Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Testování aplikací z hlediska použitelnosti

Tomáš Steska

© 2017 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Tomáš Steska

Informatika

Název práce

Testování aplikací z hlediska použitelnosti

Název anglicky

Application testing in terms of usability

Cíle práce

Diplomová práce je tematicky zaměřena na problematiku testování a testovací analýzy.
Hlavním cílem práce je vytvoření testovací analýzy a postupu testování zvoleného projektu.

Dílčí cíle práce jsou:
- analýza metod pro hodnocení kvality Sw
- porovnání metod pro hodnocení použitelnosti
- případová studie
- vytvoření reportu a vyhodnocení testování
- formulování obecných a specifických závěrů

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů.

Vlastní práce spočívá ve vytvoření přehledu testovacích principů, následném výběru nejvhodnějších metod v souladu s vytvořenou testovací analýzou a jejich aplikací na projekt.

Součástí práce bude také návrh testovacích scénářů, reportování a vyhodnocení testů zvoleného projektu.

Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry diplomové práce.

Doporučený rozsah práce

55

Klíčová slova

Testování; Analýza; Reporting; Testovací analýza; Projekt; Metodika

---

Doporučené zdroje informací

ČERMÁK, V. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. PROVOZNĚ EKONOMICKÁ FAKULTA, – HAVLÍČEK, Z. *Návrh a tvorba internetových aplikací.* Disertační práce. Praha: 2009.

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

MACHÁČKOVÁ, K. – VYSKOTOVÁ, J. *Jemná motorika : vývoj, motorická kontrola, hodnocení a testování.* Praha: Grada, 2013. ISBN 978-80-247-4698-2.

VANÍČEK, J. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA, PROVOZNĚ EKONOMICKÁ FAKULTA, – KARDOŠ, D. *Statistický a dynamický přístup k jakosti informačních systémů.* Disertační práce. Praha: 2010.

VANÍČEK, J. – ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE. KATEDRA INFORMAČNÍHO INŽENÝRSTVÍ. *Měření a hodnocení jakosti informačních systémů.* V Praze: Česká zemědělská univerzita, Provozně ekonomická fakulta, 2004. ISBN 80-213-1206-8.

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

doc. Ing. Zdeněk Havlíček, CSc.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 21. 10. 2016

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 10. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 19. 03. 2017

Čestné prohlášení

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

V Praze dne 29. 3. 2017 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce doc. Ing. Zdeňku Havlíčkovi, CSc. za cenné rady a strávený čas, který mně věnoval při zpracovávání této diplomové práce. Dále bych rád poděkoval Martinu Mičkalovi za odborné rady z oblasti testování a celé své rodině za poskytnutou podporu při psaní této práce.

Testování aplikací z hlediska použitelnosti

Souhrn

Software se v současné době stává každodenním standardem pro náš běžný život. Se stále rostoucím počtem technologií, které nás obklopují, spoléháme čím dál více na jejich funkčnost. Díky tomu se klade větší důraz na testování zajišťující správnost všech funkcí pomocí nejrůznějších metriky a metody testování.

V rámci rešerše je zpracován základní přehled testování, ať už se jedná o názvosloví nebo základní popisy testovacích modelů, fází, či typů testů, až po složitější kapitoly zaměřující se na tvorbu analýzy a techniky návrhů testů, či sestavování testovacího plánu jako hlavního podkladu projektu. V závěru teoretické části jsou představeny způsoby pro zpracování kvality a použitelnosti.

Praktická část využívá teoretických východisek k sestavení testovacího plánu, včetně všech náležitostí, analýzy, nastavení reportingu a vyhodnocování kvality na projet z bankovního sektoru. V rámci analýzy je připraven návrh jednotlivých testovacích případů, včetně návrhu automatizace.

Výstupem je sestavení testovacího podkladu, včetně návrhu reportů a metod pro budoucí testování grafického prostředí.

Klíčová slova: Testování; Analýza; Reporting; Testovací analýza; Projekt; Metodika

Application testing in terms of usability

Summary

Nowadays, software is becoming a standard of our daily lives. With growing number of technologies around us, we are relying more and more on their functioning. Due to that greater emphasis is placed on testing, which should secure good functionality using various metrics and methods.

In the framework of the research the essential summary of testing is provided ranging from terminology, basic descriptions of test models, stages or types of tests, to complex chapters focusing on creation of analysis, design of tests, and compilation of test plan as the core of the project. The last part of the theoretical chapter is devoted to methods for evaluation of quality and usability.

Practical part uses the theoretical framework in order to create testing plan with all its requirements, analysis, settings of reporting, and quality evaluation applied on the case from banking sector. As a component of analysis proposal various test cases including suggestion for automatization are prepared.

The expected outcome of this thesis is the testing scenario, which includes proposal of reports and methods for future testing of graphical interface.

Keywords: Testing; Analysis; Reporting; Test analysis; Project; Methodics

Obsah

1 Úvod

Aniž si to uvědomujeme, náš každodenní život je v nejrůznější podobách ovlivňován různými druhy softwaru. Počínaje aplikacemi v různorodých mobilních zařízeních, software k podpoře vzdělávání, přes robustní bankovní, průmyslové a energetické systémy až k lékařským přístrojům zachraňujících životy. Na software jsme si již zvykli zcela spoléhat a brát ho jako samozřejmou součást života, aniž si připouštíme, že jakákoliv, byť dočasná, nefunkčnost software může mít pro nás i fatální důsledky. Například velké finanční škody nebo ztráty na životech.

Se stále rozšiřující se škálou nových systému a jejich rostoucí komplikovaností, se začíná přikládat větší důraz na testování, aby se problémům v dostatečné míře předešlo. Proto většina firem přistupuje stále zodpovědněji ke kontrole výstupního produktu a sestavuje interní, externí, či kombinované testovací týmy, aby co nejvíce omezily, nebo dokonce eliminovaly možné negativní dopady.

Základním úkolem testovacího týmu spočívá v upřesnění nejasností z business požadavků tak, aby proběhla analýza a sestavení testovacího plánu v souladu se všemi požadavky systému. Na základě definovaných podkladů pak dojde k sestavení opory testování v podobě testovacích případů, které zároveň slouží jako primární podklad pro hodnocení kvality a tvorby reportů.

Dynamický rozvoj nasazování nových technologií, jako například dronů, chytrých domácností a dalších, bude vyžadovat vývoj stále sofistikovanějších systémů pro jejich ovládání. To bude přinášet i větší požadavky na oblast testování, nárůst oborů zaměřujících se na problematiku testů a tomu úměrný nárůst celé oblasti testování.

Diplomová práce je zaměřena na rozbor problematiky testování od nastavení testovacího plánu a vytvoření správné analýzy až po vyhodnocení jednotlivých testů a metrik. Teoretické části jsou doplněny konkrétními příklady.

2 Cíl práce a metodika

2.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku testování a testovací analýzy. Hlavním cílem práce je vytvoření testovací analýzy a postupu testování zvoleného projektu.

Dílčí cíle práce jsou:

- analýza metod pro hodnocení kvality Sw
- porovnání metod pro hodnocení použitelnosti
- případová studie
- vytvoření reportu a vyhodnocení testování
- formulování obecných a specifických závěrů

2.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů.

Vlastní práce spočívá ve vytvoření přehledu testovacích principů, následném výběru nejvhodnějších metod v souladu s vytvořenou testovací analýzou a jejich aplikací na projekt.

Součástí práce bude také návrh testovacích scénářů, reportování a vyhodnocení testů zvoleného projektu.

Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry diplomové práce.

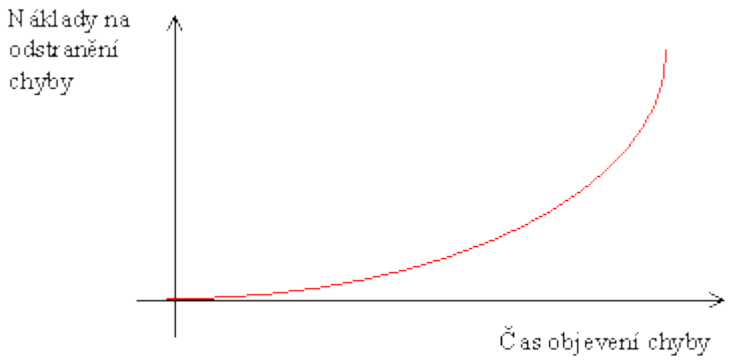
3 Teoretická východiska

3.1 Úvod do problematiky testování

V současné době je využívání softwarových systémů nedílnou součástí moderní společnosti. Vzhledem k velikosti trhu a různorodosti firem, které se na něm pohybují, většinou neexistuje univerzální odladěný software, ale spíše varianty v podobě, která je založena na „krabicové verzi“ a následně upravena na specifické prostředí uvnitř instituce. Využívané aplikace pak slouží k nejrůznějšímu využití, ať už se již jedná o bankovní či farmaceutické sektory. V různých odvětvích se objevují i různě závažné chyby, od „neškodných“ chyb způsobujících grafické odchylky, až po bugy na lékařských přístrojích, které mohou způsobit fatální důsledky. Tady lze hledat samotného testování, s cílem předejít většině chyb, které by mohly vést až k smrtelným následkům.

Do testování nepatří pouze vykonávání testů nad přiděleným softwarem, ale zahrnuje například i činnosti jako je testovací analýza, která dodává podklady pro samotné testy, nebo reportování výsledků testů. Ty můžou pak být využity pro vytvoření přehledu slabých míst testovaného softwaru, nahrazení kontroly kvality SW a mnoho jiných případů.

Dalším důvodem testování je finanční hledisko. Samotná fáze testování není, obzvlášť u větších projektů, levná záležitost, a to jak z hlediska personálních, tak finančních zdrojů, ale je třeba brát v potaz, že odstranění každé chyby odhalené v této fázi je několikanásobně levnější než jejich zpětná fixace z již produkčního prostředí. Obecně platí „Čím později chybu objevíme, tím budou náklady na její odstranění větší“ viz Obrázek 1.



Obrázek 1 Výše nákladů na odstranění chyby v poměru s časem jejího objevení (Lacko, 2006)

ISTQB standard

ISTQB neboli International Software Testing Qualifications Board, je jedna z hlavních mezinárodních organizací v oblasti testování softwaru. Vytváří celosvětový standart pro testování softwaru a vydavatel jednoho z nejznámějších certifikátů v oblasti testování, který je rozdělen do několika úrovní podle obtížnosti a detailu zpracování (většinou zaměřen na určitou skupinu).

- Foundation Level (CTFL)
- Advanced Level (CTAL)
- Expert level (CTEL) (CaSTB, 2017)

3.1.1 Základní pojmy v testování

Testovací případ

Často taky označován anglickým názvem test case nebo zkratkou „TC“, popisuje činnosti, které mají být provedeny pro pozitivní/negativní průchod softwarem a jejich očekávaný výsledek. Každý test case se skládá z jednotlivých kroků (test step), které mají být v určené posloupnosti provedeny. V rámci testovacího případu, můžou být uvedeny taky testovací data, které slouží jako vstup v rámci případu.

Testovací skript

Jedná se o skript, sloužící k automatizovanému testování softwaru. Každý skript by měl mít přidělený unikátní identifikátor a popis, k jakým testům se vztahuje, jak ho lze spustit, jaké má vstupní parametry, jak skript obnovit v případě přerušení, kam se ukládají záznamy o aktivitě (log) a jaké činnosti je potřeba provést po jeho skončení.

Chyba

Chybu udělá člověk, vývojář či analytik při práci na svém artefaktu (fragment kódu, kapitola analýzy, diagram) (Bureš, a další, 2016).

Defekt

Defekt (bug) je odchylka aktuálního způsobu fungování softwaru od očekávaného. Defekt je projevem chyby a je tady potřeba chybu najít a defekt opravit. Mezi nejčastější příčiny defektů patří:

- Nesprávné pochopení potřeby uživatele vedoucí k chybné specifikaci požadavků.
- Nesprávně nebo neúplně definované požadavky.
- Požadavky nesprávně interpretované do funkčního designu.
- Nesprávné pochopení technického designu (nesprávně naprogramovaný software).
- Nesprávně pochopené požadavky či funkční design.
- Nedostatečný výkon aplikace, špatná ovladatelnost, uživatelské nepohodlí. (Bureš, a další, 2016)

Po objevení samotného problému je potřeba tento defekt zaevidovat, a to s uvedením určitých povinných atributů: Datum a čas nalezení, autor záznamu, očekávaný a skutečný výsledek, identifikace předmětu testování a prostředí, popis defektu (může obsahovat přiložený log, snímek obrazovky atd.), míra dopadu a stav defektu. Mimo tyto informace se často eviduje i priorita, závažnost, naléhavost, historie změn, fáze projektu, ve které defekt nastal, odkaz na testovací scénář a nakonec datum a způsob vyřešení. Zaevidovaný defekt pak prochází vlastním životním cyklem (workflow), který je tvořený stavy. (Bureš, a další, 2016)

Závažnost a priorita defektu

Závažnost často označována taky jako Severity určuje, jak velký bude bezprostřední nebo budoucí dopad defektu na testovaný systém. V rámci severity není důležité kolikrát se chyba ve výsledném systému objeví, nebo kolika uživatelů se dotkne. (Black, 2009) Severitu lze rozdělit například do následující struktury:

1. Kritická (Critical) - Ztráta dat, poškození hardwaru, ukončení celého systému nebo otázky bezpečnosti. Chybná funkce je nepoužitelná a neexistuje alternativní metoda pro dosažení výsledků.
2. Vysoká (Major) - Vada, která má za následek ukončení celého systému, jedné nebo více složek systému nebo způsobuje poškození dat. Chybná funkce je nepoužitelná, ale existuje alternativa, jak dosáhnout požadovaných výsledků.
3. Střední (Moderate) - Vada, která nevede k zániku, ale způsobuje, že systém vrací nesprávné, neúplné nebo nekonzistentní výsledky.

4. Nízká (Minor) – Vada, která způsobuje částečnou ztrátu funkčnosti, ale neovlivňuje celkovou použitelnost systému.
5. Kosmetická (Cosmetic) - Kosmetická nebo triviální vada, které se často týkají vzhledu nebo vylepšení systému. (Severity and Priority, 2015)

Priorita defektu neboli priority, udává důležitost jednotlivých oprav, a to na základě toho jak velký je dopad na aktuální funkčnost modulu/aplikace a jak velký by byl dopad na koncového uživatele systému, za předpokladu, že by se chyby nepodařilo/nestihlo opravit. Základní metodiku pro určování priority určuje projektový manažer a test manažer na začátku každého projektu. Lze se setkat s případem, kdy není pevně stanovené škálování a o prioritě rozhoduje samotný nálezce defektu. (Farrell-Vinay, 2008) Pro nastavení priority se většinou volí následující struktura (mimo uvedené stavy se často využívá i stavu kritický nebo blokující, které se evidují nad stupněm high):

1. Vysoká (High) – Defekt má nejvyšší prioritu a musí být co nejrychleji opraven. Je volena pro fatální chyby v aplikaci, které způsobují nefunkčnost celého/části systému, dokud nebude chyba opravena.
2. Střední (Medium) – Úroveň chyby, která je běžně evidována a je často opravována v rámci nasazování nových verzí (buildů) aplikace. Jedná se o chybnou funkcionalitu aplikace, nikoliv však takovou, která by omezovala její chod.
3. Nízká (Low) – Chyba neovlivňuje základní funkce aplikace a může být opravena/řešena až po odstranění zásadnějších nedostatků. (Severity and Priority, 2015)

Selhání

Selhání (failure) systému může nastat v důsledku defektu, kdy systém či některá jeho část obvykle zhavaruje (Bureš, a další, 2016).

Incident

Incident je podle definice ISTQB „jakákoliv událost, která vyžaduje prozkoumání.“ V praxi je však často tento pojem spojován s označením vady nalezené v produkčním prostředí. Incidenty bývají obvykle mnohem více sledovány a jejich náprava je vždy dražší, než kdyby byly odhaleny dříve v průběhu projektu. (Bureš, a další, 2016)

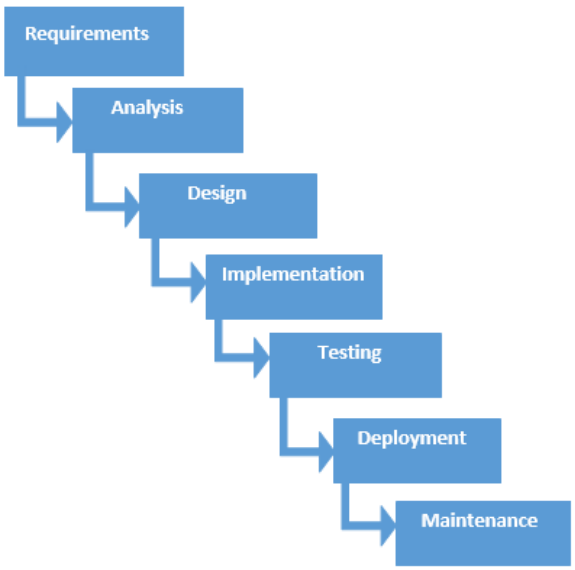
3.2 Typy testů a jejich fáze

3.2.1 Modely životního cyklu testování

Model testování nám určuje pohled na testovací proces od nápadu přes realizaci a následné testování a nasazení. V dnešní době mezi nejznámější typy modelů patří vodopádový model, spirálový model, V model a W model.

