na Obrázku 15). Souběžně probíhá příprava na UAT (Obrázek 16). Po dokončení těchto činností probíhá exekuce UAT (detail procesu je zobrazen na Obrázku 17) ze strany Businessu s podporou testingu. Proces končí podpisem akceptačního protokolu zástupci Businessu v rámci poslední části procesu.



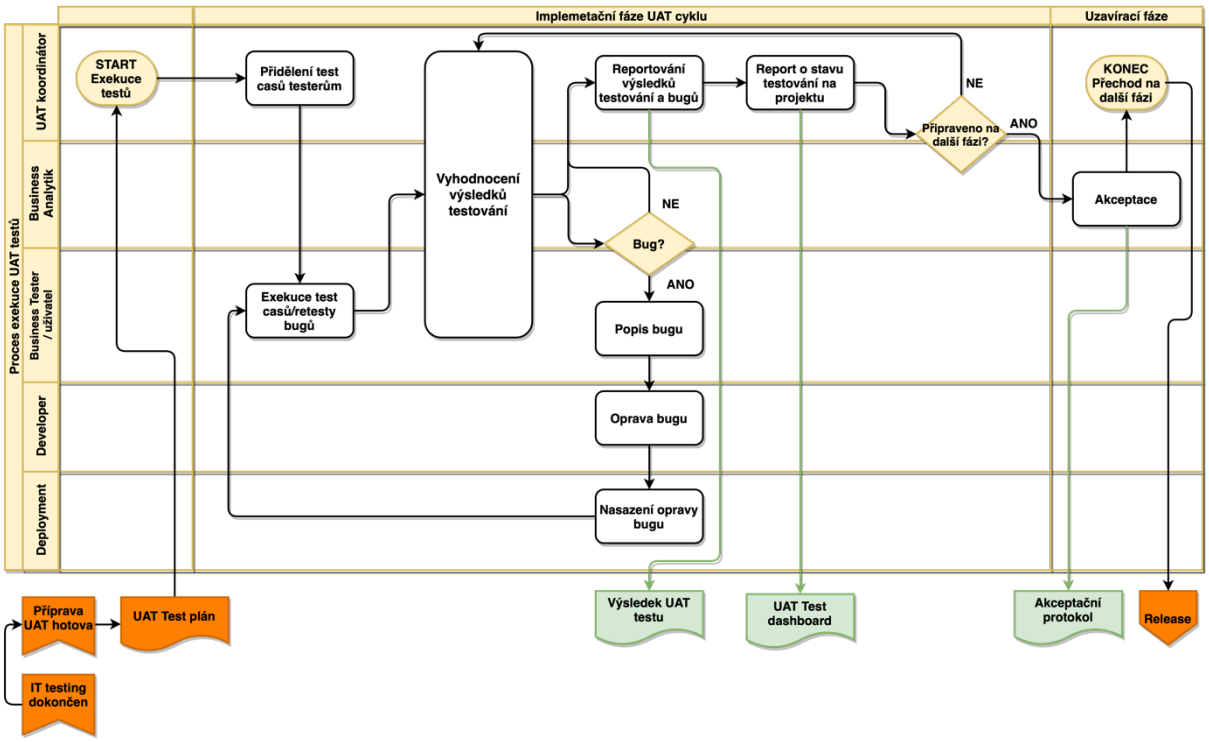
Obrázek 15: Proces IT testování v rámci implementační fáze

Obrázek 15 ukazuje průběh procesu exekuce testů. Zde se do procesu kromě testovacího oddělení zapojují také developeři a deployment. Testeři provedou testy dle detailních testovacích scénářů a zaznamenají jejich výsledky. Případné nalezené bugy jsou popsány a předány na opravu developerům. Opravu pak nasadí deployment, po čemž dojde k opakování testu, aby mohl být bug uzavřen jako opravený. Souhrnné výsledky testování na projektu jsou pak shromažďovány na Test dashboardu. V okamžiku, kdy je aplikace vyhodnocena jako dostatečně protestovaná a připravená na další fázi, je předána businessu na exekuci UAT (Obrázek 17).



Obrázek 16: Proces přípravy na UAT

Paralelně s IT testováním prováděným testingovým oddělením probíhá příprava na UAT (obrázek 16). Proces začíná po vytvoření testovacích scénářů a zahrnuje činnosti zajišťující, že uživatelé a zástupci businessu pověření UAT budou schopni vyvíjenou aplikaci/funkci testovat. Výstupy procesu jsou rámec akceptačních testů a kritérií, UAT test plán a Checklist připravenosti na UAT. Jakmile je vše připraveno, přípravný proces končí a přechází v exekuci UAT.



Obrázek 17: Proces exekuce UAT

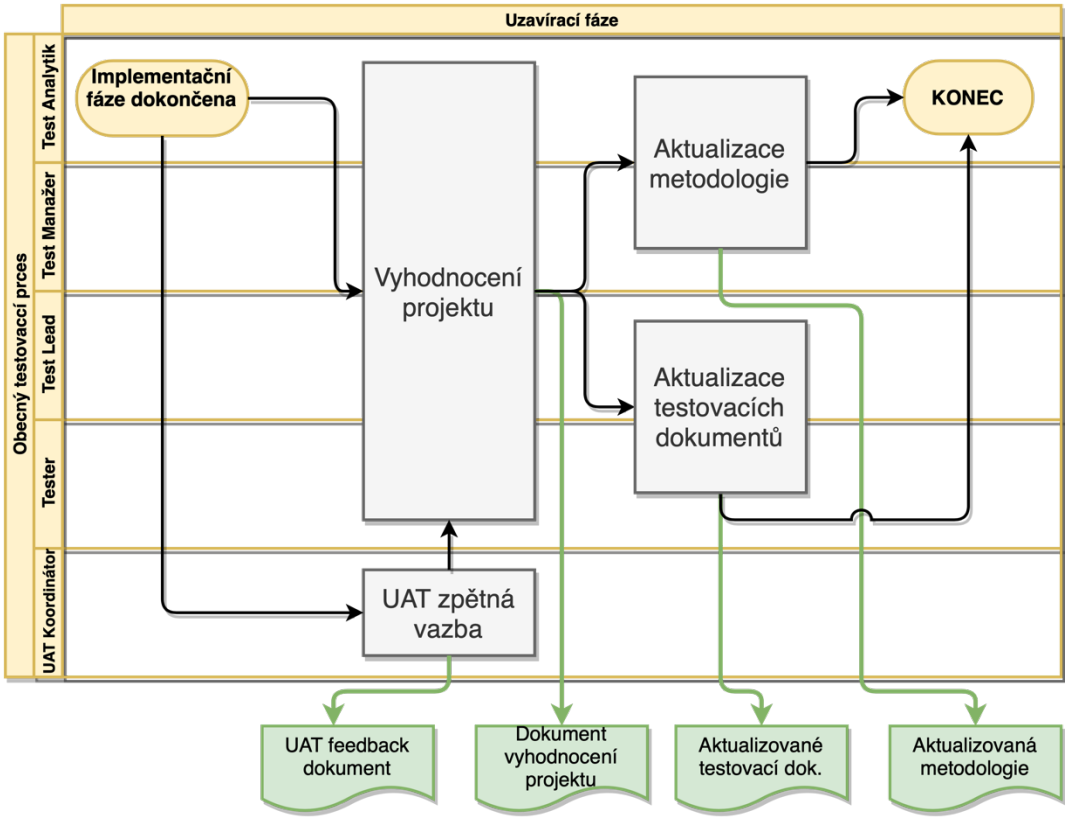
Průběh procesu exekuce UAT je obdobný procesu exekuce IT testování (Obrázek 15), jen má rozdílný počátek, zapojené role a ukončovací fázi. Proces je iniciován dokončením