3.2.1.1 Vodopádový model

Vodopádový model patří k nejstarším modelům vůbec, byl představen na přelomu 70. a 80. let. Ve vodopádovém modelu se jednotlivé aktivity provádějí postupně, navzájem na sebe navazují a vzájemně se přitom neprotínají. Etapy probíhají podle přesně stanoveného plánu a do další fáze se přechází až ve chvíli, kdy je fáze předchozí kompletně uzavřena a schválena. Jednotlivé fáze vodopádového modelu jsou požadavky (requirement), analýza (analysis), systémový návrh (systém design), implementace (implementation), testování (testing), nasazení (deployment) a údržba (maintenance). Je potřeba věnovat velké úsilí na vytvoření dokonalého podkladu v prvotních fázích, jelikož v případě odhalení chyby v pozdějších fázích se k nim již nelze vrátit. (Testingfreak, 2017)



Obrázek 2 Znázornění vodopádového modelu (Testingfreak, 2017)

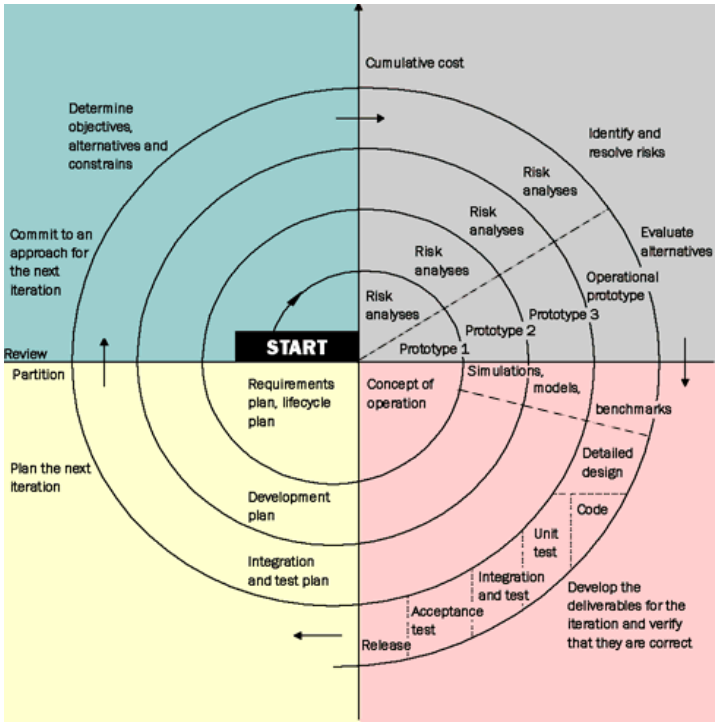
Ve fázi požadavků se všechny předpoklady shromažďují a dokumentují. Je potřeba, aby byly zaevidovány všechny možné požadavky. Ve fázi analýzy se jednotlivé požadavky třídí na ty, které jsou relevantní a které ne. Následuje vytvoření návrhu na základě validních požadavků. Výstupem je specifikace hardwaru, systémové požadavky a architektura celého systému. Následuje implementace, která vychází z poznatků vytvořených v rámci systémového návrhu. Jsou provedeny všechny vývojové práce včetně vývoje komponent a uskuteční se první unit testy (kapitola 3.2.5.1). Následuje fáze testování, ve které se ověří, že všechny komponenty fungují podle dané specifikace. Všechny různé testy se provádí v průběhu této fáze. V případě, že již nejsou v systému „žádné“ chyby může se systém nasadit do produkčního prostředí. V této fázi se začíná plně využívat zaměstnanci/zákazníky. V poslední fázi údržby se řeší problémy nalezené v rámci produkce. Po nalezení chyby se vydává nová záplata a nasazuje se nová verze systému. (Tutorialspoint, 2017)

Výhodou tohoto modelu je jeho jednoduchost a snadné pochopení. Díky specifickým výstupům je v rámci modelu velmi jednoduchý způsob řízení. Další výhodou je, že se jednotlivé aktivity navzájem nepřekrývají a než započne další fáze, musí být předchozí uzavřena. Tento model se hodí především pro zpracování menších projektů. (Waterfall model, 2017)

Jeho nevýhodou je problematické vracení se do předchozích fází při nalezení chyby/nedokonalosti, takže je celý projekt veden pod velkým rizikem. Není vhodná na projekty komplexnější, dlouhodobé, či projekty u kterých nastává v průběhu ke změně rizik. (Waterfall model, 2017)

3.2.1.2 Spirálový model

Spirálový model vychází z principu vodopádu, ale zaměřuje se mnohem více na analýzu rizik v rámců projektu. Model je vytvořený celkem čtyřmi fázemi, a to fází plánování, identifikací, analýzou a vývojem. Projekt může během svého životního cyklu projít všemi fázemi několikrát s tím, že každá další spirála znamená zdražení projektu. (Umel, VUT Brno, 2012)



Obrázek 3 Znázornění podoby spirálového pojetí životního cyklu (Kutina, 2010)

První fází je identifikace, ve které se identifikují business požadavky, nastavení jednotlivých cílů projektu, nastavení možných alternativ a vytvoření omezení. Zde se nachází začátek spirály.

Fáze analýzy má za úkol ověřit nastavené alternativy projektu, proběhne identifikace možných rizik a návrh jejich řešení. V rámci identifikace rizik se taky vytvoří první verze prototypu.

Fáze vývoje, ve které probíhá samotný vývoj aplikace v prvních spirálách. V pozdějších je pak provedeno ověření návrhu produktu a jeho kompletní testování. Tato fáze je zakončena nasazením do produkce.

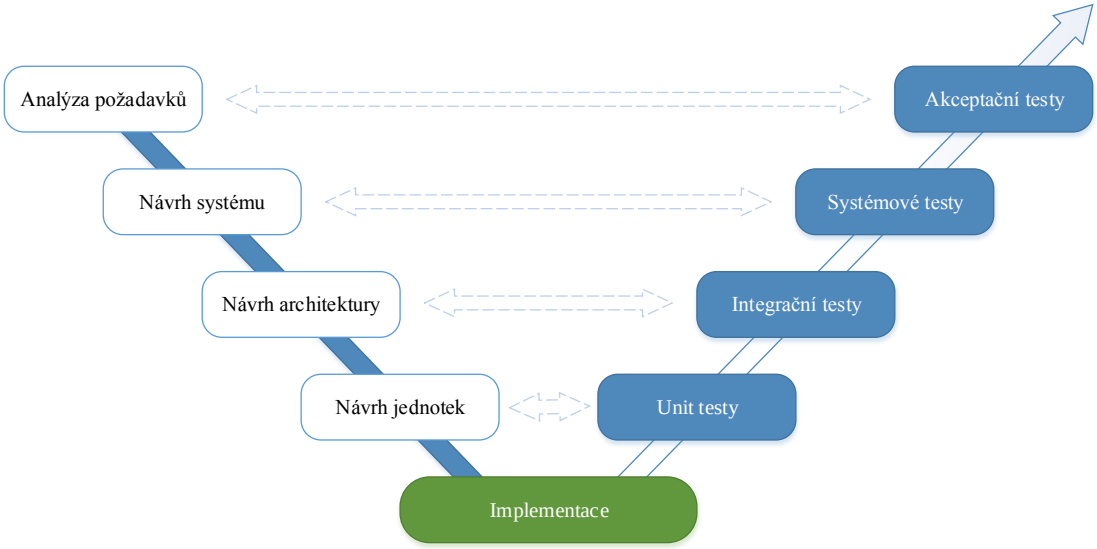
Poslední fáze, je plánování. V této fázi se postupně vytvářejí plány a harmonogramy pro následující spirály.

Výhodou spirálového modelu je důkladná analýza rizik. Díky tomu se používá na větší rizikové projekty, kde uvidíme software již v prvních fázích spirály. V rámci cyklu není problémem přidání nových funkcionalit a celý proces je podložený detailní dokumentací. Nevýhodou pak je cenová náročnost celého zpracování a jeho problematičnost na menších projektech. Hlavní nevýhodou je nastavení velmi detailní risk analýzy, která na zpracování

potřebuje velmi odborné znalosti. Na základě tohoto detailního podkladu se pak následně staví celá spirála. (Spiral model, 2017)

3.2.1.3 V-model

Běžný V model má standardně čtyři úrovně testování, a je zakreslen do tvaru písmene V, ale může se vyskytovat i v podobách, kdy má úrovní méně či více. Základní čtyři úrovně se na pravé straně označují analýza požadavků, návrh systému, návrh architektury a návrh jednotek, a naopak na levé straně se nazývají testování komponent, integrační testování, systémové testování a akceptační testování. Každá fáze na straně analýzy a návrhu pak odpovídá fázi v testování na stejné vrstvě.

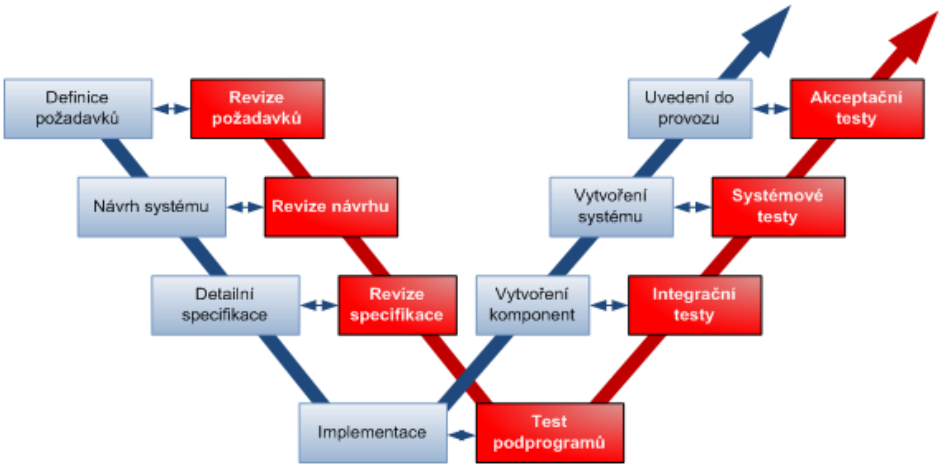


Obrázek 4 Životní cyklus pro V-model (MS Visio)

Výhodou modelu jsou ETP (early test preparation), které umožňují nalézt chyby již v analytických fázích a snižují tak rizika chyb, které by mohly nastat v testovacích fázích. Mezi další výhody patří potřeba rychlého a kvalitního dodání dokumentace, nastavení logických vztahů mezi jednotlivými fázemi, stanovuje stejnou váhu fázím vývoje a analýzy a fázím testování, či nabízí jednoduchý pohled na celý proces vývoje softwaru. Nevýhodou je drahé odstraňování chyb, které jsou způsobeny na základě chybně vytvořených požadavků a odhaleny až v pozdějších fázích testování. (Trnka, 2008)

3.2.1.4 W-model

Roku 1993 vznikl model s označením W-model, který měl za úkol odstranit nedostatky V-modelu. Model je více zaměřený na produkt a ne pouze na splnění testování v jednotlivých fázích jako V-model. Každá plánovací, analytická, nebo vývojová aktivita má vlastní testovací aktivitu, která ji ověřuje. Jednotlivé aktivity jsou rozděleny do fázích, která jsou stejné jako u V-modelu a dodržuje metodiku, že na pravé straně se nacházejí analytické a návrhové aktivity a na straně levé aktivity výbojové. (Umel, VUT Brno, 2012)



Obrázek 5 Model životního cyklu pro W-model (Umel, VUT Brno, 2012)

Výhodu tohoto systému je především rychlejší a levnější řešení chyb, které jsou odhaleny. Nevýhodou tohoto modelu je především nutnost počáteční investice, která slouží na naplánování testovacích aktivit, výběr použitých testovacích technik a identifikaci unikátních rizik. Díky této základní investici není v dnešní době tolik rozšířený jako V-model.

3.2.2 Statické a dynamické testy

Prvním pohledem na testování je rozdělení na statické a dynamické testy. Hlavním rozdílem mezi těmito testy je, zda již máme spustitelný software k dispozici či nikoliv.

3.2.2.1 Statické testy

Pro statické testy není potřeba žádný spustitelný software, ale zaměřuje se především na revizi jednotlivých dokumentací a analýz. Tyto činnosti se dají provádět ještě před začátkem vývoje SW a provádí se manuálně.

Výhodou odhalených defektů v počátečních fázích životního cyklu testování touto metodou je často snadnější odstranitelnost, která je i finančně mnohem levnější než nalezení chyby v pozdějších fázích. Lze takto odhalit například chyby v požadavcích, špatně nastavený časový harmonogram, defekty v návrhu testování, odchylky proti standardům a podobně. (Naik, a další, 2008)

3.2.2.2 Dynamické testy

U dynamických testů se předpokládá existence již spustitelné verze softwaru, která je alespoň ve fázi prototypu. Tyto testy se používají v pozdějších fázích testování.

3.2.3 Testy podle znalosti kódu

Testy podle znalosti kódu nám specifikují, jak velký má tester přístup ke zdrojovému kódu testované aplikace. Na základě tohoto se rozlišují 3 pohledy: whitebox, blackbox a greybox.

3.2.3.1 Whitebox

Whitebox testování někdy známé taky jako „glass box“, vyžaduje přesnou znalost fungování programu (víme, co do programu vstupuje, co se s daty stane a co a v jaké podobě z něj vystupuje). Musí být zajištěný přístup ke zdrojovému kódu, datovému modelu aplikace, či dokumentaci aplikace/programu. U testera se předpokládá porozumění všem zmíněným skutečnostem na dostatečné úrovni. Testování se může provádět ručně i automaticky, ale většinou v praxi nastává jejich kombinace. Princip whitebox je výhodný především pro testování webových aplikací, ověření přístupů k jednotlivým částem aplikace, aby nedocházelo k neautorizovaným přístupům, nebo objevení a odstranění nežádoucího kódu, který by mohl umožnit zneužití aplikace v podobě získání citlivých dat, přístupů do aplikace atd. (Čermák, 2010c)

Velkou výhodou tohoto principu je již zmiňované odhalení nežádoucího kódu, který by mohla aplikace obsahovat a brzké odhalení chyb, které mohly být způsobené chybou programátora a doposud neodhalené. Nevýhoda této metody je v nutné profesionalitě testera, vysokých nákladech na provádění těchto detailních testů a časové náročnosti. (Čermák, 2010c)

3.2.3.2 Blackbox

Typ testování blackbox jindy nazývaný jako „close box“ je prováděn bez znalosti vnitřní struktury programu např. do programu vstoupí dvě hodnoty a vrátí se jedna a tester neví, jaké operace se provedly uvnitř programu. Tester často nemá k dispozici ani žádnou dokumentaci. Tato metoda se provádí na základě vytvořených testovacích scénářů, které jsou testerovi buď dodány, nebo si je vytváří sám. Testovací scénáře jsou většinou vytvářeny na základě testovací analýzy, kde je vidět seznam přípustných a nepřípustných hodnot, které se mohou zadávat. Testování se provádí ve většině případů manuálně, ale pro některé typy testů může byt použita automatizovaná varianta. (Čermák, 2010a)

Tento typ testů se využívá k ověření nastavení jednotlivých přístupů, a tím zároveň ověřit aktuální zranitelnost aplikace/systému bez znalosti kódu. Dále je možné demonstrovat chyby, již objevené po whitebox testech. Výhodou této metody je triviálnost těchto testů, jelikož nevyžadují znalost zdrojového kódu a tím i rychlost při provádění těchto testů. Test je pak mnohem snáze interpretovatelný pro netechnicky zaměřené osoby. Nevýhodou je převážně chybějící kontrola kódu především v jeho efektivnosti a nemožnost odhalení nežádoucího chování aplikace na pozadí. (Čermák, 2010a)

3.2.3.3 Greybox

Greybox technika kombinuje black a white box. Kombinujeme znalost datových a programových struktur, které budou sloužit k vytvoření testovacích scénářů pro black box testy. Samotné testování je na stejném principu, tedy tester částečně zná vnitřní struktury, ale zároveň je nezná dostatečně do hloubky. Testování se provádí na úrovni blackboxů. (Čermák, 2010b)

Tento typ testů se většinou využívá k odhalení zranitelných míst programu, testování vícevrstvé aplikace, či testování nejrůznějších html formulářů a skriptů. Výhodou těchto testů je kombinace obou testovacích přístupů, tester nemá přístup přímo ke zdrojovému

kódu, ale zároveň je možné vytvářet testovací scénáře na vysoké úrovni porozumění. Nevýhody jsou stejné jako u black box testů, tedy neschopnost zamezit nežádoucímu kódu a nemožnost otestování všech datových toků. (Čermák, 2010b)

3.2.4 Pozitivní a negativní testy

V případě testování nás na začátku zajímá především cíl testu a ten může být buď pozitivní (splnění), nebo test negativní (selhání). Oba tyto testy se často prolínají a nacházejí se v jedné skupině testů.

3.2.4.1 Pozitivní test

V případě pozitivního testu je hlavním cílem dokázat funkčnost testované aplikace/systému a jeho soulad s business analýzou. Zajímají nás pouze akceptovaná data a k nim relevantní výsledky testů, které skončí v souladu s požadavky. (Patton, 2002)

3.2.4.2 Negativní test

Cílem testu selhání je ověřit, zda aplikace/systém zobrazí náležité chybové hlášky na určitých místech. Dále také ověřuje další dvě situace. První věcí je ověření stability testované aplikace/systému, zda při zadání irelevantní hodnoty nebude rušen chod aplikace a nedojde k neočekávanému stavu, který by mohl například vést k pádu. Druhou věcí, která se ověřuje, je případ, zdali se program nedostane do chybového stavu v jiných případech než těch, ve kterých je tento stav vyžadován. V průběhu těchto testů se dbá zvýšené opatrnosti, zda jsou výstupní data v korektní podobě. (Patton, 2002)

3.2.5 Úroveň testování

Úrovně testování jsou tvořeny čtyřmi hlavními etapami testování po vzniku testovatelného kódu. Každá etapa je zaměřená na specifický typ testování, ať už zaměřením, či detailností testů. První fáze testování jsou na úrovni zmíněných white/grey box a pozdější se naopak ztotožňují s black box testováním.

3.2.5.1 Unit testy

Unit testy neboli testy komponent, mají za úkol ověřit funkčnost spustitelného kódu a případně najít chyby v něm obsažené. Testuje se vše od základních objektů a tříd přes větší softwarové komponenty, moduly až po celé programy a obecně platí, čím menší části kódu se testují, tím je testování efektivnější. Testování může být prováděno buď manuálně, nebo s využitím softwaru 3rd použitého jako podpora k testování kódu. Samotný proces testů může být vykonávaný nezávisle na zbytku kódu a dalších napojení na okolí. Testy se provádí s přístupem ke kódu, často za přítomnosti programátora, který nalezené bugy rovnou opravuje, jelikož většinou v této fázi se nepoužívá klasické reportování chyb, které byly nalezeny. Unit testy nemá smysl používat zpětně u již zaběhlých projektů, z důvodu velkých časových a finančních nároků a následných zákroků, které se často nesetkají s úspěchem. Proto se preferuje jejich provádění pouze v raných fázích projektu. (SmartBear, 2017)

3.2.5.2 Integrační testy

Následující fází po vykonání unit testů jsou testy integrační. Testy ověřují komunikaci jednotlivých modulů/komponent, integraci na jednotlivé systémové komponenty, hardware a napojení na interface v rámci aplikace. Druhou možností využití integračních testů je spojení dvou nezávislých modulů. Na průniku se následně provádí integrační testy. Využívá se dvou hlavních přístupů na provádění integračního testování. Prvním variantou je shora dolů neboli „Top-down“. Začíná se převážně u konečných objektů jako je menu a postupuje se směrem dolů k menším modulům. Naopak metoda zdola nahoru „Bottom up“ začíná převážně u elementárních modulů, jakou jsou drivery, a postupně přechází ze sub-modulů směrem do větších. V případě menších projektů se integrační fáze často zcela vynechává, ale je to podmíněné provedením následujících testovacích fází. (Integration testing, 2017) V rámci integrační fáze se taky využívají testy regresní (viz kapitola 3.2.7.2), smoke testy (viz kapitola 3.2.7.1) a incremental testy.