předchozích procesů – exekucí IT testingu a přípravou na UAT. Z UAT test plánu jsou následně přiděleny UAT koordinátorem jednotlivým testerům testovací scénáře – případě UAT se nejedná o členy testingového oddělení, nýbrž o zástupce businessu a koncové uživatele. Samotný průběh testování pak probíhá stejně jako v případě IT testů. Po dokončení všech testů, retestů bugů a vyhodnocení připravenosti prostředí na release ještě přichází činnost akceptace. Během ní zkontroluje business analytik zodpovědný za UAT na projektu, jestli splňuje projekt své počáteční zadání a oficiálně stvrdí release podpisem akceptačního protokolu.

4.1.2.3 Uzavírací fáze

Testovací proces vždy končí uzavírací fází. Během ní je vyhodnocen uplynulý projekt a z něj získané poznatky. Dochází k aktualizaci testovací dokumentace – regresních testovacích scénářů, testovací metodiky a procesů. Důvodem je, aby byla dokumentace vždy co nejaktuálnější a co nejpřesněji zachycovala současnou logiku a vlastnosti softwaru.

Kromě zpětné vazby od členů testingového týmu, kterou iniciuje a řídí Test manažer, je feedback získáván i od zákazníků, kteří zadali požadavek na vývoj softwaru a také od koncových uživatelů – tuto zpětnou vazbu řídí UAT koordinátor.



Obrázek 18: Uzavírací fáze

Obrázek 18 zobrazuje uzavírací část testovacího procesu a výstupy, které z něj vycházejí formou dokumentů.

4.2 Současný stav automatizace testování

V době psaní této práce je v rámci Pojišťovny využíváno jen minimální množství automatizovaných testů. Většina existující automatizace probíhá v rámci vývoje softwaru na úrovni kódu, a to formou unit a smoke testů. Tyto testy jsou spravovány a spouštěny developery. Samotné testovací oddělení pak v současnosti provádí většinu testů manuálně. Automatizováno je jen několik málo funkcionalit a testů, jako jsou například automatizované checky prostředí.

V následujících kapitolách budou postupně rozebrány všechny automatické testy, které v současné době IT sekce používá.

4.2.1 Automatické testy v rámci vývojové fáze

Jak již bylo řečeno, většina v současnosti existujících automatizovaných testů je spojena právě s vývojovou fází životního cyklu softwaru. Během ní vývojáři naprogramují kód nových funkcionalit a rovnou ho i sami otestují. Jakmile jejich kód projde testy na lokálním prostředí, dojde k integraci jejich kódu do aplikace na serveru, kde proběhnou ještě další automatizované testy, jejichž úkolem je zjistit kompatibilitu nově napsaného kódu s aplikací a ostatními propojenými systémy.

Níže jsou rozepsány jednotlivé testovací fáze, kterými vývojářův kód projde před tím, než je integrován do kódu aplikace a systém přechází do testovací fáze, kdy je od vývojářů předán testerům.

Celý proces testování softwaru během jeho vývoje je pak graficky znázorněn diagramem na obrázku 19.

4.2.1.1 Unit testy

Unit testy jsou průběžně tvořeny a aktualizovány vývojáři. Každý z nich si píše vlastní, či upravuje již existující Unit testy dle svých potřeb. Vývojáři na nich pracují současně s tím, jak píší nový kód pro aplikaci nebo těsně poté, co kód dokončí. Každý developer musí protestovat svůj kód před tím, než s ním postoupí dál po vývojovém cyklu.

Cílem těchto testů je zkontrolovat, jestli aplikace dělá to, co jeho autor zamýšlel a také zjistit, jestli kód funguje tak, jak má. Pokud test skončí neúspěšně, musí developer kód opravit a poté spustit test znovu. Jakmile projde kód unit testem úspěšně, je čas na Smoke test.

Unit testy jsou v současnosti místem, kde probíhá největší část automatizovaného testování. Proces je již funkčně nastaven, ale zatím ještě není pevně integrován do životního cyklu vývoje softwaru. Při vývoji některých aplikací tak unit testy zatím nevznikají.

4.2.1.2 Smoke testy

Kód, který úspěšně projde Unit testy, přechází na lokální smoke test. Smoke testy zkouší, jestli nové funkcionality představené ve vývojářově kódu neohrozí současné fungování aplikace. Vývojář pustí smoke test svého kódu na lokální kopii systému, do kterého se bude kód integrovat a pokud kód projde testem úspěšně, podá vývojář pull request. V případě, že test skončí defektem, musí vývojář kód opravit.

4.2.1.3 Automatický smoke test na serveru

Poté, co zadá vývojář pull request na svůj kód, začne kompilace nového a starého kódu. Tato kompilace probíhá pro kód všech developerů, kteří vytvořili pull request najednou. Zároveň se na serveru spustí smoke test nad kódem, který se snaží vložit do aplikace. Jestliže test proběhne v pořádku, je kód zaintegrován do zdrojového (původního) kódu aplikace. Skončí-li test chybou, k integraci nedojde a kód je vrácen na přepracování developerovi, který jej napsal.

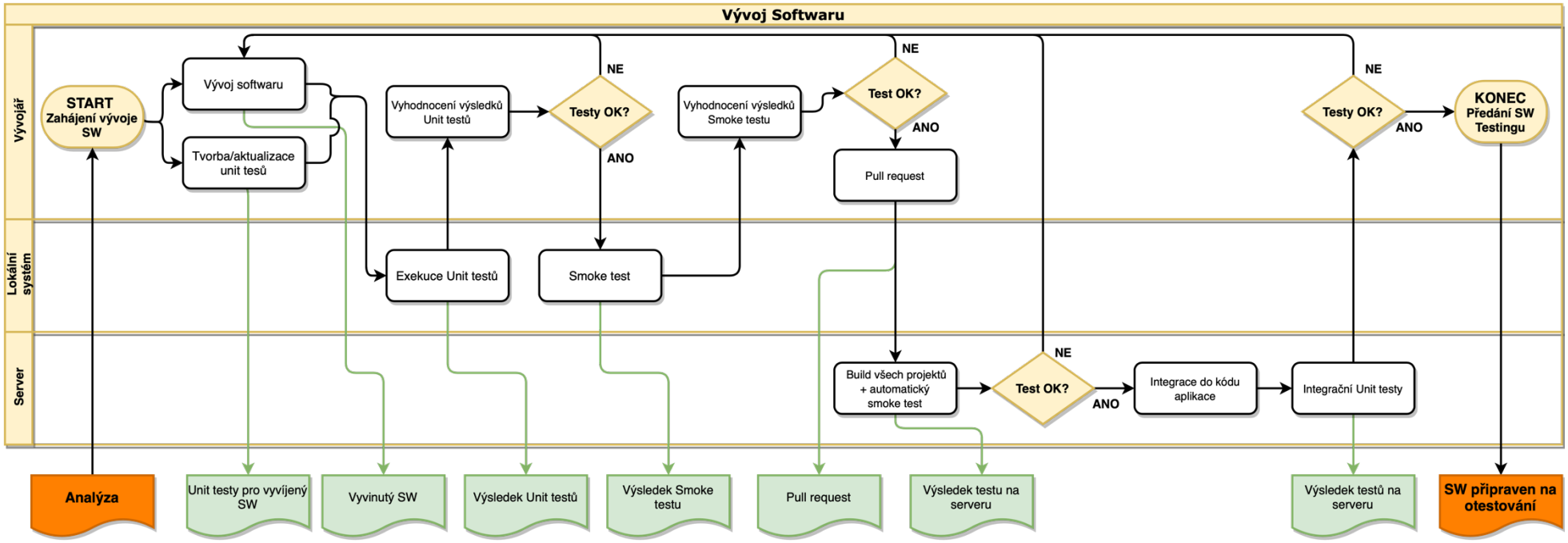
Serverové smoke testy jsou průběžně aktualizovány developery a to tak, aby vždy kontrolovaly nový kód proti aktuální konfiguraci prostředí.