Incremental test je ověřující test, který se provádí po přidání modulu k již otestované sestavě modulů. Jeho úkolem je zajistit předchozí kvalitu dosaženou při testování i po rozšíření o nový modul. Nevýhodou těchto testů je větší časová náročnost na provádění.

3.2.5.3 Systémové testy

V rámci systémových testů se ověřuje systém jako celek, již ne pouze v rámci aplikace, ale i s napojením na okolní systémy, se kterými bude komunikovat. Stav na testovacím prostředí, by v tuto chvíli, měl v ideální případě duplikovat produkční prostředí, aby se odhalilo co nejvíce chyb, které by mohly při propojení vniknout. Testovací scénáře jsou již plně založeny na „Black box“ testování a vycházejí především z business požadavků, vytvořených požadavcích, či na rizicích. Testovací fáze integračního a systémového testování se pak často kombinují a vzniká testovací fáze označovaná jako SIT (System Integration Tests). (Jorgensen, 2013) V rámci systémové fáze se již provádějí recovery, security a performance testy.

Recovery testy

Jedná se o typ testů, které mají za úkol zjistit, zda a za jak dlouho bude systém/aplikace schopná znovu fungovat po pádu. Tomu většinou předchází chyba v systému, závada na HW platformě, výpadek proudu atd. Příkladem může být přerušení síťového kabelu a zjišťuje se, za jak dlouho se dokáže aplikace vzpamatovat po opětovném připojení.

Security testy

Testy zabezpečení mají za úkol ověřit, zda je aplikace dostatečně chráněná vůči útokům zvenku, ale i zevnitř sítě. V rámci procesu je potřeba zároveň ověřit, jestli se nedá obejít přihlašování do aplikace (úplně obejít, část oprávnění není potřeba…), zda jsou veškerá data, která do aplikace/systému vstupují a vystupují dostatečně zabezpečená a nedojde k jejich zneužití. Hesla zpracovávána v rámci struktury musí být dostatečně zašifrována a bezpečně uložena. K zabezpečení se často využívá nejrůznější šifrování, firewally, či externí SW. Na základě těchto testů se dají odhalit chyby v zabezpečení, zranitelnost systému, nežádoucí kód atd.

Performance testy

Performance testy, často označované taky jako zátěžové testy, slouží k otestování výkonu aplikace, ať už z pohledu rychlosti odezvy tak maximální zátěži. Testy se provádí zasíláním velkého množství požadavků na aplikaci/systém a pozoruje se, jak moc je ovlivněna odezva a výkon. Testy se provádějí v několika kolech, aby se eliminovaly

neočekávané výkyvy na systému. Získanými výsledky pak lze odladit hned několik věcí. První je optimalizace zdrojového kódu pro větší výkon, či vybírání ze dvou variant, který kód bude výkonnější. A to buď na celý rozsah, anebo jen optimalizaci jeho částí, které mají s výkonem problémy. Dále se dá ověřit podkladový HW, zda je nastaven v adekvátní míře pro potřeby systému, zda je dostateční konektivita do sítě, ale primárně jsou využívány na testování webových aplikací, které většinou slouží k přenosu dat, či konfiguraci databází (Jorgensen, 2013). Testy dělíme tří základních kategorií (mimo hlavní kategorie se často objevují ještě i testy jako network sensitivity, failover test, volume test, soak test atd.):

- Load test
    - Test se zaměřuje na schopnost zvládání zátěže při postupně rostoucí zátěži. Jedná se o sérii souběžných procesů, které mají za úkol simulovat skutečné uživatele. Testem se pak zjistí průměrná odezva a zvládnutelné limity, za pomoci různých typů scénářů. (ISTQB, 2012)
- Stress test
    - Test nasimuluje velké zatížení systému a testují se jednotlivé funkce, které mají za úkol provádět základní operace, jako jsou výpočty, zápisy a podobně. Test má za úkol odhalit limit aplikace, kdy začne docházet v těchto operacích k chybám a odhalí se tak nejslabší článek celého systému. Často se používají dodatečné HW zdroje, aby se dokázalo dosáhnout dostatečného stress limitu. (ISTQB, 2012)
- Scalability test
    - Testy mají za úkol předvídat budoucí rozvoj, tak aby nedošlo k narušení bezproblémového chodu současné podoby systému. Rozšířením může například být rozšíření uživatelské základny, či větší množství přenášených dat. Na základě testů se můžou nastavit bezrizikové limity, v rámci kterých by nemělo dojít k neočekávaným problémům. Následně se taky stanovuje návrh na rozšíření HW podkladu pro případnou opětovnou stabilizaci výkonu. (ISTQB, 2012)

3.2.5.4 Akceptační testy

Celá fáze akceptačních testů se často označuje zkratkou UAT (User acceptance test). Jedná se o poslední fází testování před nasazením systému do produkčního provozu, a proto se jedná o nejdetailnější a zároveň tedy časově nejnáročnější fázi celého testování. V případě externího dodavatele je nutné si před začátkem akceptačního testování vydefinovat postup zadávání a opravování nalezených chyb z důvodu, že se testování provádí přímo u zákazníka, a to většinou interními zaměstnanci (často za podpory dodavatele, který systém dodává). Cílem testů jsou všechny součásti systému, a to jak správné výpočty, ukládání do databází tak i grafické chyby v případném interface systému. (Mili, a další, 2015) Fázi akceptačních testů, ale lze rozdělit do několika přístupů a to:

- Uživatelské akceptační testování
    - Jedná se o nejčastěji využívaný přístup a je prováděn interními zdroji firmy, u které se finální testování provádí. Testování probíhá na základě vytvořených scénářů od dodavatele SW. Testovaná data by již měla být, v ideálním případě, 1:1 s produkčním systémem, aby se eliminovaly neočekávané chyby. (Acceptance testing, 2017)
- Provozní akceptační testování
    - Akceptace systému systémovými administrátory, včetně:
        - testování zálohy/obnovy,
        - obnovení po havárii,
        - správu uživatelů,
        - úlohy údržby,
        - načítání dat a migrační úlohy,
        - pravidelnou kontrolu bezpečnostních nedostatků. (ISTQB, 2013)
- Smluvní a regulační akceptační testování
    - Smluvní akceptační testování je vykonáváno vůči smluvním akceptačním kritériím pro provoz softwaru vyvinutého na zakázku. Akceptační kritéria by měla být definována v době, kdy zúčastněné strany odsouhlasí kontrakt. Regulační testování je vykonáváno vůči

jakýmkoliv předpisům, které musí být dodrženy, jako například vládní, právní nebo bezpečnostní předpisy. (ISTQB, 2013)

- Alfa a beta testování
    - Vývojáři softwaru na zakázku nebo krabicového softwaru chtějí často dostat zpětnou vazbu od potenciálních nebo existujících zákazníků na trhu před tím, než je software uvolněn do komerčního prodeje. Alfa testování je vykonáváno v prostředí vyvíjející organizace, nikoliv však vývojářským týmem. Beta testování, nebo testování "v terénu", je vykonáváno zákazníky nebo potencionálními zákazníky v jejich vlastním prostředí. Organizace mohou pro systémy testované před tím a po tom, než jsou přeneseny na stranu zákazníka, používat i jiné termíny, jako například „podnikové akceptační testování" a „akceptační testování u zákazníka". (ISTQB, 2013)

3.2.6 Funkční a nefunkční testování

3.2.6.1 Funkční testy

Funkční testy jsou v podstatě testováním funkcí systému, která mají za úkol ověřit určitou komponentu, část kódu, nebo funkci systému. Často se také ověřuje, na základě funkční specifikace, zda danou operaci uživatel může vůbec provést. Testování se většinou provádí na základě dvou různých východisek, a to testování na základě stanovených požadavků, nebo testování založené na business procesech. (Black, 2009)

3.2.6.2 Nefunkční testy

Nefunkční testování nesouvisí se samotným spouštěním SW. V rámci testování nejsou podstatné určité funkce nebo akce, které může provádět uživatel. Zabývá se obecnými testy, jako jsou testy zátěžové a jeho substituty, security testy, které již byly zmíněné viz 3.2.5.3, ale taky například:

- Documentation test
    - Na základě dokumentace IEEE (Institute of Electrical and Electronics Engineers) se kontrolují veškeré plány a výsledky, které souvisí

s projektem jako například specifikace testovacích scénářů, reporty chyb, testovací plán atd.

- Compatibility test
    - Cílem testu kompatibility je ověřit, zda daný systém korektně spolupracuje s HW podkladem, operačním systémem, typy a verzemi databází, či jiným SW na který má být napojen.
- Reliability testing
    - Taky označované jako testy spolehlivosti se zaměřují na ověření, zda systém splňuje požadavky na spolehlivost, které byly stanované zákazníkem.
- Internationalization testing and Localization testing
    - Testování internacionalizace za úkol ověřit, že systém bude snadno přizpůsobitelný jiným jazykovým mutacím bez zásadních změn a test lokalizace má za úkol ověřit, že SW bude možno aplikovat na určitý region nebo jazyk přidáním nové komponenty a přeloženého textu do požadovaného jazyka. (Non-functional testing, 2017)

3.2.7 Automatické a manuální testování

Pro vytváření testů, které má smysl automatizovat, se vybírají procesy, či testovací scénáře, které se provádí opakovaně a u nepotřebují lidský faktor. Podmínkou automatizace jsou dobře zpracované testovací případy a vše ostatní je již v režii SW, který dokáže testy „proklikat“ a výsledky z nich porovnat s očekávaným výstupem. Zároveň dokáže z výsledků okamžitě vytvářet statistiky (Automatizované testování, 2016). Automatizovat je také možno za využití programování, a to formou vytvoření vlastního kódu nebo scriptu, ale zároveň i bez znalosti programování za využití externích aplikací, které umožňují celkový proces „naklikat“. Použití automatizace má smysl většinou pouze u určité kategorie testů, a to u nefunkčních testů (viz 3.2.4.2), smoke testů a testů regresních.

3.2.7.1 Smoke testy

Smoke testy někdy také označovány jako „zaváděcí testy“, mají za úkol ověřit základní průchod systémem, při vytvoření SW, nasazení nové funkce a vylepšení, opravení větších chyb, které již byly objeveny v rámci testování atd. Jejich hlavním cílem je ověření, že

kritické části systému jsou stále funkční a nedošlo při změně k jejich neúmyslnému poškození. Testy jsou vedeny formou testovacích scénářů a v řádu desítek i pro větší systémy, a to především z důvodu rychlého ověření, zda systém reaguje a může se zahájit regresní testování, které již je na detailnější úrovni. Úspěšné absolvování smoke testů je i jednou z podmínek uzavření systémové fáze testování. Ke smoke testům je vedena detailní dokumentace s jejich popisem, kterou funkční část/proces systému ověřují, a to především protože nedochází k jejich častým úpravám v rámci celého procesu testování. Díky stálosti těchto testů jsou ideálním kandidátem pro provedení automatizace.

Hlavními výhodami testů je jejich rychlost provedení díky malému množství a schopnost objevení chyby před zahájením detailnějšího testování, což vede k prevenci před zbytečným vykonáváním stovek testů. Nevýhodou je pak malá oblast, kterou pokrývají, a proto může nastat, že jeden z hlavnějších procesů nebude fungovat a zároveň nebyl odhalen smoke testy.

3.2.7.2 Regresní testy

V případě, že nastane jakákoliv modifikace systému, ať již de o malou změnu na kódu, či oprava nalezené chyby, může mít fatální dopad do systému a z tohoto důvodu se provádí regresní testy, aby se tyto dopady co nejdříve odhalily. V případě fixací může být tímto testováním odhalena i chyba, která byla způsobena opravou, nebo chyba, která byla díky této opravě nově odhalena. Testy se postupně provádějí s určitou prioritou, aby se nehlásily chyby v podsystémech, pokud hlavní systém není funkční. Opakování testů po každé větší opravě chyb, nebo výkonosti, či při přidání nové funkce eventuálně vylepšení, a to napříč všemi fázemi je vhodné zvolit automatizaci. Výhodou je nalezení většiny dopadů, které mohla změna ve svém důsledku způsobit a díky tomu zachování určité kvality SW. Hlavní nevýhodou je časová náročnost, a to především díky tomu, že se musí vykonávat po jakékoliv, ať už malé nebo velké, změně v systému. (Progresní a regresní testy, 2016)

Výhody a nevýhody automatických testů

Hlavní výhodou, kterou lze dosáhnout automatizací opakujících se testů je především velká časová úspora. Zároveň se samotná exekuce testů urychlí, jak již z pohledu exekuce samotných testů, tak i časové flexibility, jelikož testy můžou být spuštěny v kterýkoliv čas, bez ohledu na pracovní dobu. Další výhodou je odbourání lidského faktoru, a tedy eliminace

chyb z nepozornosti, které se můžou objevovat při testování a zároveň díky tomu snížení finanční náročnosti na exekuci testů.

Hlavní nevýhoda testů se objevuje především při změnách systému, které zasahují i do oblastí, které jsou již automatizované. Po této změně je potřeba automatizované řešení ihned aktualizovat, a to především z toho důvodu, aby SW nereportoval chyby, které jimi nejsou, a naopak nepouštěl do systému chyby, které nebude kontrovat, nebo si bude „myslet“ že jsou správně. Další nevýhodou je pak omezená skupina testů, kterou má smysl vůbec automatizovat. (Automatizované testování, 2016)

3.3 Testovací analýza

Testovací analýza by měla být součástí každého prováděného projektu, který obsahuje testování jako část harmonogramu. Testovací analýzu ve většině případů provádějí test analytici a na základě dosažených výsledků se připravují samotné testovací scénáře, které slouží jako podklady pro testování. Analýzu lze provést nejrůznějšími způsoby, které různě zohledňují vstupní podmínky pro testovací scénáře. Možné kombinace, pro tvorbu TC, můžou jít i do miliónů, a proto se jednotlivé kombinace omezují na specifické vstupní hodnoty, nebo slučování samotných TC, tak aby měli větší pole působnosti. Proto lze samotné analytické přístupy k vymezení vstupních podmínek mezi sebou kombinovat tak, aby se pokryla co možná největší oblast a limitovalo se riziko nepokrytých variant. Optimální úroveň testovacích scénářů je pak definována hranicí, kdy náklady na další detail otestování jsou již vetší než samotná oprava chyby nalezené v produkci.

3.3.1 Techniky založené na specifikaci (black box)

3.3.1.1 Třídy ekvivalence

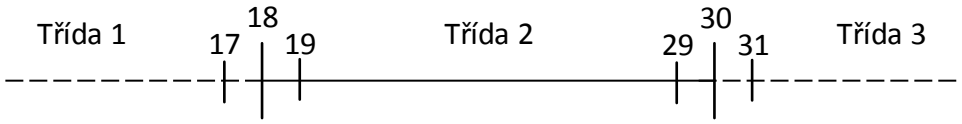
Každý testovací případ má specifický cíl, přičemž k jeho dosažení je potřeba poskytnout systému určité vstupy, ať už to jsou přímo číselné hodnoty, textové řetězce nebo kupříkladu stavy objektů. Může se stát, že existuje jen několik možných vstupů, a tak lze provést test pro každý z nich. Ovšem běžně tomu tak nebývá a jsme postaveni před problém, jak z velkého množství validních vstupů vybrat vhodná testovací data. (Roudenský, a další, 2013)

V případě jasných podmínek, jako je například pohlaví, nebo dotazníková hodnota ano/ne se tříd ekvivalence většinou nevyužívá. (Roudenský, a další, 2013) Využití se ovšem objeví v případě vzniku situace, kde lze tvořit intervaly mezi jednotlivými hodnotami. Příkladem může být automat na jízdenky, který tiskne 18 a 30 korunové kupóny. V tomto případě lze brát jako hodnotu ekvivalence $(-\infty, 18) \cup < 18,30) \cup < 30, \infty)$. Jak je vidět, je potřeba brát v potaz celý interval hodnot, a nikoliv pouze ony dvě hodnoty, které jsou zadané (mínus hodnoty jsou samozřejmě irelevantní a jsou zachovány pouze pro univerzálnost příkladu). Jak je již patrné, celá množina je rozdělena na tři třídy (je potřeba ošetřit všechny hodnoty pomocí uzavřeného a otevřeného intervalu, aby žádná nebyla z návrhu vyloučena). Po vytvoření jednotlivých tříd určení validních a nevalidních tříd a míru detailnosti TC, která bude pro návrh použita. Detailnost se obvykle vyhrazuje na jednu hodnotu na interval (počet se pohybuje většinou maximálně v jednotkách), která se navíc kombinuje s výběrem hraničních hodnot viz 3.3.1.2 Takto zvolené hodnoty se většinou směřují na prostřední hodnoty intervalu (jedna hodnota – interval dělíme na půl a vybíráme mezní hodnotu; dvě hodnoty - interval dělíme na třetiny a vybíráme hodnotu v $^1/_3$ a $^2/_3$). Metodu lze mimo číselné hodnoty využít taky na jiné typy proměnných, které jsou použity.