4.2.1.4 Integrační unit testy

Po sloučení developerova kódu se zdrojovým kódem aplikace dojde ještě k poslednímu typu automatických testů na serveru, a to k integračním unit testům. Během této fáze jsou postupně spuštěny unit testy všech součástí systému, jestli fungují tak, jak mají. Poté jsou spuštěny integrační testy, které kontrolují, zda jsou nově přidané části kódu správně zaimplementované a jestli je aplikace propojená s ostatními částmi systému.

Tyto testy běží vždy během noci společně pro pull requesty od všech developerů. Výsledek testů je zaznamenán do logu, který pak každé ráno prochází pověřený developer. Pokud dojde k nějakým chybám, je vadný kód vrácen svému autorovi, k opravě.

V okamžiku, kdy projdou testem úspěšně všechny pull requesty, je software připraven na přechod do další části životního cyklu softwaru, a to na předání SW testingu na otestování.



Obrázek 19: Testování v rámci vývoje SW

Obrázek 19 znázorňuje průběh procesu testování softwaru v rámci jeho vývoje. Všechny testy zapojené do procesu jsou automatizované a spouští je buď sám developer (Unit a Smoke testy na lokálním systému) nebo jsou spuštěny automaticky v předem daných časových intervalech na serveru. Proces končí předáním softwaru testovacímu týmu.

4.2.2 SpecFlow testy

SpecFlow je softwarový nástroj používaný pro testování aplikací. Jeho výhodou je to, že dokáže automaticky spouštět testy psané v prostém smysluplném textu pomocí jednoduché gramatiky (syntax, kterým jsou testy psány je určen v rámci jazyku Gherkin). Vzhledem k tomu, že se jedná de facto o uživatelský programovací jazyk, jeho použitím můžeme značně rozšířit skupinu lidí schopných psát automatické testy. Kromě vývojářů je tak mohou začít psát i programátorsky méně zdatní testeři. V rámci těchto testů je nadefinováno chování aplikace.

V Pojišťovně aktuálně slouží SpecFlow testy jako testy funkční – ověřuje se jimi funkčnost samotného kódu a jen malá část z nich se používá pro testování business logiky (ta interpretuje business pravidla podniku do programu – jak lze data vytvářet, ukládat a upravovat), ke které se SpecFlow nejvíce hodí. Testy se spouští vždy společně s buildem.

Testy zatím pokrývají jen část systémů a funkčností vyvíjených IT sekcí. S ohledem na to, že se v současnosti existující SpecFlow testy používají jako unit testy, má jejich tvorbu a údržbu na starost vývoj, nikoliv testing.

4.2.3 Automatické testy v rámci testovací fáze

Oddělení testingu věnuje většinu své stávající kapacity manuálnímu testování a automatizaci se věnuje jen velmi okrajově. Existující produkty automatizace jsou rozepsány níže.

4.2.3.1 Automatické checky prostředí

Automatická kontrola prostředí ulehčuje testerovi práci tím, že mu řekne, jestli je testovací prostředí vůbec testovatelné. Checky jsou každé ráno manuálně spouštěny a vyhodnocovány testerem. Jejich úkolem je spustit všechna prostředí, na kterých se aktuálně testuje, přihlásit se do nich a zjistit, jestli jsou s nimi propojené všechny požadované systémy.

Případný úspěch, či neúspěch kontroly pak vyhodnotí tester. Jestliže kontrola skončí defektem, nemá cenu na prostředí provádět jakékoliv testy a pověřený tester ihned zahlásí bug.

4.2.3.2 Front-end testy

V rámci aktivit automatizace v testingovém oddělení vzniklo také několik front-end testů simulujících aktivitu manuálního testera. Jak již název napovídá, tyto testy mají za úkol replikovat to, co by jinak dělal tester sám manuálně.

Testy vznikají tím, že je nejdříve tester sám provede a při tom nahraje svůj postup pomocí speciálního softwaru. Vzniklou nahrávku testu může pomocí automatizačního nástroje kdykoliv spustit znovu a použitý nástroj pak přesně krok za krokem zopakuje testerův předchozí průchod aplikací.

Tyto testy fungují na principu absolutní pozice prvků, nebo na porovnávání jejich grafického vzhledu. Z toho také plyne jejich hlavní nevýhoda – a to že fungují pouze na původním, či lehce pozměněném grafickém rozhraní aplikace. Jakmile dojde k větší změně designu, musí se testy nahrát znovu, a to proto, že prvky, které tester používal během své původní exekuce testu byly přesunuty na jiné místo nebo byly změněny tak, že je už testovací program nedokáže rozeznat.

Aktuálně jsou simulace aktivit manuálního testera v omezené míře používány pro testování průchodů online portálem pro klienty a k základnímu otestování softwaru pro správu agendy likvidátorů.

4.3 Shrnutí analýzy stávající situace

Pro stručné shrnutí lze provedenou analýzu současné situace automatizace testování v IT sekci řešené společnosti shrnout do jedné tabulky zobrazující aktuální stav automatizace na jednotlivých aplikacích, které společnost vyvíjí:

		Typ automatizovaného testování						
Vyvíjený software		Unit testy	Smoke testy	Integrační testy	Performance testy	SpecFlow testy	Checky prostředí	Simulace testera
	Správa likvidace							
	Správa importů							
	Portál pro partnery							
	Portál pro klienty							
	Bankovní reporty							
	Bankovní příkazy							

Tabulka 2: Současný stav automatizace testování

Řádky tabulky reprezentují všechny aplikace, které jsou v rámci společnosti vyvíjeny. Sloupce zastupují jednotlivé typy automatizovaných testů. Barevná pole pak reprezentují aktuální stav automatizace. V tabulce jsou zobrazeny celkem čtyři různé stavy automatizace:

- Tmavě zelená – automatické testy se v dané oblasti efektivně využívají
- Světle zelená – automatizace v dané oblasti již v určité míře běží, ale její implementace není konzistentní napříč jednotlivými projekty
- Oranžová – automatické testy existují, ale jejich použití není dostatečně efektivní
- Bílá – k žádné formě automatizace v této oblasti zatím ještě nedošlo

5 Návrh řešení

V této kapitole je popsán vlastní návrh procesu automatizace testování pro Pojišťovnu. Návrh vychází z analýzy současného stavu, jak je popsán v předchozí kapitole a je založen na teoretických základech představených v kapitole 3. Návrh se skládat ze dvou hlavních částí:

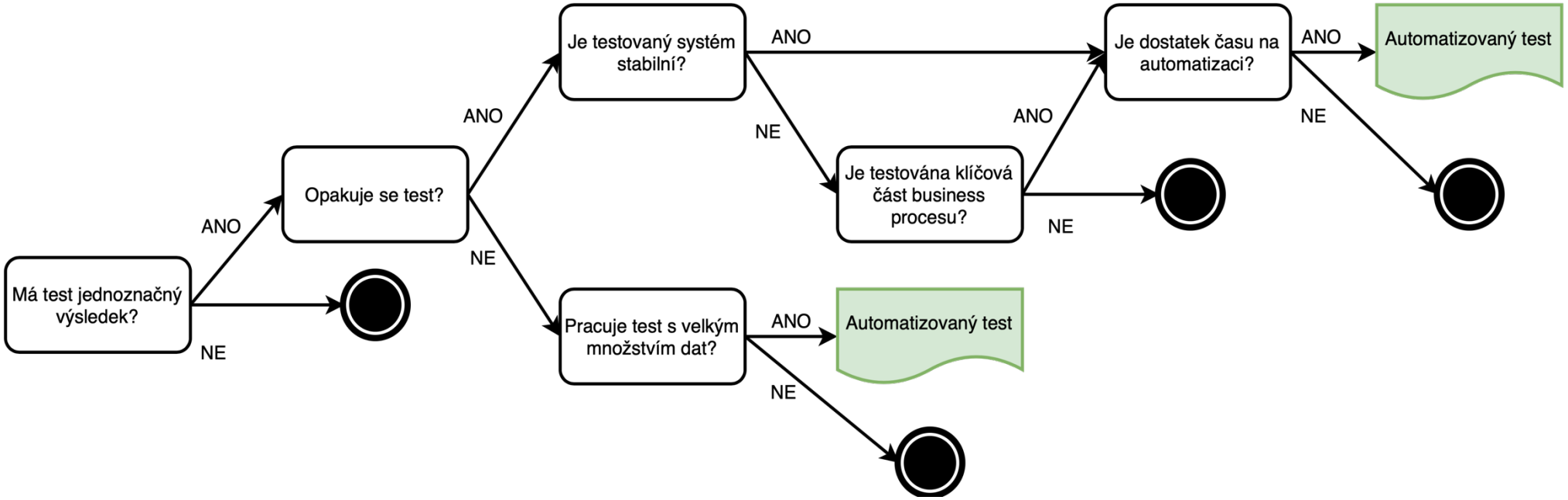
- *Rozhodovací schéma* – schéma slouží jako podpůrný nástroj při rozhodování o tom, které z jednotlivých testovacích scénářů by měly být automatizovány.
- *Návrh procesu automatizace testování* – proces zobrazuje jednotlivé činnosti spojené s automatizováním testování a tvorbou jednotlivých testů.

Doporučení, jak postupovat při nasazování navrženého procesu do stávajícího stavu věcí a jak pracovat s automatizací testování, jsou uvedena v kapitole *6 – Doporučení k implementaci.*

5.1 Rozhodovací schéma

Rozhodovací schéma, často také označované jako rozhodovací strom, je přehledné grafické znázornění rozhodovacího procesu. Diagram rozhodovacího schéma má stromovitou strukturu – skládá se z rozhodovacích uzlů, větví a koncových bodů. V počátečním uzlu (kořeni grafu) se uživatel rozhodne pro jednu ze dvou možností a postupuje po příslušné větvi do dalšího uzlu, kde se postup opakuje, dokud není dosaženo některého z koncových bodů rozhodovacího stromu.

Rozhodovací schéma navržené v rámci řešení této práce by mělo sloužit jako podpůrný nástroj, který bude moci test manažer použít jako pomoc při rozhodování o tom, které testovací scénáře se při testování nové funkcionality vyplatí automatizovat a které ne. Jakmile zjistí, co daný test obnáší a na jakém systému bude probíhat, může tyto znalosti aplikovat na navrhnutém schématu. Schéma je zobrazeno níže, na obrázku 20. Rozhodovací uzly pokládají otázky představené v kapitole *3.1.1 – Kritéria použití automatizace testů*. Postupným průchodem stromem a odpovídáním na jednotlivé otázky dojde test manažer k odpovědi na otázku, zda je automatizace v daném případě vhodná či nikoliv.



Obrázek 20: Rozhodovací schéma

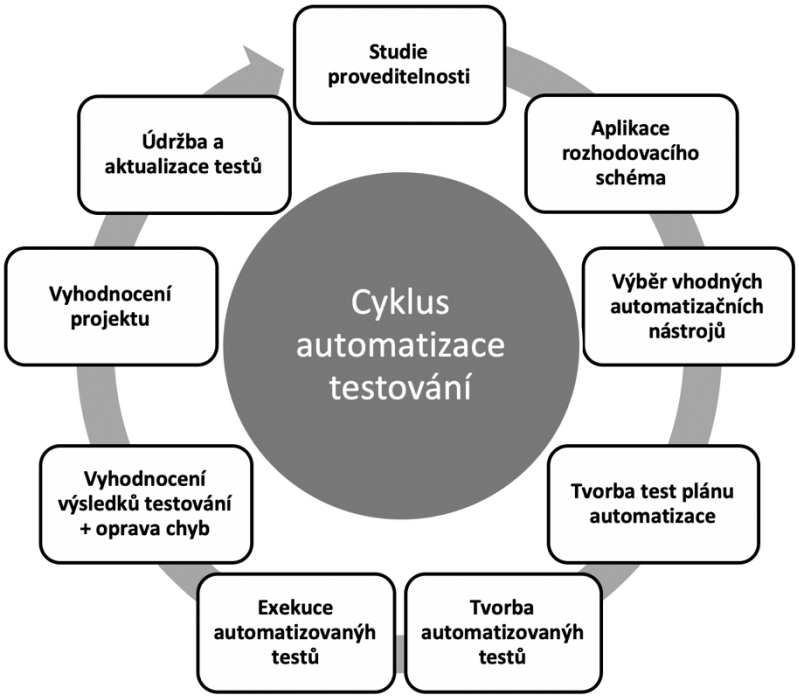
Schéma záměrně, pro jednoduchost jeho aplikace, nepředkládá všechny možné faktory, které by mohly potencionálně ovlivnit automatizaci řešeného testu. Pokládá jen základní otázky, které by měly postačit při rozhodování u valné většiny testů. Najdou se samozřejmě i případy, kdy s použitím rozhodovacího schéma dojde test manažer k nesprávnému doporučení. V těchto situacích se musí test manažer rozhodovat s citem a podle vlastní zkušenosti se zřetelem na ekonomickou a technickou stránku věci.