3.3.1.2 Analýza hraničních hodnot

Je-li definována třída ekvivalence, pak hraniční hodnoty jsou ty, které tvoří její okraje na obou stranách. Analýza hraničních hodnot neboli *Boundary values analysis*, je technika zaměřující se na testování hraničních a jejich okolních hodnot, neboť z různých důvodů jde o místo častého výskytu defektů, které navíc technika rozdělení do tříd ekvivalence nemusí odhalit. (Roudenský, a další, 2013)

Pro demonstraci odhalení problému, který může nastat na mezních hodnotách, lze vzít již vytvořený příklad automatu s třídou ekvivalence a to například z nepozornosti vývojového týmu a to v podobě záměny znaménka „>=“ za hodnotu „>“. Díky tomuto nedostatku je pak z testování na třídě ekvivalence kompletně vyřazena jedna hodnota, která je navíc jednou ze vstupních. Pro se provádí testování hraničních hodnot, které by v případě aplikace na předchozí případ vypadalo následovně.



Obrázek 6 Analýza hraničních hodnot – příklad (MS Visio)

Za speciální možnost lze brát případ, kdy hranice nelze určit, a to především v případech s neomezenou hodnotou (například není omezená maximem). Takto jsou na první pohled neomezené vstupy limitovány rozsahem použitého typu a lze je ošetřit krajní hodnotu použitého datového typu.

3.3.1.3 Rozhodovací tabulky

Rozhodovací tabulky (decision tables) jsou tabulky, které obsahují všechny kombinace podmínek, proměnných a výsledných vstupů. To je užitečné především tehdy, jestliže různé vstupy a jejich kombinace vedou k rozdílným výstupům, takže rozdělení do tříd ekvivalence by bylo příliš náročné. Běžně se proto využívají pro složitá pravidla či byznys logiku systému. (Roudenský, a další, 2013) Rozhodovací tabulka je obvykle tvořena řádky obsahujícími jednotlivé podmínky na systém a důsledky zvolení určité kombinace podmínek (ať už se jedná o kladné, nebo záporné vyhodnocení). Ve sloupcích se pak nachází jednotlivé varianty kombinací, kterými jsou tvořeny vstupní testy pro daný problém (zachována unikátnost sloupců).

V rámci vytváření podmínek se lze setkat taky s případem, který na papíře může existovat, ale samotný systém (obzvlášť komplexnější) nemusí zadání této kombinace podporovat. (Roudenský, a další, 2013) V tomto případě se TC může z rozsahu vyškrtnout na základě nemožnosti zadání podmínek do systému.

3.3.1.4 Testování přechodů mezi stavy

Technika testování přechodů stavů (state-transition testing) je zaměřena na testování stavů určené entity a přechodů mezi nimi. Cílem je potvrdit správnost specifikovaných přechodů a případně identifikovat ty neplatné anebo chybějící/nespecifikované. (Roudenský, a další, 2013)

Je důležité, aby celý proces byl z pozice testera chápán jako přechod od jednoho stavu k dalšímu při vzniku určitých událostí, změn stavů, vstupů dat, které změnu způsobí.

V rámci testování přechodů mezi stavy je identifikován stav počáteční a stav koncový, který je brán jako cíl testování. Všechny stavy jsou brány jako samostatné, identifikovatelné a celý model je tvořený konečným počtem reálných stavů, které na sebe navazují. Mimo pozitivní případy, se zpracovávají taky sady testů pro negativní testy. (Spillner, a další, 2014)

Každý diagram stavů, by pak měl vždy zobrazovat:

- Všechny jedinečné stavy, v nichž se může software nacházet.
- Vstup nebo podmínku, při které se přechází z jednoho stavu do druhého (vstupem může být například stisk klávesy, výběr příkazu z nabídky, vstupní signál ze senzoru atd.). V případě, že není žádná z podmínek naplněna, není možné přejít ze současného stavu do stavu předchozího či budoucího.
- Nastavení podmínek a generovaný výstup při vstupu v daného stavu. (příkladem lze vzít zobrazení nabídky, tlačítka, nastavení příznaků, provedení výpočtu atd.). (Patton, 2002)

Testování přechodu stavů je velmi používáno v odvětví vestavěných (embedded) softwarových produktů a v technické automatizaci obecně, nicméně tato technika je taktéž vhodná pro modelování uživatelských objektů, které mají specifické stavy nebo testování toků obrazovek. (ISTQB, 2013)

3.3.1.5 Testování podle případů užití

Případy užití (use cases) slouží k popisu určité funkcionality, respektive tedy vždy jednoho ze způsobů, jakým může být systém používán. Tento model popisuje funkcionalitu jako interakci mezi systémem a uživateli při dosahování konkrétního cíle. Soubor případů užití tak zachycuje kompletní funkčnost systému a představuje funkční specifikaci, typicky jako součást specifikace požadavků na software. (Roudenský, a další, 2013)

Případy užití jsou mnohdy dokreslovány za využití UML, kde je popsána role aktéra (osoba která přichází do styku s testovaným systémem a spouštějící příslušné scénáře. Každý aktér může testovat více scénářů, ale zároveň každý scénář může být taky testován více aktéry) v scénáři případu užití (jedná se o popis interakce aktéra a systému v podobě jednotlivých kroků, sloužících jako návod a každý případ užití může být pokryt více scénáři, které mimo pozitivní průchody testují i negativní běhy a dopady). (Spillner, a další, 2014)

Každý use case musí obsahovat určité aspekty, aby bylo možné jej prohlásit za funkční:

- Každý případ užití má ohraničení, tvořené vstupními podmínkami, které jsou tvořeny odvoditelnými nebo pozorovatelnými podmínkami a stavem finálním, který slouží pro jeho ukončení.
- Jednotlivé případy jsou obvykle tvořeny základním (nejpravděpodobnější) scénářem a alternativním scénářem. (Patton, 2002)

Případy užití popisují "procesní toky" přes systém založené na jeho pravděpodobném reálném použití, proto jsou testovací případy odvozeny z případů užití nejpřínosnější při odhalování defektů v procesních tocích v průběhu skutečného používání systému. Případy užití jsou velmi účelné pro navržení akceptačních testů za účastí zákazníka/uživatele. Taktéž pomohou odhalit integrační defekty způsobené interakcí a vzájemnou interferencí rozličných komponent, které by testování jednotlivých komponent neodhalilo. Navrhování test case z případů použití může být kombinované s jinými technikami testování založených na specifikaci. (ISTQB, 2013)

3.3.2 Techniky založené na struktuře (white box)

3.3.2.1 Testování a pokrytí příkazů

V testování komponent představuje pokrytí příkazů stanovení procenta vykonatelných příkazů, které byly vykonány sadou testovacích případů. Technika testování příkazů odvozuje testovací případy s cílem vykonat specifické příkazy, zpravidla za účelem zvýšení pokrytí příkazů.

Pokrytí příkazů je určené podílem počtu vykonatelných příkazů, které jsou pokryté (navrženými nebo vykonanými) testovacími případy, a počtem všech vykonatelných příkazů v testovaném kódu. (ISTQB, 2013)

3.3.2.2 Testování a pokrytí rozhodování

Pokrytí rozhodování se vztahuje k testování větvení a je definováno procentem výsledků rozhodování (např. větev Pravda a Nepravda IF příkazu), které byly vykonány (sadou testovacích případů). Technika testování rozhodování odvozuje testovací případy s cílem vykonat specifické výsledky rozhodování. Větve vytvářejí rozhodovací body v kódu a zobrazují přenos řízení do jiných oblastí v kódu.

Pokrytí rozhodování je určené podílem počtu všech výsledků rozhodování, které jsou pokryté (navrhnutými nebo vykonanými) testovacími případy, a počtem všech možných výsledků rozhodování v testovaném kódu.

Testování rozhodování je forma testování řídicích toků, protože sleduje specifický tok řízení přes rozhodovací body. Pokrytí rozhodování garantuje („je silnější“) pokrytí příkazů: 100 % pokrytí rozhodování garantuje 100 % pokrytí příkazů, ale ne opačně. (ISTQB, 2013)

3.4 Organizační struktura v rámci testování

3.4.1 Test manager

Jedná se o hlavního koordinátora testování, na kterého může být vymezena unikátní pozice, ale zároveň může být i zastoupen projektovým manažerem, manažerem vývoje, manažerem pro zabezpečení kvality nebo manažerem testovací skupiny. Mezi jeho hlavní povinnosti patří

- psaní a koordinace testovací politiky v rámci organizace,
- vytváření testovacích metod a testovacího plánu,
- představovat testovací záměr v rámci projektu,
- zajistit testovací zdroje,
- výběr a zavádění vhodných strategií a metod,
- zlepšování testovacích nástrojů, zajišťování vzdělávacích nástrojů a kanálů,
- rozhodování o testovacím prostředí a automatizaci testů,
- zavádění nebo optimalizace podpůrných procesů,
- představení, používání a vyhodnocování metriky, které byly definovány v rámci testovacího plánu,
- přizpůsobovat testovací plány na základě testových výsledků a průběhu testů,
- identifikace vhodných metrik pro měření průběh testu a vyhodnocování,
- zajištění kvality testování a produktu. (Spillner, a další, 2014)

Z praxe test manager tráví méně času formalizovanými činnostmi, a naopak více času je věnováno formálním a neformálním interakcím s dalšími osobami. Tyto činnosti lze rozdělit do tří hlavních skupin, a to interpersonální role (manažer jako spojka a lídr), informační role (manažer jako nervové centrum, šiřitel a mluvčí) a rozhodovací role

(manažer jako řešitel mimořádných událostí, přidělovatel zdrojů a vyjednávač). (Bureš, a další, 2016)

3.4.2 Test analyst

Hlavní rolí test analytika je identifikovat a definovat jednotlivé oblasti systému, které bude potřeba otestovat. V rámci tohoto procesu má za úkol navrhnout a dohlížet na vyhodnocování jednotlivých testů, jejich progres a kontrolování jednotlivých výstupů včetně vyhodnocení, aby byla zajištěna kvalita softwaru. (Naik, a další, 2008)

Mezi jednotlivé činnosti, které musí test analytik vykonávat, patří

- revidování a podílení se na tvorbě testovacích plánů,
- analýzu, revidování a posouzení požadavků uživatelů, specifikací a modelů testovatelnosti,
- vytváření/udržování testovací dokumentace,
- posuzování testovacích požadavků, specifikací a modelů,
- definování funkčních specifikací v rámci testovaného softwaru,
- příprava testovacích případů a dat,
- vyhodnocování výsledků testovacích případů. (Spillner, a další, 2014)

3.4.3 Tester

Hlavní náplní testera je exekuce jednotlivých testovacích případů, zaznamenávání jednotlivých výsledků testů do programů tomu určených a report chyb, které byly v průběhu testů nalezeny. Role testera bývá často pokryta ze strany test analytika, který pak zastupuje obě role zároveň. Mezi náplň testera patří

- nastavování testovacího prostředí (často koordinováno s administrací systému a síťovým managementem),
- implementaci testů na všech úrovních testování, vykonávání a zaznamenávání testů, vyhodnocování výsledků testů a dokumentování odchylek od očekávaných výsledků,
- používání nástrojů pro administraci a řízení testování a nástrojů pro monitorování testování podle potřeby,
- automatizování testů (může být podpořeno vývojářem nebo expertem pro automatizaci testů),

- měření výkonu komponent nebo systémů (když je to vhodné),
- revidování testů vytvořených jinými testery/test analytiky. (ISTQB, 2013)

3.5 Testovací plán

Testovací plán je hlavní dokument projektu. Vzniká před jeho započetím a slouží jako jeho hlavní osnova (mnohdy je část testovacího plánu použita na více projektů). Dokument je poskytnut všem stranám, které můžou být nějakou jeho částí zasaženi. Testovací plán je obvykle tvořený čtyřmi částmi:

1. Zásady testování (test policy) – zde je popsán hlavní cíl a celkový přístup organizace k samotnému testování.
2. Testovací strategie (test strategy) – vychází z předchozího dokumentu a podrobněji popisuje požadavky a způsoby, jakými bude testování prováděno. Obvykle je psán na dostatečné univerzální míře, ale ho bylo možno použít na větší skupině projektů.
3. Hlavní plán testování (Master test plan) – jedná se již o popis specifického projektu, především o aplikaci testovací strategie na problémy, které by mohly nastat.
4. Plán pro jednotlivé úrovně testování (Level test plan) – slouží především pro rozsáhlejší projekty, kde je potřeba specifikovat jednotlivé detaily pro samotné testovací fáze (unit testy, integrační atd.). (Roudenský, a další, 2013)

Každý projekt má určité požadavky a řadu omezení. Mezi základní omezení patří čas, cena a kvalita. Bohužel nelze vytvořit kombinaci, která by brala to nejlepší z každého vrcholu trojúhelníku (viz Obrázek 7 neboli vysoká kvalita, rychle dodání a nízká cena), a proto je potřeba si v rámci projektu, vždy vytyčit určitý kompromis, který bude popisovat upřednostněnou kombinaci.

Obrázek 7 Trojúhelník kvality (cncaps, 2016)

3.5.1 Základní náležitosti testovacího plánu

Testovací plány se můžou vesměs lišit, ale základní struktura zůstává většinou stejná. Mezi základní náležitosti, které by měly být popsány v rámci jakéhokoliv testovacího plánu, patří:

- Stanovené cíle a rozsahu testování
  - Zde je popsán samotný produkt, který má být testován, a záměr a cíl testování. Mimo to je nutné evidovat požadovanou funkčnost a očekávanou kvalitu softwaru. (Patton, 2002)
- Lidé, místa a věci
  - Je potřeba vydefinovat skupinu lidí, kteří se budou na testování podílet, včetně jejich odpovědnosti a kontakt. Pod místem je rozuměno místo pro ukládání jednotlivých dokumentů, stažení nové verze softwaru, či umístění jednotlivých testovacích nástrojů. Věci pak popisují kde a jak získat hardware, který by mohl být dodatečně potřeba pro testování specifických modulů. (Patton, 2002)

- Názvosloví
    - Stanovení obecných základních pojmů, které budou využívány v průběhu projektu celým realizačním týmem. Omezuje se tak neznalost určitých zkratek či pojmů. (Patton, 2002)
- Vzájemné povinnosti mezi skupinami
    - V rámci projektu může figurovat hned několik skupin (vývojáři, architekti, analytici, testeři, vedení projektu atd.), a proto se vyjasňují povinnosti každé skupiny. Nejklasičtější řešením je vytvoření tabulky s přehledem povinností pro skupiny. (Patton, 2002)
- Předmět testování
    - Zde se definuje priorita pro otestování individuálních oblastí softwaru, případně zda se vůbec budou testovat. Pokud se určitá část netestuje, zde by mělo být podáno vysvětlení proč. (Patton, 2002)
- Fáze testování
    - Jsou popsány jednotlivé testovací fáze, často odvíjející se od zvoleného modelu testování. Každá fáze musí mít pevně stanovena, za jakých podmínek může skončit a začít následující. (Patton, 2002)
- Strategie testování
    - Vydefinování, jak bude testování prováděno (postupy, nástroje, přístupy, techniky) a to jak pro celý projekt, tak i pro jednotlivé fáze testování. (Patton, 2002)
- Požadavky na prostředky
    - V rámci požadavků a prostředků je potřeba určit jednotlivé potřeby na osoby (kolik lidí a s jakou kvalifikací), vybavení (hardwarové vybavení), prostory a laboratoře (umístění prostor, jak mají být veliké a na jak dlouho budou potřeba), software (určení programů, nástrojů a databází potřebných pro realizaci), externí společnosti (bude třeba poptat externí zdroje?) a další potřeby (telefony, školení, materiály atd.), které budou za potřebí pro úspěšnou realizaci projektu. (Patton, 2002)

- Pověření testerů
    - V této sekci již dochází k upřesnění jednotlivých povinností (realizaci přidělených úkolů atd.) na konkrétní pracovníky na projektu. Každý člen pak musí mít přístup ke kompletním informacím pro úspěšnou realizaci zadaného úkolu. (Patton, 2002)
- Harmonogram
    - Jedná se o obtížnou součást, která má za úkol přenést předchozí informace do časového rozvrhu. V rámci harmonogramu je nutno brát v potaz obtížnost jednotlivých fází i míru rizika ke zdržení. Dále potřeba identifikovat potřebu personálních zdrojů v jednotlivých etapách testování (rozdílné pro analýzu, přípravu a samotnou exekuci). Podle vytvořeného přehledu se často pak uzpůsobuje celkový harmonogram projektu. (Patton, 2002)
- Metriky
    - Slouží ke stanovení postupu a úspěchu testování a celého projektu (celkový počet chyb za den, seznam chyb čekajících na opravu, aktuální chyby dělené podle závažnosti atd.). (Patton, 2002)
- Rizika
    - Je důležité, definovat jednotlivá rizika, která by mohla v rámci projektu nastat, a to i včetně odhadnuté pravděpodobnosti nastání daného problému a jejich dopadů, na zbytek testovaného softwaru. Za identifikaci rizik jsou pak odpovědní samotní členové testovacího týmu. (Patton, 2002)

3.6 Kvalita a použitelnost softwaru

3.6.1 Kvalita

Kvalita softwaru je určována na základě stanovených měřítek a kritérií a její měření probíhá v rámci celého cyklu projektu, a to především na základě reportování od realizačního týmu. Měření by mělo probíhat především za těmito účely:

- Zjištění aktuálního stavu projektu.
- Možnost sledování postupu oproti stanovenému plánu.
- Odhalení projektových rizik.
- Základní podklad pro změny v procesu.
- Celkový dohled a řízení všech oblastí projektu.

Kvalita se dá měřit pro jednotlivé fáze projektu jak již kvalita celého produktu, tak i kvalita vývoje a testování. Díky rozdílnosti těchto oblastí je potřeba stanovit správnou metriku (maximalizovat poměr úsilí, riziko a vypovídající hodnota), na základě které bude kvalita vyhodnocována. (Roudenský, a další, 2013) Lze je tedy na základě jednotlivých metrik na:

Dělení na tvrdé a měkké metriky

Na základě přesnosti zdrojových dat se dá metriky dělit na tvrdé a měkké. Tvrdé metriky jsou založeny na přesných, objektivních hodnotách, jako je například počet nalezených defektů, počet provedených testů a atd. Měkké metriky jsou naopak založeny na subjektivních hodnoceních, jako je třeba spokojenost uživatelů.

V rámci testování je většinou využíváno tvrdé metriky, a to z důvodu pevně stanovených výsledků z realizovaných testů. Měkké pak lze spíše uplatnit při rozhodování nad celkovou funkčností systému jako celku, kdy je rozhodování již subjektivní za určitou oblast lidí. (Roudenský, a další, 2013)

Dělení metrik podle použitých dat

Vychází se především z dosažených dat získaných v průběhu testování, a to především na základě chybovosti.

Jako jednu z metrik spadající do této kategorie lze vzít metriku na základě hlášení defektů. Poskytuje striktní informace o stavu testování v podobě počtu objevených defektů, průměrné rychlosti objevování nových defektů, rychlosti jejich oprav a odhadovanou dobu kdy počet defektů klesne pod přijatelnou úroveň produkčního prostředí. Takto nalezené defekty lze pak dělit na základě spousty kategorií jako například stavu a priority defektů, rychlosti nalezení defektů, intenzitě chyb v dané oblasti, účinnosti testování, počty nahlášených defektů v čase, či cena oprav jednotlivých chyb. (Roudenský, a další, 2013)

Metriky založené na testech

Jedná se o skupinu metrik vycházejících z výsledků provedených testů. Jedná se většinou o přehled postupu a výsledku testování, který je většinou předkládán jako výstup pro management (pro management přehlednější než metriky podle použitých dat)

Většinou se rozlišují dva přístupy, a to matice pokrytí oblastí testy, která říká, zda byly testy jednotlivých částí softwaru spuštěny a s jakým výsledkem. Jedná se o stručný a rychlý přehled a stavu jednotlivých modulů. Druhou metrikou jsou výsledky testů v čase, u kterých není natolik vypovídající aktuální stav, ale spíše progres v exekuci jednotlivých testů. Na základě tohoto zjištění pak lze částečně předpokládat dobu ukončení testování. (Roudenský, a další, 2013)

3.6.2 Použitelnost

V rámci kvalitního zpracování je taky potřeba dbát na samotnou použitelnost neboli grafické znázornění softwaru pro uživatele. Je důležité zpracování na základě přístupnosti (použití dostatečného kontrastu, správná volba barev použitých prostředí, klávesové zkratky atd.) a zároveň propojení všech komponent tak, aby ovládání bylo pro samotné uživatele intuitivní. Pokud se jedná o interní aplikaci, která nebude společnost reprezentovat na venek, stačí využít jednoduchého designu bez obrázků. Přehlednost taktéž lze zvýšit správně zvoleným detailem popisu v jednotlivých úrovních. Na nejvyšších úrovních stromu není třeba uvádět dlouhé popisy, které by se měly objevit až u detailnějších funkcí (např. tooltips) a chybových tabulek, a to především pro zachování jednoduchosti ovládání. Dále je potřeba zvážit samotné rozmístění prvků (při velkém počtu zvážit jednotlivé skupiny), v rámci interface softwaru. Testování použitelnosti je pak ideální provádět přímo na cílové skupině, který by měla se softwarem pracovat pro jeho nastavení „na míru“ uživatelům. (Krug, 2010)

4 Vlastní práce

V rámci praktické části bude využito podkladů z reálného projektu v oblasti bankovnictví. Z důvodu zachování vnitřních předpisů instituce o ochraně interních dat a osobních údajů, byly vybrané údaje do určité míry anonymizovány či záměrně vynechány tak, aby mohly být publikovány v rámci modelového příkladu praktické části této práce.

4.1 Popis zvoleného systému

Zvoleným systémem je modul pro práci se stavebním spořením, tedy jeden ze základních systémů pro stavební spořitelny a bankovní instituce. Samotné stavební spoření je velmi rozsáhlý a komplexní systém s dopady a návaznostmi na spousty dalších (CRM, úvěry atd.). Z důvodu zachování přiměřené míry detailu celkového zpracování praktické části, byla vybrána pouze jedna funkcionalita celého systému, a to funkcionalita „založení stavebního spoření“. Přesněji založení stavebního spoření čistě elektronickou formou, která by měla fungovat jako nadstavba nad klasickým podáním žádosti v písemné podobě. Funkcionalita založení stavebního spoření představuje pro finanční instituce obdobného typu klíčový proces, který zajišťuje velkou měrou generování finančního zisku, a tedy prosperitu společnosti. Založení stavebního spoření tedy pro účely ilustrace funkcionality celého systému představuje ideální zkoumaný vzorek.

Tato vybraná funkcionalita byla na základě business podkladů přepracována do zjednodušeného zadání. Původní zadání je umístěno na webové adrese: http://spolecnost1.cz/projekty/stavebni_sporeni/Business_zadani_112016.docx[1] Z důvodu, že samotný projekt není v současné době v procesu realizace, bude využito simulace výsledků testování, problémů, které by v rámci projektu mohly nastat (defekty) za účelem měření kvality a tvorby reportingu projektu. Předpokladem pro zpracování hlavních i dílčích částí práce se předpokládá plná funkčnost frontendu aplikace neboli jejího grafického prostředí napříč všemi webovými platformami, správné ošetřování a kontrola vstupních hodnot, aby se zamezilo vstupu irelevantních informací do aplikace, a proto nebudou případy pro tyto testy v rámci práce nijak zahrnuty ani vyhodnocovány.

[1] Jedná se o fiktivní adresu

4.1.1 Popis funkcionality modulu ŽoSS

Hlavními údaji, které aplikace zpracovává je věk klienta (z důvodu možné povinnosti zákonného zástupce/opatrovníka, či prémie pro založené stavební spoření mladším klientům), v případě nezletilosti je potřeba uvést rodné číslo zákonného zástupce. Na základě informací aplikace určuje správný úrok, který je ovlivňován celkovou délkou spoření (trváním nad 6 let) a bude pro spoření poskytnut. Následuje volba občanství, a zda klient žádá o státní příspěvek. Vyplacení stavebního spoření je možné po 6 letech a 10 letech.

Tabulka 1 Přehled produktů stavebního spoření

ID	Produkt	Trvání	Úrok (%)
1000	PRODUKT1	6 let	0,7
1001	PRODUKT2	10 let	0,8

Tabulka 2 Výše prémie za založení stavebního spoření do určité věkové hranice

Věk	Prémie (Kč)
do 15	4500
do 18	3000
do 21	1500

4.1.2 Vstupní údaje

Následující kapitola obsahuje popis jednotlivých vstupů do aplikace, jako jsou údaje klienta, který o stavební spoření žádá (jméno, příjmení, rodné číslo atd.). Dalším vstupem je možnost volby cílové částky spoření, který však musí být větší než 30 000 korun českých.

4.1.3 Okolní prostředí a vazby modulu ŽoSS

Samotný modul pro založení stavebních spoření má pak dopady do CRM (Customer relationship management), ve kterém se klient zobrazuje po provedení přepočtu D-1 a na tisk dopisů. V rámci dopisů se pak generují dva dopisy D100 (dopis, který se posílá klientovi jako smlouva o založení stavebního spoření a obsahující základní informace. Dopis se generuje ve dvou kopiích v případě, že nedojde k elektronickému podpisu smlouvy. Kopii je pak potřeba doručit podepsanou na pobočku do 30 dní, aby nabyla právní moci) a D101 (Dopis, který se zařazuje do registru a slouží pro archivační účely banky). Oba moduly využívají k přenosu dat webové aplikace.

4.2 Testovací plán

Jako základní stavební kámen pro analýzu bude vytvořen testovací plán na základě popsaných náležitostí v kapitole 3.5.1. Následující kapitoly budou sloužit jako hlavní podklad pro celý testovaný modul pro založení stavebního spoření elektronickou formou.

4.2.1 Stanovené cíle a rozsahu testování

Cílem testování je nová metoda pro založení stavebního spoření elektronickou formou, přesněji otestování jednotlivých procesů zpracování vstupních dat zadaných do aplikace. Zpracovaná data mají pak dopady do oblasti CRM (propsání klienta do aplikace) a oblasti dopisů, kde dochází k vyvolání dalších činností týkajících se generování dopisů. Mezi hlavní cíle testování pak patří:

- Integrační testy pro ověření napojení na databázi a ověření funkčnosti modulu stavebního spoření.
- Smoke testy ověřující funkčnost webových služeb.
- Systémové testy ověřující zobrazení uživatele v CRM.
- Systémové testy ověřující vygenerování dopisů.
- UAT testy procesu zpracování vstupních dat (založení klienta, rozpoznání nezletilého klienta atd.).

Do testování nebude zahrnuto testování webového prostředí aplikace a kontrola vstupních dat. Pro testy se předpokládá jejich funkčnost. Dále nebudou prováděny performance a security testy dané funkcionality z důvodu obsažení těchto testů v jiné části projektu.

4.2.2 Lidé, místa a věci

Hlavní realizační tým za stranu testování bude tvořit test manažer, test analytik a skupina testerů. Za vývoj pak odpovídá vývojář z interního oddělení společnosti, který bude řešit opravu nalezených chyb v průběhu testů.

Tabulka 3 Test plán – přehled lidí na projektu

Jméno	Role	Kontakt
Osoba1	Projektový manažer	Telefon1 / Email1
Osoba2	Test lead	Telefon2 / Email2
Osoba3	Analytik	Telefon3 / Email3
Osoba4	Tester	Telefon4 / Email4
Osoba5	Vývoj	Telefon5 / Email5

Veškeré pracovní dokumenty se budou ukládat v rámci sharepoint úložiště na následující adrese http://spolecnost.cz/projekty/stavebni_sporeni/Zalozeni_ss_web[2].

V rámci realizovaného projektu nebude třeba využít dodatečných hardwarových komponent.

4.2.3 Názvosloví

Tabulka 4 Test plán – stanovení obecného názvosloví

Název	Popis
Typ testu	Zaměřuje se na funkční nebo nefunkční aspekty řešení, např. testy funkcionality, rozhraní, negativní testy, zátěžové testy, objemové testy atd.
Testovací úroveň	Reprezentuje úroveň integrace jednotlivých systémů v průběhu procesu testování. Mezi testovací úrovně patří: Unit testy (UT), integrační testy (SI), Systémové testy (ST), Uživatelské akceptační testy (UAT)
Defekt	Nesoulad mezi definovaným požadovaným chováním systému a jeho reálným chováním v průběhu testů.
Incident	Zaevidovaný defekt k opravě.
Workflow	Proces zaznamenávání, oprav a retestů defektů.
Change request	Požadavek na opravu, za účelem odstranění chyby.
Release	Nasazení nové verze aplikace, ve které je odstraněn seznam přiložených defektů.
Nefunkční testy	Testy zaměřené na technické aspekty řešení. Do této kategorie patří například objemové a zátěžové testy, DR/HA testy apod.
Funkční testy	Testy zaměřené na ověření funkčních požadavků na řešení.
Testovací případ	Případ popisuje postup pro otestování specifického požadavku, procesu apod. Mezi základní atributy testovacího případu patří identifikace případu, popis obsahu testu, testovací podmínky, vstupní předpoklady, výčet systémů zahrnutých do testovacího případu a očekávaný výsledek testu.

[2] Odkaz by měl být nahrazen skutečným umístěním dokumentů v rámci sharepoint organizace

4.2.4 Vzájemné povinnosti mezi skupinami

Tabulka 5 Test plán – přehled rolí a jejich povinností

Role	Zodpovědnost
Projektový manažer	Zodpovídá za celkové řízení projektu Schvaluje nasazení dodávky do produkce Akceptuje UAT testy
Test manažer	Zodpovídá za plánování, přípravu, řízení a reportování o průběhu a výsledcích testování Řídí testovací tým Koordinuje opravy defektů Účastní se projektových schůzek Vyhodnocení testování
Analytik	Podílení se na tvorbě testovacího plánu Analýza vstupů Příprava testovacích případů Koordinace testování
Tester	Exekuce testovacích případů Revidování incidentů Re-testy defektů
Vývoj	Zodpovídá za implementaci systému dle požadavků zadavatele Oprava nalezených chyb

4.2.5 Předmět testování

V rámci testování založení stavebního spoření přes webovou aplikaci, není potřeba upřednostňování oblastí testování. Testování bude rozřazeno pouze na základě testovacích fází.

4.2.6 Fáze testování

Pro otestování založení stavebního spoření přes webovou aplikaci, bude využito fáze integrační, systémové a akceptační.

4.2.7 Strategie testování

Testování bude prováděno na úrovni black box testů, na základě V-modelu, a to jedním pracovníkem na úrovni tester. Na integrační a systémovou fázi bude aplikována analýza testování přechodů mezi stavy a na jejím základě vytvořeny relevantní testovací případy. Testy v rámci fáze akceptačních testů budou navrženy na základě kombinace tříd

ekvivalence, analýzy hraničních hodnot a rozhodovacích tabulek. Vzniklé testovací scénáře budou zpracovávány manuálně a v relevantních případech automatizovány.

4.2.8 Požadavky na prostředky

Personální požadavky na zvolený projekt jsou jeden plně alokovaný tester a vývojář, který bude řešit objevené defekty v rámci testování. Kompletní přehled týmu na projektu je představen v kapitole 4.2.2. Požadavky na software v rámci projektu jsou popsány v Tabulka 6.

Tabulka 6 Test plán – požadavky na software v rámci projektu

Nástroj	Účel	Popis
SpiraTest	Řízení testů Řízení defektů	Řízení a reportování testů, správa testovacích případů a zaznamenávání výsledků. Řízení a reportování defektů.
SharePoint	Sdílené úložiště	Sdílené úložiště kde budou uchovávány všechny dokumenty.
SoapUI	Testovací program	Program na provedení testů webových aplikací
SQL Developer + MSSQL modul	Databázový program	Program sloužící na práci s databází s modulem umožňující připojení na MSSQL databázi.

4.2.9 Pověření testerů

Díky složení realizačního týmu samotných testů, který je tvořený jedním členem v rámci každé kategorie, není potřeba vydefinovat rozdělení povinností mezi jednotlivé členy testovacího týmu.

4.2.10 Harmonogram

Tabulka 7 Test plán – vypracovaný harmonogram projektu

Termín	Milník
10. 02. 2017	Výchozí verze testovací strategie – projektový tým obeznámen s hlavními body testovací strategie.
17. 02. 2017	Zahájení analýzy a příprava testovacích případů
27. 03. 2017	Validace a uzavření analýzy
03. 04. 2017	Nasazení a zahájení testování
30. 04. 2017	Uzavření testování
03. 05. 2017	Závěrečný report
08. 05. 2017	Nasazení na produkční prostředí

4.2.11 Metriky

Během exekuce testů budou zaznamenávána následující metriky:

- Protestováno (% otestovaných testovacích případů z celkového počtu).
- Celkový počet nalezených incidentů v poměru s testy úspěšnými.
- Počet nalezených incidentů podle závažností a priorit.

4.2.12 Rizika

Tabulka 8 Test plán – přehled možných rizik v rámci testování

Riziko	Možná příčina	Pravděpodobnost (N,S,V)	Závažnost (N,S,V)	Řešení a omezení rizik
Nedostatečná kvalita podkladů pro testování	Časté změny požadavků na vývoj	Střední	Střední	Revize podkladů ze strany testovacího týmu
Nedostatek kapacit	Projekty s vyšší prioritou	Střední	Vysoká	Bez řešení
Zpoždění testování	Opoždění nasazení dodávky k otestování	Nízká	Vysoká	Průběžná evidence změn v zadání a jejich dopadů
Zpoždění testování	Chybně nastavené testovací prostředí	Nízká	Vysoká	Zahájení nastavování již v průběhu analýzy a provádění průběžné kontroly
Chyby v produkci v důsledku neotestování	Opomenutí požadavků na některé funkcionality	Nízká	Střední	Validace připravených testovacích scénářů zadavatelem
Zpoždění testování	Problém s připravením testovacích dat	Střední	Střední	Zajištění dostatečné business podpory pro přípravu dat
Zpoždění testování a nekonzistentní výsledky	Problém paralelních požadavků na testovací prostředí	Nízká	Střední	Dostatečné naplánování testování, aby nedocházelo k současnému spouštění různých typů testů na testovací prostředí
Předělání logiky zpracování dat	Problém se současnou legislativou GDPR	Střední	Vysoká	Zpracovávání dat v souladu s novou legislativou GDPR

4.3 Testovací analýza

V následujících kapitolách bude vytvářena testovací analýza pro jednotlivé fáze testování (integrační, systémovou a akceptační) zvoleného systému. Analýzu je potřeba provádět separátně pro každou fázi, a to především z důvodu rozdílného cílení testů. Díky této rozdílnosti jsou použité jiné techniky analýzy na vymezení požadavků na testy. Na základě výsledků analýzy budou následně vytvořeny testovací scénáře, a to včetně jednotlivých kroků, které budou fungovat jako podklad pro testera, který zadaný systém bude mít za úkol otestovat. V rámci analýzy bude použito zjednodušeného zobrazení testovací případů, u kterých dojde k vynechání některých náležitostí jako je popis testovacího případu (bude sjednocen s názvem), vlastník testu, ID požadavku, z kterého vychází release, pro který je test určen, fáze testování, pro kterou je test určen a veškeré volitelné pole, které by mohli být k testům přidány.

Vytvořené sady testů pak podle jednotlivých fází ověřuje základní funkčnost aplikace, její správnou komunikaci s databází, interakci systémů s ostatními systémy bankovní instituce a kompletní zpracování vstupních údajů zadaných klientem a jejich správné vyhodnocení podle zadaných business procesů.

4.3.1 Integrační testy

První fáze black box testování jsou integrační testy. Jejich cílem bude ověření funkčnosti aplikace po nasazení na testovací prostředí, nebo ověřování základní funkčnosti aplikace při vydání nového releasu, který obsahuje opravy či změnu stávající konfigurace. Tuto sadu testů je potřeba provádět primárně, před začátkem ostatních testů, jelikož vyjadřují připravenost aplikace pro následující fáze testování. V případě, že testy neprojdou již na této úrovni, nemá smysl spouštět následující fázi testů. Na základě vydefinovaného zadání je potřeba vytvořit dva testovací případy, jež budou aplikaci prověřovat.

Jelikož cílem je zakládání nového klienta do systému, budou se muset zadané údaje uložit na databázové pole uvnitř banky. Díky tomu lze odvodit první testovací případ, který bude ověřovat připojení aplikace k databázi MSSQL. Testovací případ lze provádět jak manuálně, tak automaticky za využití connection skriptu. Za předpokladu, že by nebylo možné navázat připojení k databázi, nemá smysl pokračovat v exekuci následujících fází a program by musel být opraven. Druhým úskalím je varianta, kdy se tester dokáže

k databázi připojit, avšak nedokáže do databáze zapisovat či v ní upravovat, nebo číst data. Tento problém taktéž znemožňuje další postup v testování a je potřeba ho opravit.