Vytvořené schéma je dále použito jako součást návrhu procesu automatizace testování, který je představen v následující kapitole.

5.2 Návrh procesu automatizace testování

Cílem tohoto návrhu je vytvořit proces, podle kterého bude možné řídit vývoj a provoz automatizovaných testů v rámci IT sekce řešené společnosti.

Návrh vychází ze současného stavu v Pojišťovně a z teorie představené v kapitole 3, využívá především myšlenky životního cyklu vývoje a provozu systému (SDLC). Navržený proces automatizace testování se také skládá z několika na sebe navazujících fází. Jednotlivé fáze jsou znázorněny pomocí obdélníků na níže uvedeném obrázku 21. V okamžiku, kdy se proces dostane na svůj poslední krok „Údržba a aktualizace testů", začíná celý cyklus nanovo, a to až do té doby, než se stanou dané automatizované testy redundantní.



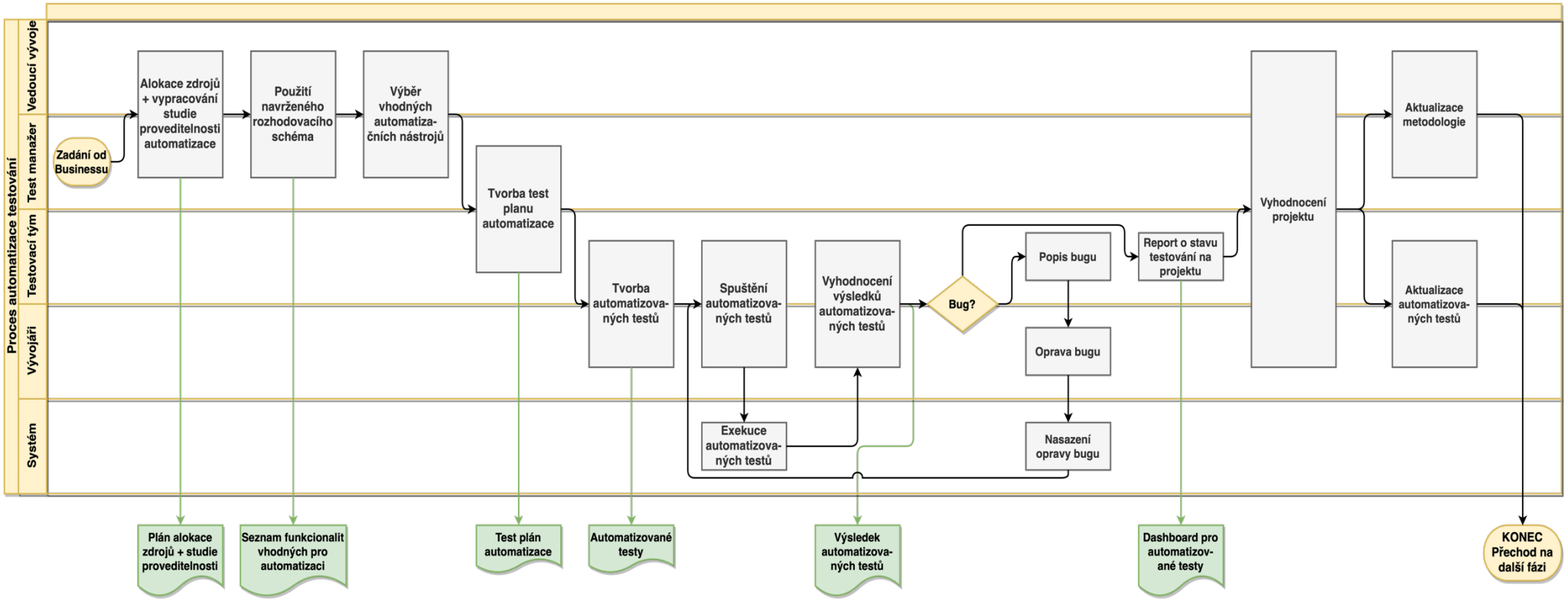
Obrázek 21: Cyklus automatizace testování

Jednotlivými fázemi cyklu automatizace testování jsou:

- *Studie proveditelnosti* – tato studie má za cíl zjistit, zdali jsou vhodné podmínky pro realizaci daného projektu. Test manažer společně s vedoucím vývoje posoudí potencionální přínos projektu zadaného Businessem (vedení Pojišťovny odpovědné za obchod) a na základě svých zkušeností odhadnou náročnost realizace projektu. Pokud dojdou ke shodě, pokračuje se s projektem dál, jinak musí dojít k jeho přepracování, či je rovnou zamítnut. Jestliže je projekt vyhodnocen jako vhodný pro automatizaci, alokují na něj test manažer a vedoucí vývoje potřebné zdroje ze svých oddělení.