Mimo základní připojení k databázovému prostředí je třeba ověřit spustitelnost samotné aplikace. Na základě toho lze definovat druhý testovací scénář, jehož účelem bude spuštění samotného portálu aplikace stavebního spoření a zvolení modulu pro založení stavebního spoření, který by již měl uživatele přesměrovat na samotný interface založení. Jelikož založení stavebního spoření lze považovat jako jeden modul, bude z tohoto důvodu zvolen pouze jediný testovací případ. Pokud by se však v rámci testování došlo k rozšíření množství modulů, je potřeba vytvořit test pro každý modul zvlášť, aby bylo dosaženo komplexního otestování. Toto otestování je z hlediska nasazení zcela nezbytné.

Oba testovací případy lze považovat jako smoke testy (viz 3.2.7.1), a proto budou prováděny na začátku každé závažnější změny provedené na aplikaci (mimo další release aplikace lze na základě dohody zahrnou i hotfixy po předchozí domluvě s vedením projektu).

4.3.1.1 Návrh testovacích případů

ID	Název TC	Priorita	TS	Popis	Očekávaný výsledek	Vstupní data
1	Kontrola funkčnosti portálu pro založení SS	Vysoká	1	Kontrola, zda se načte portál pro stavební spoření	Portál se načetl	ss.firma1.cz[3]
			2	Vebere možnost založení SS	Otevřel se portál pro založení SS	
ID	**Název TC**	**Priorita**	**TS**	**Popis**	**Očekávaný výsledek**	**Vstupní data**
2	Kontrola napojení na databázi a vyhledávání v ní	Vysoká	1	Kontrola, zda se SQL Developer připojí k databázi	Spojení navázáno	Veškeré informace potřebné k připojení jsou ve složce databáze na SP (nutné oprávnění)
			2	Tester vyhledá testovacího uživatele v tabulce SS	Uživatel nalezen	RC: 1234567890

[3] Jedná se i fiktivní adresu portálu pro stavebního spoření

4.3.1.2 Automatizace TC pro SI

V rámci integračních testů lze automatizovat oba uvedené testovací případy, které mají za úkol ověřovat dostupnost určitých částí systému. Z důvodu oprávnění a častého opakování těchto testů je automatizace vítána. Obě automatizace budou provedeny přes PHP v podobě kontrolního skriptu, který navrací hodnotu, zda test byl či nebyl úspěšný. Popis jednotlivých částí kódu je odlišen zeleným označením, přímo v těle skriptu.

První test se zaměřuje na ověření dostupnosti samotného portálu, kde bude umístěný odkaz na založení stavebního spoření. Výstupem je, zda je zadaná URL adresa dostupná.

```
<?php
:: Vydefinování kontrolované URL adresy
 $url = "google.com";

:: Kontrola dostupnosti adresy
 if ( checkdnsrr($url, "ANY") ) {
  echo "URL adresa je dostupna. ";
 }
 else {
  echo "URL adresa nebyla nalezena. ";
 }
?>
```

Druhý test ověřuje dostupnost databázové platformy MSSQL, kde následně vyhledá zadané rodné číslo. Výstupem je zda se povedlo připojit, případně zda bylo rodné číslo vyhledáno.

```
<?php
:: Vydefinování vstupních proměnných
$SQLserver = "localhost";
$SQLdatabase = "";
$SQLlogin = "";
$SQLpassword = "";

:: Nastavení rodného čísla
$rc = "1234567890";

:: Pokus o připojení k databázovému serveru
$dbc = mysqli_connect($SQLserver, $SQLlogin, $SQLpassword) or die("Nepodarilo se pripojit k
databazovemu serveru.");

:: Nastavení znakové sady pro komunikaci a připojení k DB
mysqli_query($dbc, "SET NAMES utf8");
mysqli_select_db($dbc, $SQLdatabase) or die("Nepodarilo se vybrat databazi.");
```

```
:: Select, který vyhledává zadané rodné číslo a vyhodnocení zda byl nalezen
$query = "SELECT * FROM rc WHERE rc LIKE '".mysqli_real_escape_string($dbc, $rc)."'";
$result = mysqli_query($dbc, $query);
if ($rc = mysqli_fetch_row($result))
{
        echo "Zaznam nalezen";
} else {
        echo "Zaznam nenalezen";
}
?>
```

4.3.2 Systémové testy

Po ukončení fáze integračních testů (podmínkou je úspěšné spuštění testů), se spouští druhá fáze testování označována také jako ST neboli systémové testy. Testy mají primárně za úkol ověřit interakce jednotlivých systémů, které jsou navzájem propojené. Na základě zadání (viz 4.1) se bude jednat o napojení na sytém CRM a systém pro zpracování a tisk dopisů, a to vždy ve směru ze stavebního spoření. Opačná komunikace není cílem testování. Úspěšnost systémové fáze je důležitým krokem pro započetí testů akceptačních.

Samotné testování bude tvořeno třemi až pěti testovacími případy (důvod tohoto počtu je popsán níže). Pokud se bude jednat o delší spojené testy, v rámci stejné oblasti, budou se exekvovat pouze tři. V případě, že však každý test bude mít vlastní TC, bude potřeba připravit pět TC. Pro zachování větší přehlednosti bude zvolena varianta obsahující tři testovací případy, avšak bude popsáno všech pět případů.

První testovací případy budou ověřovat zobrazení založeného klienta v CRM a budou se kontrolovat všechny propsané informace včetně jejich umístění do správných polí. Pro scénář budou připravovány testovací data formou fiktivního klienta, jehož rodné číslo bude vstupem do testovacího případu.

Stejný vstup bude zároveň využit pro ověření jedné sady vygenerovaných dopisů pro klienta. Pro druhou sadu bude třeba opět připravit nová testovací data tak, aby se ověřily varianty, kdy klient podepsal a nepodepsal smlouvu online (pokud nepodepsal, generují se dva stejné dopisy D100 za sebe).

Mimo scénáře, které budou kontrolovat propsání informací do těchto dvou systémů, bude potřeba kontrolovat webové aplikace, které slouží k přenosu jednotlivých dat pomocí XML tvaru. Čtvrtým testovacím případem bude webová aplikace ověřující přenesení informací z online prostředí do databáze a do dopisového systému. Poslední test opět ověřuje

webovou službu, přesněji přenos dat ze systému stavebního spoření do systému CRM. Oba tyto testy budou exekvovány na základě připravených projektů pro program SoapUI, který slouží k testování webových služeb.

4.3.2.1 Návrh testovacích případů

ID	Název TC	Priorita	TS	Popis	Očekávaný výsledek	Vstupní data
3	Kontrola zobrazení klienta v CRM	Vysoká	1	Tester otevře aplikaci CRM (nutno počkat den po založení)	Spojení navázáno	RC: 1234567890 COD
			2	Zadání RC do příslušného pole a kliknutí na vyhledat	Klient byl vyhledán	
			3	Otevřít systém SS a vyhledat klienta podle RC	Klient byl vyhledán	
			4	Provést kontrolu jednotlivých údajů (jméno, datum narozeni atd.)	Údaje se shodují a jsou umístěny ve správných polích	

Následující testovací případ lze rozdělit na dva případy pro kontrolu každého RC zvlášť. Rozdíl mezi RC je pouze v tom, zda byla smlouva podepsána online či nikoliv

ID	Název TC	Priorita	TS	Popis	Očekávaný výsledek	Vstupní data
4	Kontrola vygenerování dopisu D100 a D101	Vysoká	1	Tester otevře systém správy dopisů	Systém se načetl	RC: 1234567890 07. 04. 2017 RC: 9876543210 07. 04. 2017
			2	Zadání RC do příslušného pole a kliknutí na vyhledat	Klient byl vyhledán	
			3	Ověření vygenerování dopisů pro datum založení obou klientů	Dopisy pro obě RC se v uvedeném datu vygenerovaly	

	4	Otevřít dopis D100 a provést kontrolu podle šablony a údajů klienta	Dopis obsahuje všechny náležitosti a správné údaje	Šablona viz složka dopisy na SP
	5	Otevřít dopis D101 a provést kontrolu podle šablony a údajů klienta	Dopis obsahuje všechny náležitosti a správné údaje	Šablona viz složka dopisy na SP

V případě, že by pro následující testovací případ nebyla předvyplněná data, bylo by potřeba kontaktovat operátory sítě a vstupní data získat. Testování webových služeb probíhá za předpokladu, že pro účely testování není vyžadováno žádných speciálních účtů, certifikátů ani oprávnění.

ID	Název TC	Priorita	TS	Popis	Očekávaný výsledek	Vstupní data
5	Kontrola vygenerování dopisu D100 a D101	Vysoká	1	Tester otevře aplikaci SoapUI a naimportuje připravené projekty	Projekty se načetly	Projekty jsou místěny na sdílením disku G:\testovani\SS\ zalozeni\web_A[4]
			2	Spouštění Funkce1 projektu1	Funkce korektně doběhla	
			3	Spouštění Funkce1 projektu2	Funkce korektně doběhla	
			4	Spouštění Funkce2 projektu2	Funkce korektně doběhla	
			5	Spouštění Funkce3 projektu2	Funkce korektně doběhla	

[4] Jedná se i fiktivní úložiště, které bude využito v nastavení automatizace

4.3.2.2 Automatizace TC pro ST fázi

Ze zpracovaných testů má smysl automatizovat pouze testy pro webové služby, a to pouze nedestruktivní funkce, protože ostatní testy kontrolují dotažení určitých údajů a korektnost textací v případě dopisů. Kvůli složitosti automatizace těchto testů by bylo dosaženo téměř nulové efektivnosti celého řešení, a proto nejsou pro automatizaci vhodné a nebudou zahrnuty.

Pro minimalizaci požadavků na další programy, byla automatizace webových služeb provedena přímo za využití programu SoapUI 5.0 a příkazové řádky operačního systému Windows. Výsledný test bude spouštěný formou batch souboru, jehož podoba je vidět níže. Podmínkou pro úspěšnou automatizaci je nainstalovaná poslední verze Java, kterou využívá program SoapUI. V rámci kódu není potřeba zohledňovat jednotlivé funkce, ale pouze celé nastavené projekty, u kterých se provedou všechny obsahující metody které má. Samotný popis jednotlivých částí, je označen zelenou barvou přímo ve zdrojovém kódu batche.

```
@echo off

:: Umístění projektů webových služeb, které mají být testovány
set pathFolder="G:\testovani\SS\zalozeni\web_A
:: Místo na disku kam se mají ukládat reporty z testování
set report="ZMENIT_NA_ADRESU_PRO_ULOZENI_REPORTU"
:: Kompletní cesta k programu SoapUI, přesněji umístění souboru testrunner.bat
set testrunner="C:\Program Files\SmartBear\SoapUI-5.0.0\bin\testrunner.bat"

:: Nastavení data do názvu reportu ve tvaru YYYY_MM_DD
set year=%date:~6,4%
set month=%date:~3,2%
set day=%date:~0,2%
set mydate=%year%_%month%_%day%

:: Nastavení zbytku cesty projektu pro každou web app zvlášť
set Projekt1=%pathFolder%projekt1_service_soapui_test.xml"
set Projekt2=%pathFolder%projekt2_service_soapui_test.xml"

:: Exekuce nastavených projektů a vytvoření reportu z testování
ECHO running Projekt1
call %testrunner% -r>%report%%mydate%_projekt1_summary.txt -f%report% -I %Projekt1%
ECHO running Projekt2
call %testrunner% -r>%report%%mydate%_projekt2_summary.txt -f%report% -I %Projekt2%

:: Ukončení batche
PAUSE
```

4.3.3 Akceptační testy

Po dokončení předchozích dvou fází, lze započít poslední testovací fázi, která je zároveň i fází nejrozsáhlejší. Fáze označovaná jako UAT detailně testuje jednotlivé zpracování dat v systému ze vstupních formulářů. Jedná se o nejdelší fázi celého testování, ve které je zároveň největší šance odhalení chyb. Cílem je otestovat jednotlivé kombinace pro založení nového klienta do stavebního spoření. Jelikož nelze ověřit každou kombinaci, která by mohla nastat, je cílem vytvořit shluky kombinací vstupních podmínek tak, aby bylo otestováno co největší procento všech reálných kombinací na co nejmenším počtu testovacích případů. Pro vytvoření těchto testů je potřeba vytvořit analýzu jednotlivých vstupních hodnot. Základem bude analýza tříd ekvivalence a hraničních hodnot jednotlivých vstupů.

V první fázi je potřeba vydefinovat jednotlivé vstupy a výstup, které budou do analýzy vzaty. Na základě kapitoly 4.1 lze určit vstupy jako je věk klienta, který stavební spoření zakládá, zda má nebo nemá klient zákonného zástupce, zda klient požádal o státní příspěvek ke svému spoření, jakou zvolil klient cílovou částku, zvolená délka spoření jakou si klient vybral a poslední parametr, jestli se jedná o cizince či nikoliv. Jako hlavní výstup pak lze brát, zda bylo stavební spoření založeno či nikoliv. Mimo tento primární údaj je potřeba také zpracovat, zda klientovi byla připsána prémie za založení stavebního spoření a určení její výše, na základě volby produktu (trvání) je pak určováno, jak velký bude poskytnut pro stavební spoření úrok. Posledním výstupem je schválení státního příspěvku, zda byl či nebyl schválený (jelikož cizinci nemají u stavebního spoření na státní příspěvek nárok).

Seznam vydefinovaných je potřeba nyní rozdělit na dvě skupiny: ty, na které lze aplikovat analýzu tříd ekvivalence a hraničních hodnot a ty, které nabývají pouze dvou až tří hodnot a je tedy zbytečné pro ně analýzu vytvářet. Díky tomu lze vypreparovat proměnné věk a částka, jenž obě můžou nabývat velkého počtu hodnot, a proto pro ně bude potřeba vytvořit detailnější analýzu vstupních hodnot pro tvorbu testovacích případů. Ostatní proměnné nabývají pouze dvou kombinací vstupu, viz Tabulka 9.

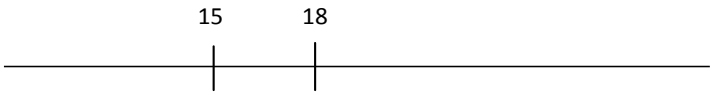
Tabulka 9 UAT – tabulka vstupů pro TC bez proměnných věk a částka

Vstupy	Relevantní hodnoty	
Věk	Bude doplněno	
Zákonný zástupce	ANO	NE
Státní příspěvek	ANO	NE

Vstupy	Relevantní hodnoty	
Částka	Bude doplněno	
Cizinec	ANO	NE
Délka spoření	6 let	10 let

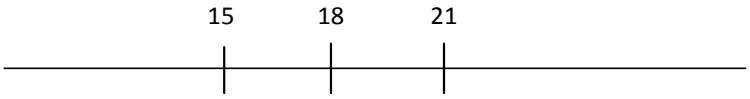
Věk

První definovanou proměnnou je „Věk“. Úkolem je stanovení jednotlivých hraničních hodnot, které budou doplňovat Tabulka 9. Základem je vymezení všech proměnných, které jsou na věk navázány, a to včetně vstupů tak výstupů, aby se předešlo vynechání důležitého milníku na otestování. První proměnnou, která bude věk segmentovat, je zákonný zástupce, či opatrovník. Prvním bodem bude věk 18 let, kdy se klient stává plnoletým a zákonný zástupce již nemusí být u zakládání smluv uveden. Jako druhý bod pak lze vzít den 15. narozenin, který znamená zisk občanského průkazu, avšak stále zůstává povinnost zákonného zástupce. Na těchto údajích lze vytvořit základní osu (Obrázek 8), do které budou doplňovány další údaje.



Obrázek 8 UAT – třída ekvivalence pro proměnnou věk se zahrnutím zákonného zástupce

Další a zároveň poslední proměnnou, kterou je potřeba vzít v potaz patří již mezi výstupy a jedná se o bonus „Prémie za založení“. Prémie se vyplácí rozdílně do 15 let, do 18 let a do 21 let. Nad 21 let se následně již prémie za založení nevyplácí (důležité brát i tento interval). Na základě zadání se bere „Do“ jako otevřený interval, a tudíž se daný věk do intervalu pro vyplacení nezahrnuje (do 15, bere všechny hodnoty 14 a méně). Jelikož věk 15 a 18 již na ose jsou vynesené, není potřeba je brát v potaz a novou hodnotou bude pouze věk 21 (nový tvar osy lze vidět na Obrázek 9).



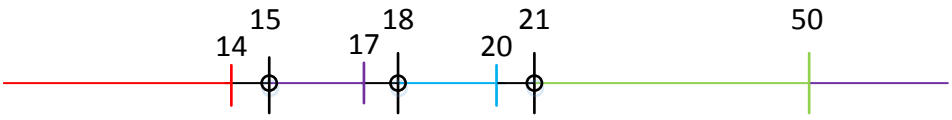
Obrázek 9 UAT – třída ekvivalence pro proměnnou věk se zahrnutím zákonného zástupce a prémie za založení

Tímto byly zahrnuty všechny proměnné obsahující věk a lze tedy na základě tohoto podkladu vytvořit analýzu hraničních hodnot (kapitola 3.3.1.2), které budou do testovacích případů zahrnuty. Prvním počinem je doplnění maximálních a minimálních hodnot, ale jelikož většina zanesených bodů nabývá nízkých hodnot, bude doplněna pouze hodnota

vysoká, která může reálně nastat, tedy věk nad 50 let. Mimo extrémy je potřeba zohlednit hodnoty kolem vytyčených bodů, a to z důvodu, že v těchto číslech je velká náchylnost ke vzniku defektu. V Tabulka 10 a Obrázek 10 je již zobrazena finální podoba hodnot, které budou výstupem analýzy.