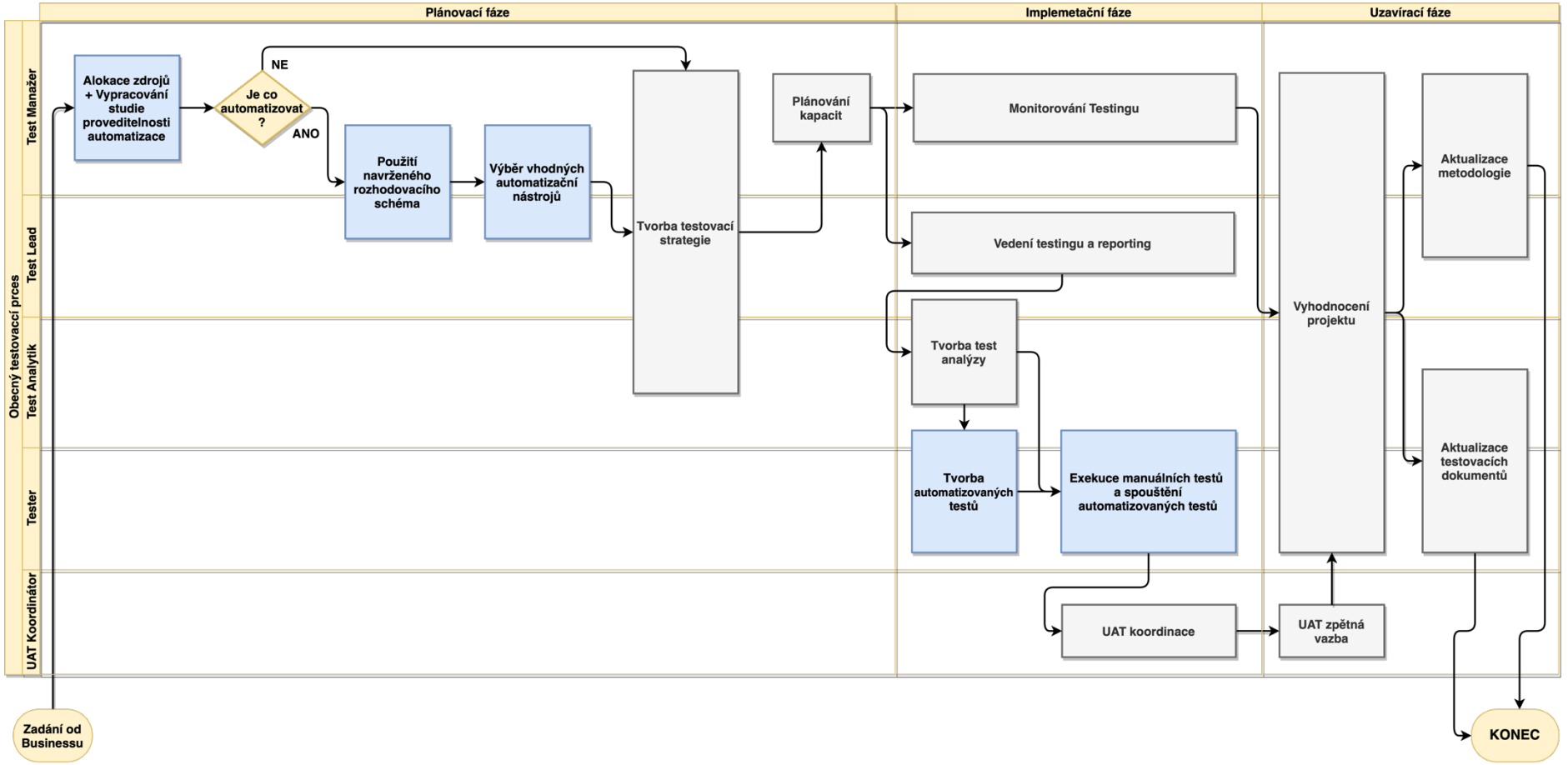
- *Aplikace rozhodovacího schéma* – v této fázi použijí test manažer spolu s vedoucím vývoje rozhodovací schéma navrhnuté v předchozí kapitole k tomu, aby si ověřil, zda se řešené funkcionality a scénáře vyplatí automatizovat, či nikoliv.
- *Výběr vhodných automatizačních nástrojů* – v závislosti na testovaném systému a na tom, jaké testy bude potřeba automatizovat, test manažer a vedoucí vývoje společně vyberou vhodné nástroje pomocí nichž automatizace proběhne. Test manažer zodpovídá za testy spravované týmem testingu a vedoucí vývoje za ty, které patří pod Development.
- *Tvorba test plánu automatizace* – test manažer společně se svým týmem (především test leadery a test analytiky) stanoví test plán, podle kterého se bude postupovat při tvorbě a exekuci automatizovaných testů na projektu.
- *Tvorba automatizovaných testů* – během této fáze dochází již k samotné tvorbě automatizovaných testů. Na tvorbě se podílí jak vývojáři, tak i testingový tým, a to v závislosti na tom, jestli se jedná o developerské automatizované testy nebo o testy, za jejichž tvorbu a provoz je odpovědný testing.
- *Exekuce automatizovaných testů* – nově vytvořené automatizované testy jsou spouštěny na testovaném systému. Testy spouštějí buď testeři, nebo vývojáři, a to podle toho, kdo je za ně odpovědný.
- *Vyhodnocení výsledků testování + oprava chyb* – poté co jsou testy spuštěny a automaticky proběhly musí být jejich výsledky vyhodnoceny. Vyhodnotí je vždy to oddělení, které je spouštělo. V případě, že byl automatizovaným testem nalezen nějaký bug, je nahlášen a předán na opravu vývojářům. Jakmile je oprava bugu nasazena do systému, je ovlivněný automatizovaný test spuštěn znovu, aby bylo ověřeno, že funkcionalita již funguje tak, jak má.
- *Vyhodnocení projektu* – jakmile jsou všechny automatizované testy úspěšně dokončeny, přichází na řadu fáze vyhodnocení projektu. Během ní hodnotí všechny zapojené strany průběh projektu a poskytují k němu zpětnou vazbu. Na základě tohoto hodnocení pak vzniká výstup poslední fáze cyklu automatizace testování.
- *Údržba a aktualizace testů* – poslední fáze cyklu. Dochází zde k aktualizaci pracovní metodologie spojené s automatizací testování a k údržbě a aktualizaci samotných automatizovaných testů. Aktualizace vychází z poznatků získaných v průběhu testování a během fáze vyhodnocení. Jestliže je potřeba větších úprav současně existujících testů nebo je potřeba vyvinout nové testy, začíná celý cyklus nanovo. V případě, že automatizované testy byly zaměřeny na novou funkcionalitu, která byla jejich pomocí již úspěšně otestována, stanou se z funkčních testů testy regresní a budou nadále užívány. Nyní už ne jako test nové funkcionality, ale jako kontrola, jestli nové zásahy do systému nijak neovlivnily dosavadní fungování systému.

Proces automatizace testování rozepsaný v jednotlivých fázích cyklu automatizace testování je graficky zobrazen pomocí diagramu na obrázku 22 na další straně. Dílčí činnosti zmíněné v popisu jednotlivých fází jsou zde znázorněny pomocí šedě orámovaných obdélníků, které jsou vždy zařazeny do dráhy entity zodpovědné za její realizaci. Zeleně jsou pak v diagramu znázorněny výstupy procesu, jako například samotné automatizované testy, či test plán automatizace. Událostí spouštějící celý proces je obdržení zadání požadavku na vývoj od Businessu.



Obrázek 22: Proces automatizace testování

Diagram na obrázku 22 zobrazuje pouze samotný, nově navržený proces. Pro lepší porozumění toho, jaké změny pro jednotlivé role v týmu testingu s sebou přinese zasazení tohoto návrhu do stávajícího testovacího procesu a jeho činností, byl vytvořen následující diagram, viz obrázek 23. Ten vychází z diagramu činností v rámci testovacího procesu v kapitole 4.1.1 a aktualizuje ho o nové činnosti, které testingu přibydou s implementací navrženého procesu automatizace testování.



Obrázek 23: Činnosti v rámci testovacího procesu po implementaci návrhu procesu automatizace testování

Diagram na obrázku 23 zobrazuje jednotlivé činnosti v rámci testingového týmu v průběhu všech fází testovacího procesu (plánovací, implementační a uzavírací fáze). Změnou oproti původnímu diagramu z kapitoly 4.1.1 je implementace činností související s automatizací testování (v diagramu vyznačeny modrou barvou). Test manažer tak nově odpovídá i za vypracování studie proveditelnosti automatizace, výběru oblastí pro automatizaci v rámci projektu (jako podpora této činnosti mu poslouží rozhodovací schéma navrhnuté v kapitole 5.1) a výběru vhodných automatizačních nástrojů.

Výše uvedené činnosti by měl test manažer vykonávat ve spolupráci s vedoucím vývoje (diagram se zabývá pouze testovacím týmem, proto na něm není vedoucí vývoje zobrazen), který odpovídá za automatizované testy spravované developery. Test manažer si může vzít na pomoc při rozhodování test leadera. Testerům a test analytikům přibydou navíc činnosti tvorby a spouštění automatizovaných testů.

Testery lze rozdělit do dvou skupin – na testery manuální a na testery automatizace. Manuální testeři budou mít na starost pouze agendu manuálních testů a automatizované testy budou plně v agendě testerů automatizace. Tento přístup může být efektivní ve velkých týmech. V případě, kdy se jedná o menší tým, kolem 10 členů, je lepší, když jsou alespoň částečně do automatizace zapojeni všichni členové týmu, aby se mohli vzájemně snadno zastoupit a měli přehled o všech testovacích aktivitách, které v podniku probíhají.

Změny se projeví rovněž i v uzavírací fázi. V rámci vyhodnocení projektu dojde nově i k zhodnocení automatizace a při aktualizaci metodologie a testovacích dokumentů dojde k aktualizaci automatizovaných testů.

6 Doporučení k implementaci

Aby byl proces automatizovaného testování co nejefektivnější, je potřeba ho co nejvíce propojit s ostatními procesy vývoje softwaru, zvláště pak s manuálním testováním. V této kapitole jsou rozepsána doporučení k implementaci navrženého procesu automatizace testování a další pokyny, jak s automatizovanými testy pracovat.

Jako první jsou shrnuta doporučení k automatizaci testů:

- Dodržovat jednotný postup při vytváření automatizovaných testů. Proces by měl být přístupný nejen pro všechny členy testingu a developerského týmu, ale i pro všechny ostatní, kteří se podílí na vývojovém procesu.
- Nastavit systém sdílení informací ohledně změn v testovaných systémech. Ideální je stav, kdy jsou všechny plánované změny v systémech hlášeny dopředu. Jestliže tomu tak je, Testing se dokáže včas připravit na změny a zareagovat aktualizací testovacích scénářů.
- Vést společnou evidenci manuálních a automatizovaných testů. Společný seznam všech testů pomůže odhalit duplicity testování (není třeba provádět manuální test funkcionality, kterou již kontroluje automatizovaný test) a také najít oblasti nepokryté testy. Společný seznam může navíc napovědět, které testy by bylo vhodné automatizovat jako další. Tuto společnou evidenci je samozřejmě nutné udržovat a průběžně aktualizovat.
- Pokud nastane požadavek na automatizaci testování u nově vznikajícího systému, je dobré zapojit specialistu na automatizaci již od začátku projektu. Pomocí relativně levných rozhodnutí týkajících se návrhu architektury a stylu vývoje sytému lze ušetřit mnohem vetší prostředky v průběhu testování (8).
- Zapojit testery co nejvíce do procesu automatizace testování. Například předáním odpovědnosti za spouštění automatizovaných front-end testů manuálním testerům se rozšíří jejich povědomí o rozsahu automatizace a navíc se ušetří čas vývojářů.
- Naplno využít již existujících automatizovaných testů. Jestliže má být testovaný systém spuštěn na nové platformě, operačním systému nebo v nové jazykové lokalizaci, není většinou nutné tvořit nové testy, ale stačí poupravit již existující. To samé platí u testů s různými datovými kombinacemi.

Co se týče doporučení pro implementaci nově navrženého procesu mezi již existující firemní procesy, nejdůležitější je, aby byla navrhovaná změna všeobecně přijata. Bez toho bude snaha zavést jakoukoliv změnu velmi obtížná. Proto je důležité důsledně a srozumitelně komunikovat se všemi, jichž se změna dotkne a zdůraznit jim, jaké přínosy a důsledky pro ně nově nastavené procesy budou mít.

Stejně jako u každého projektu, i zde, při zavádění automatizace testování, je třeba, aby byl řádně řízen. To znamená, že je třeba definovat cíle, vytvořit rozfázovaný plán implementace s jednotlivými milníky, sestavit harmonogram a přiřadit projektu zdroje. S tím také souvisí volba vlastníka procesu. Mělo by se jednat o osobu respektovanou všemi zúčastněnými stranami, který dokáže nový proces ve firmě prosadit. Ideálním kandidátem na vlastníka procesu automatizace testování je test manažer, ale může to být i někdo, jiným, kdo má dostatečný přehled o problematice.

Z technického hlediska je třeba vybrat vhodné automatizační nástroje a naučit s jejich pomocí testery a vývojáře vytvářet a spravovat automatizované testy. Nástrojů existuje celá řada od volně přístupných po placené, záležet tedy bude na rozhodnutí aplikačního managementu, zda vybrané nástroje schválí. Volba nevhodného automatizačního nástroje může způsobit znatelné zpoždění a prodražení projektu. Proto je důležité klást na výběr velký důraz a dát si na něm záležet.

Než bude proces automatizace plně spuštěn, je potřeba vytvořit vzory jednotlivých typů automatizovaných testů. Tyto vzory pak budou sloužit jako etalony úrovně a struktury automatizovaných testů ve společnosti.

Po nasazení nového procesu automatizace testování ve společnosti by bylo vhodné, aby se nejprve začalo úpravou již existujících testů, které potřebují doladit (oranžově a světle zeleně vyznačené oblasti v tabulce 2 v kapitole 4.3) a teprve potom přejít k automatizaci nových oblastí.

7 Shrnutí a zhodnocení výsledků práce

Cílem práce bylo navrhnout proces pro řízení tvorby automatizovaných testů v Pojišťovně. Pro účely práce byla provedena rešerše relevantních pramenů teorie, a to z oblastí:

- Základní teorie vývoje softwaru
- Teorie testování softwaru
- Automatizace testování
- Tvorba podnikových procesů

Po základní charakteristice Pojišťovny a představení její IT sekce, byla provedena analýza stávající situace v testingovém oddělení a současného stavu automatizovaného testování.

Analýza popsala složení týmu testingového oddělení, představila, co obnášejí jednotlivé role, které se v oddělení vyskytují a zaměřila se na procesy v rámci vývoje a provozu systému, na nichž se testing v současnosti podílí. Procesy byly detailně popsány a graficky znázorněny pomocí diagramů v BPMN standardu. Dále byl podrobně zanalyzován současný stav automatizace testování. Všechny automatizační aktivity, v současnosti v Pojišťovně vykonávané, byly popsány a znázorněny pomocí souhrnného diagramu.

Z provedené analýzy vyšlo najevo, že v Pojišťovně je sice již část testů automatizována, ale tato činnost je nekoordinovaná a neřízená. Pro větší přehlednost byla vytvořena tabulka zobrazující všechny softwarové aplikace vyvíjené Pojišťovnou a na jaké úrovni je automatizace jejich testování . Jako další nedostatek, kromě nenastaveného procesu pro tvorbu automatizovaných testů, byla vyhodnocena neexistence kritérií pro výběr testovacích scénářů vhodných pro automatizaci.

Závěry této analýzy posloužily, společně s rešerší relevantních pramenů, jako základ pro návrh vlastního řešení. V rámci vlastního řešení bylo vypracováno rozhodovací schéma, které by mělo sloužit jako podpůrný nástroj při rozhodování o tom, jaké testy se vyplatí automatizovat. Vytvořené schéma využívá kritéria pro použití automatizace testování představené v teoretické části práce. Dalším přínosem práce je vytvoření procesu automatizace testování. Tento návrh se skládá ze dvou částí:

1. *Cyklus automatizace testování* – vychází z životního cyklu vývoje a provozu systému a představuje jednotlivé fáze, kterými se prochází při automatizaci testů.
2. *Proces automatizace testování* – dále rozvíjí navržený cyklus a specifikuje činnosti, které v rámci procesu probíhají a zároveň ukazuje, kdo je za jednotlivé činnosti odpovědný a jaké výstupy z nich vznikají.