Tabulka 10 UAT – analýza hraničních hodnot pro proměnnou věk

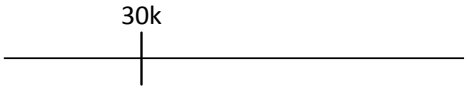
Vstupy	Relevantní hodnoty							
Věk	14 a méně	15	15–17	18	18–20	21	21–50	50 a víc



Obrázek 10 UAT – analýza hraničních hodnot pro proměnnou věk (MS visio)

Částka

Pro proměnnou „Částka“ bude aplikován stejný postup pro získání hodnot, jako byl použitý pro proměnnou věk. První fází bude vytvoření třídy ekvivalence na základě zadání. Jelikož je jediným omezením informace, že cílová částka musí při založení přesáhnout hodnotu 30 tisíc, bude podoba osy pouze s jednou hodnotou, viz Obrázek 11 (mimo toto omezení není pro tuto analýzu brát v potaz další bod, který lze určit jak na základě vstupů, tak na základě výstupů).

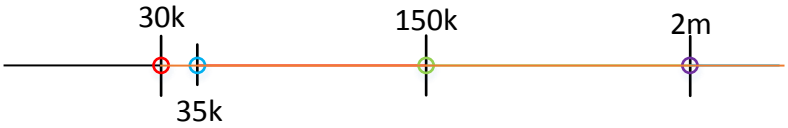


Obrázek 11 UAT – třída ekvivalence pro proměnnou částka

Do grafu tříd ekvivalence budou doplněny body na základě vytvoření analýzy hraničních hodnot, které by mohly odhalit potencionální defekty při exekuci testovacích případů. Prvním bodem mimo 30k bude přidána částka těsně nad touto hodnotou. Dále bude přidána mezní a vysoká částka, jenž zajistí testování většího spektra hodnot, a to v podobě intervalů. Proto bude přidána hodnota 150k, jakožto zástupce pro oblast často volených částek a částka 2 milionů jako zástupce částky vysoké. Mimo tyto hodnoty můžou být body doplněny intervaly pro detailnější pokrytí testy. Výsledek je zaznamenám v Tabulka 11 a Obrázek 12 kde je vytvořený kompletní přehled všech vstupních hodnot pro testovací případy.

Tabulka 11 UAT – analýza hraničních hodnot pro proměnnou částka

Vstupy	Relevantní hodnoty			
Částka	30k	35k	150k	2M



Obrázek 12 Obrázek 7 UAT – analýza hraničních hodnot pro proměnnou částka (MS visio)

Z takto vydefinovaných vstupů lze vydefinovat kompletní tabulku hodnot, které budou sloužit jako vstup pro jednotlivé testovací případy pro fázi akceptačních testů.

4.3.3.1 Návrh testovacích případů

Tvorba TC doplněním rozhodovací tabulky

Testovací případy lze vydefinovat doplněním rozhodovací tabulky tak, aby každá hodnota byla vždy otestována minimálně jednou, viz Tabulka 12. Tabulka byla, mimo proměnnou věk, doplněna na základě pravděpodobnosti výskytu pro jednotlivé hodnoty. Ke vstupům je pak potřeba nadefinovat i relevantní výstupy, které bude muset tester ověřit v rámci korektního absolvování testovacího případu. Mimo to je potřeba zohlednit žádosti podávané cizinci, kteří nemají nárok na získání státního příspěvku pro stavební spoření, a tedy i v případě, že o něj ucházejí, musí být žádost na příspěvek posouzena negativně. Z důvodu většího počtu testovacích případů, které jsou určeny pro testovací fázi akceptačních testů, bude vytvořen pouze jeden ukázkový testovací případ.

Tabulka 12 UAT – rozhodovací tabulka pro testovací případy

Vstupy								
TC	1	2	3	4	5	6	7	8
Věk	14 a méně	15	15-17	18	18-20	21	21-50	50 a víc
Zákonný zástupce	ANO	ANO	ANO	NE	NE	NE	NE	NE
Státní příspěvek	ANO	ANO	ANO	NE	NE	ANO	ANO	ANO
Částka	35k	2M	30k	150k	150k	2M	150k	35k
Cizinec	NE	NE	ANO	NE	ANO	NE	NE	NE
Délka spoření	6 let	10 let	6 let	10 let	6 let	6 let	10 let	6 let
Výstupy								
Založeno	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Prémie za založení	4500	3000	3000	1500	1500	0	0	0
Úrok	0,7	0,8	0,7	0,8	0,7	0,7	0,8	0,7
Státní příspěvek	ANO	ANO	NE	NE	NE	ANO	ANO	ANO

Tabulka 13 UAT – ukázkový testovací případ pro akceptační testy

ID	Název TC	Priorita	TS	Popis	Očekávaný výsledek	Vstupní data
6	Testování vstupních hodnot pro založení SS	Střední	1	Založit testovacího uživatele na základě zadaných hodnot.	Uživatel založen.	Věk: 14 a méně ZZ: ANO Státní pří.: ANO Částka: 35k Cizinec: NE Délka: 6 let
			2	Vyhledat založeného uživatele podle RČ, nebo přiděleného ID.	Uživatel nalezen.	
			3	Zkontrolovat údaje po založení, zda odpovídají výstupním hodnotám.	Všechny výstupní údaje odpovídají.	Založeno: ANO Prémie: 4500 Úrok: 0,7 Státní pří.: ANO

Tvorba TC za využití programu allpairs

Druhou možností pro lepší pokrytí celého systému je zvolit více kombinací. Lze však použít pouze všechny relevantní kombinace (negativní testy by neprošly přes webový formulář, a tudíž není potřeba tyto testy zahrnovat), které by v případě nezletilých tvořily 96 možných kombinací a pro zletilé 160 testovacích případů. I přes to, že počet nevstupuje do tisíců, či dokonce milionů, je celkový počet příliš vysoký pro tak malý testovací tým za vymezený časový úsek (s předpokladem retestů některých TC z důvodu defektů).

Proto lze využít program pro tvorbu kombinací jednotlivých testovacích scénářů, jehož úkolem je vytvoření testovací případy, pro co největší pokrytí systému testy. K tomuto účelu byl využit již napsaný program „Allpairs“ (Bach, 2016). Pro určení vstupu do programu bylo použito předchozí analýzy hodnot a byl rozdělen na dvě kategorie, bez zákonného zástupce a se zákonným zástupcem. Vstupy pro zpracování zachycují

Tabulka 14 Vstup pro program allpairs – bez ZZ

Věk	Zákonný zástupce	Státní příspěvek	Částka	Cizinec	Délka spoření
14 a méně	ANO	ANO	30k	ANO	6 let
15		NE	35k	NE	10 let
15-17			150k		
			2M		

Tabulka 15 Vstup pro program allpairs – se ZZ

Věk	Zákonný zástupce	Státní příspěvek	Částka	Cizinec	Délka spoření
18	NE	ANO	30k	ANO	6 let
18-20		NE	35k	NE	10 let
21			150k		
21-50			2M		
50 a víc					

Program vstupy zpracuje a na základě algoritmu sjednocuje jednotlivé testovací případy tak, aby byla zachována velká míra pokrytí testů. Výstupy jednotlivých běhů programu jsou pak uvedeny v příloze (viz Příloha 1 a Příloha 2), kde lze najít jednotlivé vygenerované kombinace i seznam testovacích případů, ve kterých byla proměnná použita. Z první části výstupu je možno určit počet propojených testovacích případů na základě indexu „pairings“. Čím větší je index pairings, tím více scénářů bylo spojeno, a zároveň tím je důležitější, aby byl testovací případ zahrnutý do testování. Testy s indexem 1 a 2 pak jsou téměř unikátní neboli pokrývají minimální oblast pro testování. V rámci dalšího snižování počtu testů na exekuci lze tyto testy vyřadit se zachováním velkého procenta pokrytí testovaného systému. Takto selektované případy je opět potřeba převést do rozhodovacích tabulek pro zachování přehlednosti a zároveň doplnit relevantní výstupy pro kombinaci vstupních podmínek tak, aby testovací případ odpovídal specifikaci.

Pro testovací případy bez zákonného zástupce bylo vybráno celkem 9 z 12 možných testů tak, aby byl zachovaný co nejmenší počet požadovaných exekucí s co největším pokrytím testovacího systému. Výsledně vzniklé testovací případy a jejich doplnění o jednotlivé výstupy lze vidět v Tabulka 16.

Tabulka 16 Allpairs – návrh testovacích případů bez zákonného zástupce

Vstupy									
TC ID	6	7	8	9	10	11	12	13	14
Věk	14 a méně	15	15-17	14 a méně	15	15-17	14 a méně	15	15-17
ZZ	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Stát. přís.	ANO	NE	ANO	NE	ANO	NE	NE	ANO	NE
Částka	30k	35k	150k	2M	30k	35k	150k	2M	30k
Cizinec	ANO	NE	NE	ANO	ANO	ANO	NE	NE	NE
Délka	6 let	10 let	6 let	10 let	10 let	6 let	10 let	6 let	10 let
Výstupy									
Založeno	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Prémie	4500	3000	3000	4500	3000	3000	4500	3000	3000
Úrok	0,7	0,8	0,7	0,8	0,8	0,7	0,8	0,7	0,8
Stát. přís.	NE	NE	ANO	NE	NE	NE	NE	ANO	NE

Naopak testy se zákonným zástupcem vycházely již z 20 možností, ale vybráno, na základě pairings indexu, bylo pouze 13 případů. Testy byly opět doplněny o relevantní výstupy k vybraným vstupním podmínkám.

Tabulka 17 Allpairs – návrh testovacích případů se zákonným zástupcem

Vstupy													
TC ID	15	16	17	18	19	20	21	22	23	24	25	26	27
Věk	18	18-20	21	21-50	50 a víc	18	18-20	21	21-50	50 a víc	18	21	21-50
ZZ	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE
Stát. přís.	ANO	NE	ANO	NE	NE	ANO	ANO	NE	NE	ANO	NE	ANO	ANO
Částka	30k	35k	150k	2M	30k	35k	150k	2M	150k	2M	150k	30k	35k
Cizinec	ANO	NE	NE	ANO	NE	ANO	ANO	NE	NE	ANO	NE	ANO	NE
Délka	6 let	10 let	6 let	10 let	6 let	10 let	6 let	10 let	6 let	10 let	10 let	10 let	6 let
Výstupy													
Založeno	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Prémie	1500	1500	0	0	0	1500	1500	0	0	0	1500	0	0
Úrok	0,7	0,8	0,7	0,8	0,7	0,8	0,7	0,8	0,7	0,8	0,8	0,8	0,7
Stát. přís.	NE	NE	ANO	NE	NE	NE	NE	NE	NE	NE	NE	NE	ANO

Takto vytvořených 22 testovacích případů by mělo pokrýt testovaný systém minimálně z 95 a více procent. Na základě toho bude pro další fáze (hodnocení kvality

a reporting) využito těchto vytvořených testů, pro maximalizování míry detailnosti pokrytí testovacími případy a minimalizace celkového počtu testů na ověření systému.

4.3.3.2 Automatizace TC pro UAT

Automatizace fáze akceptačních testů je velmi náročná a nákladná, především kvůli využití „botů“ pro automatizaci. Automatizovat se dá pouze určitá část testů, ale z důvodu krátkého trvání projektu a nízkého počtu testů, není automatizace v tomto případě žádoucí metodou zpracování testovacích případů.

4.4 Simulace výsledků testování

Vzhledem k tomu, že projekt ještě není ve fázi exekuce, je potřeba nasimulovat možný průběh testování jednotlivých release systému. Výsledná simulace bude poté sloužit jako vstup pro kapitolu 4.5, která bude výsledky zpracovávat a vyhodnocovat na základě zvolených metrik. Všechny hodnoty budou fiktivní a nijak nekopírují skutečné výsledky testování projektu.

Pro optimální přehlednost a dostatek vstupních dat budou provedeny tři release systému, který bude kontrolován na základě vytvořených testovacích případů z kapitoly 4.3. Jednotlivé testovací případy budou vyhodnoceny na základě stavů „Passed“, „Failed“ a „Not run“. Všechny výsledky z jednotlivých fází testování a verzí spuštění budou zaznamenány do tabulek pro zvýšení přehlednosti.

Tabulka 18 Výsledky simulace testování na základě připravených TC

TC ID	Fáze	Release 1	Release 2	Release 3
1	SI	Passed	Passed	Passed
2	SI	Passed	Passed	Passed
3	ST	Passed	Passed	Passed
4	ST	Passed	Passed	Passed
5	ST	Passed	Passed	Passed
6	UAT – bez ZZ	Passed	Passed	Passed
7	UAT – bez ZZ	Passed	Passed	Passed
8	UAT – bez ZZ	Failed	Failed	Passed
9	UAT – bez ZZ	Passed	Passed	Passed
10	UAT – bez ZZ	Passed	Passed	Passed

TC ID	Fáze	Release 1	Release 2	Release 3
11	UAT – bez ZZ	Passed	Passed	Passed
12	UAT – bez ZZ	Passed	Passed	Passed
13	UAT – bez ZZ	Failed	Failed	Passed
14	UAT – bez ZZ	Passed	Passed	Passed
15	UAT – s ZZ	Failed	Passed	Passed
16	UAT – s ZZ	Failed	Failed	Passed
17	UAT – s ZZ	Failed	Failed	Passed
18	UAT – s ZZ	Failed	Failed	Passed
19	UAT – s ZZ	Failed	Passed	Passed
20	UAT – s ZZ	Failed	Failed	Passed
21	UAT – s ZZ	Not Run	Passed	Passed
22	UAT – s ZZ	Not Run	Failed	Passed
23	UAT – s ZZ	Not Run	Passed	Passed
24	UAT – s ZZ	Not Run	Failed	Passed
25	UAT – s ZZ	Not Run	Failed	Passed
26	UAT – s ZZ	Not Run	Failed	Passed
27	UAT – s ZZ	Not Run	Failed	Passed

Závažnost je jednotně nastavena pro SI a ST na „Critical“ a to především z důvodu nemožnosti dalšího testování v případě objevení chyby. Fáze UAT pak v rámci celého testování obsahuje hodnotu „Moderate“ z důvodu podobnosti testů, které mají dopady pouze na výstupní hodnoty systému. V rámci priorit nastává pak dělení detailnější, a to především z důvodu, že se jedná o testování funkčnosti. Proto budou všechny chyby v prvních fázích (integračních a systémových testů) hodnoceny prioritou „High“ a testy v rámci akceptace pak jako „Medium“.

Výjimka nastává pouze v případě prvního release, kde je chyba u testovacích případu 15 až 20 priority „High“. Navýšení je způsobené nesprávným zpracováním logiky pro zákonného zástupce, která způsobuje nekonzistentnost celé sady testů, vyúsťující v ukončení testování této sady v polovině exekuce.

Další chyby jsou směřovány především do logiky žádostí o státní příspěvky, kdy jsou všechny požadavky o příspěvek hodnoceny negativně, a to právě i v případech, u kterých je žádost relevantní.

Poslední simulovaná chyba, odhalena především v release 2, nastává ve zvolené délce produktu, který si uživatel vybral. Problém se projevuje ve výstupních hodnotách v podobě špatně vypočteného úroku, který této době náleží.

4.5 Reportovaní a měření kvality

4.5.1 Reportování

Reportování neboli informování určených osob o současném stavu systému je nedílnou součástí každého projektu. Existuje spousta způsobů, jak výsledný report může být připravený, ale v rámci projektu byl zvolený nástroj MS Excel, jakožto nerozšířený nástroj dostupný téměř kdekoliv. Reporty berou jako podklad výsledky testování, tedy simulaci z kapitoly 4.4, a jsou vytvořeny pro každý release zvlášť.

Z důvodu, že se jedná pouze o zpracování výsledků simulace, u které neprobíhalo detailní zakládání incidentů je cílem návrhu vytvořit souhrnný report za jednotlivé fáze. Výsledné řešení bude zpracováno pomocí kontingenčních tabulek, které budou sloužit jako vstup pro jednotlivé grafy, které informují o úspěšnosti jednotlivých fází. Tabulky nebudou vytvořeny pro třetí release, a to z důvodu 100 % úspěšného otestování, a tedy nedostatečné vypovídající hodnoty případného grafu.

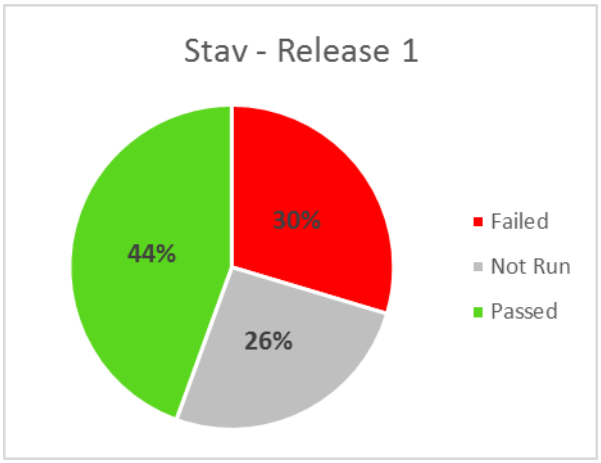
4.5.1.1 Release 1

Jako výsledný report pro release 1 byly vytvořeny 2 souhrnné grafy, které by měli sloužit jako průběžný přehled o stavu testů a celkové konzistentnosti testovaného systému.

Prvním zvoleným grafem je kompletní přehled výsledků testovaných případů, který vyjadřuje, kolik testů bylo exekvováno a s jakým výsledkem skončily. Report se skládá z Tabulka 19 a Obrázek 13 a je znázorněný ve formě koláčového grafu.

Tabulka 19 Souhrnný report za release 1

Popisky řádků	Počet z TC ID
Failed	8
Not Run	7
Passed	12
Celkový součet	**27**

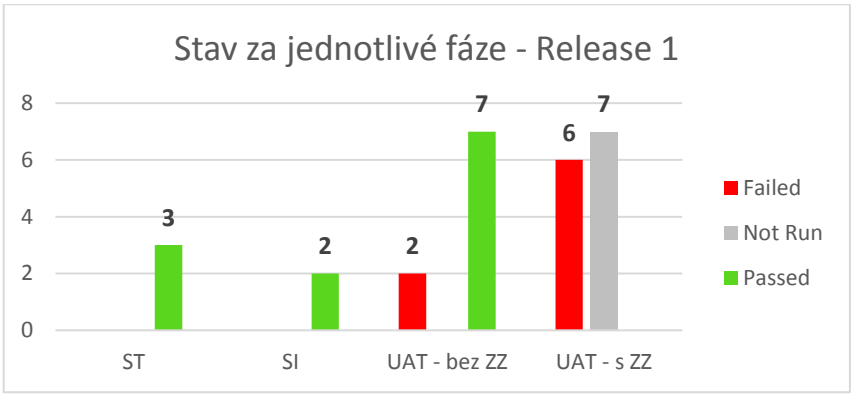


Obrázek 13 Graficky znázorněný souhrnný report za release 1

Druhou částí reportu přehled úspěšnosti za jednotlivé fáze testování a s jakým stavem skončili testovací případy. Tento přehled je znázorněn formou sloupcového grafu viz Obrázek 14, který vychází z Tabulka 20.