Nově navržený proces je postaven na zásadách tvorby podnikových procesů a bere v potaz životní cyklus vývoje a provozu systému. Následně je tento proces graficky zpracován do diagramu, ve kterém jsou nově vytvořené činnosti zasazeny do kontextu již existujících činností testovacího týmu v rámci celého testovacího procesu.

Na závěr byla vypracována řada doporučení zaměřujících se na implementaci navrženého procesu automatizace testování do Pojišťovny. Vyzdvižena byla především nutnost otevřené komunikace při zavádění nového procesu do firmy, a to jak ve směru k vedení, tak i k zaměstnancům a také významnost volby správných nástrojů pro automatizaci testů. V kapitole Doporučení k implementaci jsou rovněž uvedeny pokyny pro práci s automatizovanými testy.

8 Závěr

V práci byl navržen proces pro řízení tvorby automatizovaných testů v rámci IT sekce české pobočky nadnárodní pojišťovny a následně byla vytvořena doporučení pro jeho implementaci.

Před vytvořením vlastního návrhu byla představena řešená Pojišťovna a provedena rešerše relevantních zdrojů teorie týkající se vývoje a testování softwaru, automatizace testování a tvorby podnikových procesů. Dalším krokem byla analýza současného stavu v rámci testingového oddělení a jeho přístupu k automatizaci.

Vlastní řešení práce, vycházející z představené teorie a analýzy současného stavu, se skládá z rozhodovacího schématu a návrhu procesu automatizace testování. Návrh byl nejprve popsán a následně znázorněn pomocí BPMN diagramů. Navržený proces byl poté zasazen do diagramu zobrazující současné aktivity testingového oddělení.

Část s vlastním návrhem je uzavřena doporučeními k implementaci, které shrnují pokyny pro práci s automatizovanými testy a pro zavedení procesu do současného stavu věcí v Pojišťovně.

Všechny dílčí úkoly byly splněny a stanoveného cíle práce bylo dosaženo. Výsledky práce je možné v Pojišťovně použít ihned, po alokování potřebných zdrojů na projekt automatizace testování.

Vlastní návrh práce je možné dále rozšířit především o další použití navrženého rozhodovacího schématu na softwarové aplikace vyvíjené v rámci Pojišťovny. Po podrobné analýze jednotlivých aplikací a použití rozhodovacího schématu na tyto aplikace by bylo možné detailně určit oblasti vhodné pro automatizaci testování.

Seznam použitých odborných pojmů

- *SDLC (Software Development Life Cycle)* – zkratka pro anglický název životního cyklu vývoje a provozu systému.
- *Bug* – defekt neboli chyba v kódu aplikace. Jakmile tester zjistí, že se testovaný software nechová, tak jak má, nahlásí bug do systému na reporting.
- *Business* – část Pojišťovny, která je odpovědná za návrh, vývoj a provoz pojišťovacích produktů.
- *Business proces* – soubor navazujících činností, jejichž cílem je poskytnutí služby, nebo produktu klientovi.
- *Front-end* – část systému, kterou vidí běžný uživatel. Například grafické prostředí webových stránek.
- *Back-end* – část sytému, kterou běžný uživatel nevidí. Například aplikační logika za e-shopem na webových stránkách.
- *Check* – kontrola. Tento pojem se používá například v souvislosti s Checkem prostředí. Což je rychlá kontrola toho, jestli jde dané prostředí spustit.
- *Manuální tester* – člen testingového týmu, který vykonává testy sám, manuálně, bez pomoci jakéhokoliv automatizačního softwaru.
- *Build* – zprovozněná verze vyvíjeného softwaru, která je předána vývojáři testingu.
- *Release* – předání systému k užívání jeho koncovým uživatelům.
- *UAT* – zkratka z anglického User Acceptance Testing. Jedná se o akceptační testy, které provádí koncový uživatel systému. Podle jeho výsledku se rozhodne, zda je systém připraven k nasazení na produkci.
- *Specflow* – softwarový nástroj pro automatizaci testů. Umožňuje psát testy i pomocí jednoduché angličtiny. Tudíž odpadá nutnost znát programovací jazyk.
- *Unit test* – česky označovány jako jednotkové testy jsou typem automatický testů, které testují kód aplikace po co nejmenších spustitelných jednotkách. Testy jsou většinou vyvíjeny a udržovány developery.
- *Smoke test* – ověřují, zdali fungují klíčové funkcionality systému.
- *Open-source* – druh licence počítačového softwaru, v rámci které umožňuje držitel autorských práv dalším uživatelům volnou distribuci a úpravy zdrojového kódu zajakýmkoliv účely.

Seznam použité literatury a zdrojů

1. webové stránky řešené společnosti. [Online] [Citace: 12. 1 2020.].

2. Národní soustava povolání. [Online] [Citace: 29. 1 2020.]
https://www.nsp.cz/jednotka-prace/analytik-it .

3. Národní soustava povolání. [Online] [Citace: 29. 1 2020.]
https://www.nsp.cz/jednotka-prace/samostatny-programator.

4. Národní soustava povolání. [Online] [Citace: 29. 1 2020.]
https://www.nsp.cz/jednotka-prace/softwarovy-tester.

5. POUR, J., GÁLA, L., ŠEDIVÁ, Z. *Podniková informatika.* Praha : Grada, 2015.
ISBN 978-80-247-5457-4.

6. Cambridge Dictonary. [Online] [Citace: 25. 1 2020.]
https://dictionary.cambridge.org/dictionary/english/testing.

7. Business encyklopedie. *managementmania.* [Online] [Citace: 29. 1 2020.]
https://managementmania.com/cs/nejlepsi-praxe-best-practice.

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

9. Jones, Capers. A SHORT HISTORY OF THE COST PER DEFECT METRIC. *softwarevalue.*
[Online] 29. 12 2012. [Citace: 18. 04 2020.]
https://www.softwarevalue.com/media/389295/cost-per-defect-2013.pdf.

10. ISTQB Glossary. [Online] [Citace: 17. 4 2020.] https://glossary.istqb.org/en/.

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

12. *A Study of Automated Software Testing: Automation Tools and Frameworks.* Mubarak Albarka Umar, Chen Zhanfang. 06 Nov-Dec 2019, Jilin, China : International Journal of Computer Science Engineering (IJCSE), 2019, Sv. 8. 2319-7323.

13. ŘEPA, V. *Podnikové procesy: procesní řízení a modelování.* Praha : Grada, 2007.
ISBN 80-247-1281-4.

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

15. ŘEPA, V. *Procesně řízená organizace.* Praha : Grada, 2012. ISBN 978-80-247-4128-4.

16. Definice osobních údajů. *GDPR.* [Online] [Citace: 29. 2 2020.]
https://www.gdpr.cz/gdpr/osobni-udaje/.

17. Definice citlivých osobních údajů. *GDPR.* [Online] [Citace: 29. 2 2020.]
https://www.gdpr.cz/gdpr/heslo/citlive-osobni-udaje/.

Seznam obrázků

Seznam grafů

Seznam tabulek

Všechny obrázky, grafy a tabulky, které u sebe nemají uvedený zdroj, byly vytvořeny autorem pro účely této 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: 15. 05. 2020 Podpis:

Jméno	Oddělení/Pracoviště	Datum	Podpis