Tabulka 20 Souhrnný report za jednotlivé fáze v rámci release 1

Počet z TC ID	Popisky sloupců		
Popisky řádků	Failed	Not Run	Passed
ST			3
SI			2
UAT – bez ZZ	2		7
UAT – s ZZ	6	7	
Celkový součet	**8**	**7**	**12**



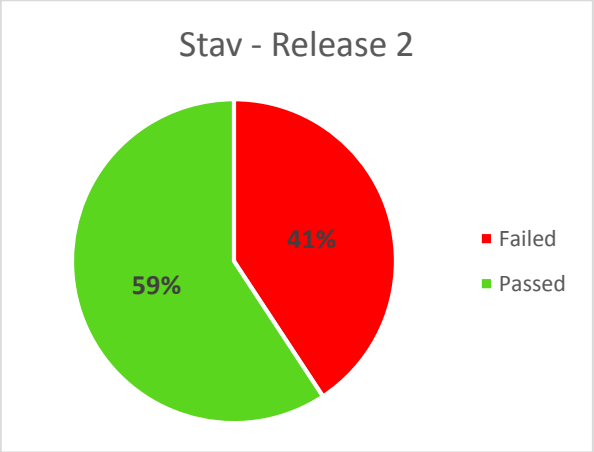
Obrázek 14 Graficky znázorněný souhrnný report za jednotlivé fáze – release 1

4.5.1.2 Release 2

V rámci druhého release již ubyl stav „Not run“, jak je patrné na Obrázek 15, a tudíž byly již všechny testovací případy alespoň jednou spuštěny a systém tedy poprvé kompletně otestován. Zdrojové data, pro souhrnný graf, jsou pak zachyceny v Tabulka 20.

Tabulka 21 Souhrnný report za release 2

Popisky řádků	Počet z TC ID
Failed	11
Passed	16
Celkový součet	**27**

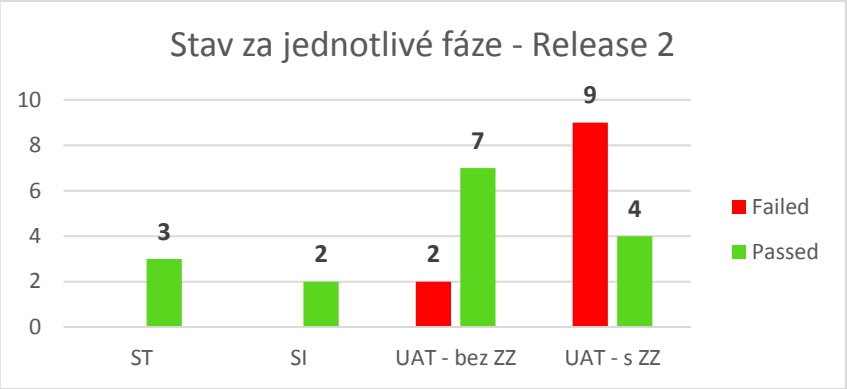


Obrázek 15 Graficky znázorněný souhrnný report za release 2

Stejně jako v release 1 je potřeba mít přehled o výsledcích všech fází, aby bylo případně možné rychle odvodit, kde mohl vzniknout problém. Proto je i v release 2 vytvořen souhrn za jednotlivé oblasti, který je zachycený na Obrázek 16, jenž vychází z Tabulka 22.

Tabulka 22 Souhrnný report za jednotlivé fáze v rámci release 2

Počet z TC ID	Popisky sloupců	
Popisky řádků	Failed	Passed
ST		3
SI		2
UAT – bez ZZ	2	7
UAT – s ZZ	9	4
Celkový součet	**11**	**16**



Obrázek 16 Graficky znázorněný souhrnný report za jednotlivé fáze – release 2

4.5.2 Měření kvality

Měření kvality, které hlídá kvalitu software v rámci jeho celého životního cyklu je silně spjaté se samotným reportováním, avšak přesto rozdílné oproti vytváření souhrnných reportů. Základ pro jednotlivé metriky, které budou sledovány, je již v samotném testovacím plánu celého projektu. V kapitole 4.2.11 je soubor všech hlídaných ukazatelů pro měření celkové kvality.

Mimo hlavní metriky bude navíc sestavena souhrnná tabulka, kde bude zobrazen celkový průběh testování. V Tabulka 23 je znázorněn kompletní vývoj testů v rámci celého životního cyklu. Zeleně jsou vyznačeny oblasti, které nepotřebují žádná zásahy a běží z velké části správně. Žlutá barva pak označuje části, kdy je potřeba daným problémům věnovat zvýšenou pozornost, avšak stále se nejedná o kritické části. A nakonec červeně vyznačené pole zachycují kritické oblasti, ve kterých je potřeba provést okamžitou opravu a věnovat zde velkou část času.

V release 1 lze vidět tuto oblast především u testů se zákonným zástupcem, a to z důvodu nulového procenta úspěšných testů, a naopak velkého počtu chyb. U druhého release je pak viditelná změna u UAT testů, kde již není nulová úspěšnost u stavebních spoření se zákonným zástupcem, ale na druhou stranu i po první vydané opravě se nezmenšil počet chyb u SS bez zákonného zástupce.

Tabulka 23 Kompletní hodnocení testování napříč všemi releasy

Fáze	Podoblast	Release 1			Release 2			Release 3		
		Passed	Failed	Not run	Passed	Failed	Not run	Passed	Failed	Not run
SI	–	100%	0%	0%	100%	0%	0%	100%	0%	0%
ST	–	100%	0%	0%	100%	0%	0%	100%	0%	0%
UAT	bez ZZ	78%	22%	0%	78%	22%	0%	100%	0%	0%
UAT	s ZZ	0%	46%	54%	31%	69%	0%	100%	0%	0%

4.5.2.1 Metriky

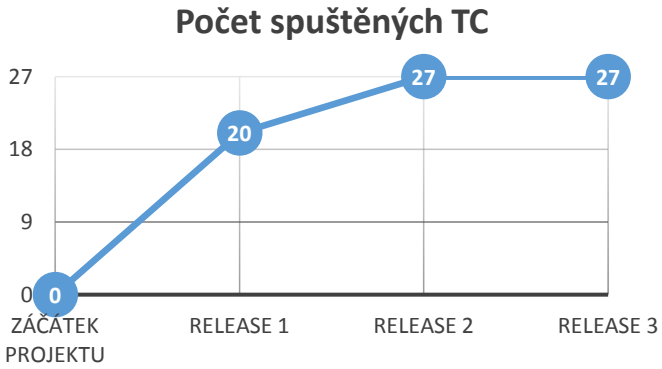
Pro každou metriku bude zpracováno samotné zhodnocení v rámci každé iterace podle stanovených metrik. Pro zjednodušení celého modelu bude brán jeden testovací případ jako jedna potencionální chyba (některé TC můžou selhat na stejné chybě). Severita a priorita jednotlivých případů bude použita ze samotné simulace v kapitole 4.4.

Protestováno

Metrika hlídající celkovou otestovatelnost systému je důležitá především kvůli budoucímu odhadu pracnosti na spuštění. V případě, že již v prvních iteracích systému se nestíhá protestovat velká část testů, se dá předpokládat, že změna nenastane ani v dalších opakováních testování. Proto je potřeba přizpůsobit celkovou pracnost na testy realističtěji. Podkladem pro toto rozplánování může být Tabulka 24 a Obrázek 17.

Tabulka 24 Zachycení protestovatelnosti systému napříč všemi release

Release 1	Release 2	Release 3
74 %	100 %	100 %

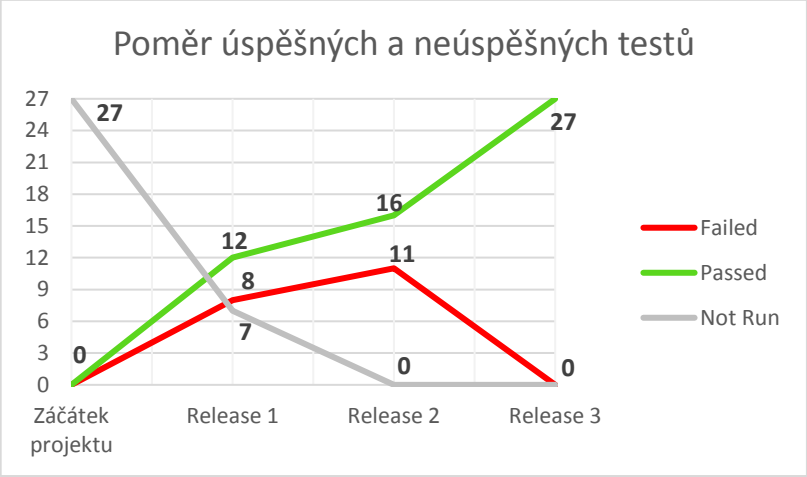


Obrázek 17 Grafické znázornění protestovatelnosti systému

Samozřejmě se dá očekávat, že první běhy testování budou na čas náročnější než spuštění následující, a to především z důvodu velké pravděpodobnosti objevení chyb a tím spojené jejich zakládání do systému.

Celkový počet nalezených incidentů v poměru s testy úspěšnými

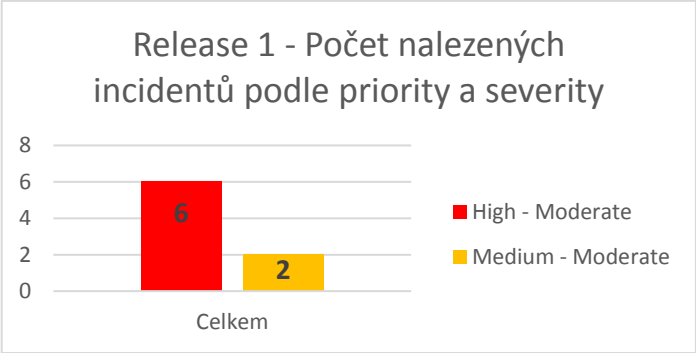
V rámci statistiky se hlídá kvalita na základě poměru úspěšných a neúspěšných testovacích scénářů, a to vždy ze současného otestovaného seznamu testů viz červená a zelená spojnice na Obrázek 18. Z grafu pak lze vyčíst, v jakém stavu se systém nachází a zda celkový počet chyb ubývá (rostoucí tendence chybných oproti úspěšným testům, by znamenala závažnější problém a bylo by potřeba podniknout opatření).



Obrázek 18 Grafické znázornění poměru jednotlivých stavů testů (Not run, passed, failed)

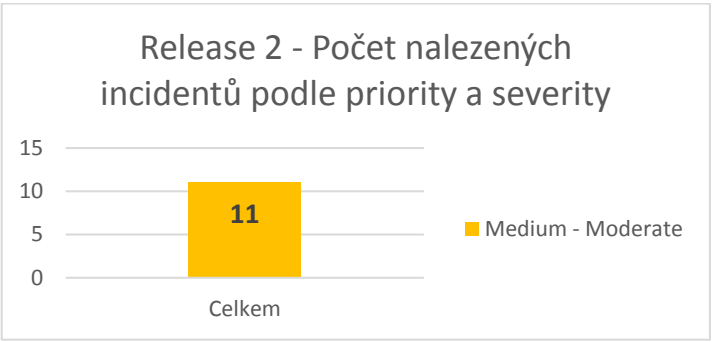
Počet nalezených incidentů podle závažností a priorit

Poslední sledovanou metrikou je celkový počet nalezených chyb v rámci procesu testování a jejich následné rozdělení do kategorií podle kombinace priority a severity. Pro release 1 je přehled znázorněný na Obrázek 19, kde je parný velký počet chyb s vysokou prioritou, tedy chyby významně ovlivňující funkčnost systému (zpracování stavebních spoření bez zákonného zástupce).



Obrázek 19 Počet nalezených chyb podle priority a severity – release 1

Naopak na Obrázek 20 lze pozorovat mírný nárůst chybovosti systému, ale zároveň odstranění všech chyb s prioritou „High“, tedy pouze chyb, které neovlivňují velkou část systému. Jedná se už „pouze“ o jednotné nefunkčnosti v logice zpracování. Z toho důvodu nemohou být taky zařazeny níže než do kategorie „Medium“.



Obrázek 20 Počet nalezených chyb podle priority a severity – release 2

4.6 Analýza použitelnosti

Z důvodu, že projekt není ve fázi realizace, nemá tedy funkční grafické prostředí, na které by se dali aplikovat jednotlivé metody pro hodnocení použitelnosti, či vytvářet jednotlivé návrhy na testování.

V budoucích fázích by mělo být navrženo jednoduché grafické prostředí celkové webové stránky tak, aby splňovalo všechny standardy a normy dnešních webových stránek. Celkové ovládání by mělo být intuitivní a přehledné, tedy zapracováno do jednoduchého moderního designu. Výsledné řešení by mělo být testováno jak s účastí koncových uživatelů, tak bez účasti koncových uživatelů. Zahrnutými technikami v rámci obou skupin by mělo být alespoň testování pomocí testování prototypů, skupinových průchodů, kritických

případů, performance testy, kontrola standardů, kontrola konzistence webu, heuristické testování a kognitivní průchody.

Samotné založení stavebního spoření by mělo být rozděleno do dvou částí, které na sebe budou navzájem navazovat. V první části by mělo proběhnout vyplnění základních informací o klientovi jako je datum narození, rodné číslo, jméno a příjmení (viz Obrázek 21). Zadané informace je potřeba zpracovat a zobrazit všechna povinná pole pro založení spoření online, včetně možné prémie a úroku podle vybraného spořícího produktu. Je potřeba taky brát v potaz rozdílnost zobrazených polí pro zletilé a nezletilé klienty (musí se zobrazit pole pro zákonného zástupce a pole podpis musí taktéž být určeno pro něj).

Obrázek 21 Wireframe pro část úvodní stránky týkající se založení stavebního spoření (MS Visio)

Druhá obrazovka je již specifická, a bude určena až na základě přesnějších business požadavků, která pole se mají na obrazovce objevovat a zpracovávat. Kvůli chybějícím požadavkům nelze připravit náhled samotného zpracování, které by prezentovalo možnou grafiku a rozmístění.

5 Zhodnocení

V úvodu praktické části diplomové práce byly vyspecifikovány požadavky na testovaný systém na základě business analýzy, přesněji pro část systému zabývající se založením stavebních spoření elektronickou formou. Bylo stanoveno, že modul má několik vstupů v podobě věku klienta, řeší variantu zákonného zástupce pro nezletilé osoby, volbu cílové částky na spoření, výběr produktu spoření s různou délkou trvání. Také zohledňuje, zda zákazník žádá o státní příspěvek či nikoliv a v posledním parametru, zda se jedná o občana české republiky. Podle zadaných vstupů se pak vytvářely relevantní výstupy, a to podle zadané kombinace dat. Mezi výstupy se pak objevují položky jako úrok, který bude pro stavební spoření poskytnut, zda měl klient nárok na získání finanční prémie za založení, jestli bylo vyhověno jeho žádosti o státní příspěvek a zda bylo spoření založeno. Poslední výstup byl vždy kladný, a to především z důvodu webového formuláře, který neumožňuje zadání nekompletních údajů. Mimo jiné byly také nadefinovány dopady ze založení stavebního spoření do ostatních systémů instituce. Jednalo se o komplexní systému CRM, do kterého se klient propisoval a systému na tisk dopisů s generováním tiskových a archivačních podkladů.

Další aktivitou bylo sestavení relevantního testovacího plánu na tento projekt, který se stal postupem pro realizaci celého projektu. Byly nadefinovány všechny potřebné náležitosti, ať už se jednalo o samotné vymezení předmětu testování (zároveň i vymezení co se testovat nebude), sestavení týmu lidí, kteří se budou na projektu podílet, vydefinování názvosloví jednotného pro celý realizační tým, tak i jejich jednotlivé povinnosti rozdělené podle rolí a závazný harmonogram postupu. Zdůraznění si zaslouží tři kategorie z testovacího plánu. První je strategie testování použitá pro projekt. Zde je vytvořen přehled všech metod, které budou na projekt aplikovány a jak bude celé testování postupovat. Druhým aspektem jsou metriky, které mají za úkol hlídat kvalitu samotného softwaru v průběhu celého testování. Posledním jsou rizika, která mohou při projektu vyvstat, a to včetně jejich dopadů, pravděpodobnosti vzniku i závažnosti jejich objevení. Na jejich základě byla navržena opatření na snížení šance na jejich objevení.

Na takto vydefinovaných podkladech již mohla začít samotná testovací analýza, která měla za úkol vydefinovat všechny testovací případy pro úspěšné otestování zvoleného systému. Rozsah testů musel být zachovaný v rozumné míře, aby samotné odstranění chyby v produkci již nebylo levnější než probíhající testování. Analýza byla provedena pro všechny

tři testovací fáze s využitím black box přístupu. Výsledkem bylo vytvoření dvaceti sedmi testovacích případů, napříč všemi fázemi testování, a provedení automatizace relevantních testů tak, aby pracnost byla co nejnižší v průběhu celého testování. Proto byla automatizace provedena především v prvních fázích, kde byly testovací případy zařazeny mezi smoke testy, za využití jazyka PHP a programu SoapUI v kombinaci s příkazovou řádkou systému Windows. Následně proběhla simulace předpokládaného průběhu testování, a to včetně vytvoření sady chyb tak, aby mohly být připraveny podklady pro hodnocení kvality a reportování.

Samotné výstupy ze simulace byly zpracovány formou kontingenčních tabulek a na jejich základě pak vytvořeno grafické znázornění výsledků testů pro jednotlivých release. Pro lepší přehled byly vytvořeny dva pohledy. První se zaměřil na release jako celek a druhý pak ukazoval úspěšnost po jednotlivých fázích, případně skupinách. Mimo reporty probíhalo také hlídání kvality, a to na základě metrik definovaných v testovacím plánu. Z jednotlivých metod byla vybrána forma metrik dle použitých dat a metrik založených na testech. Všechna data byla použita z výsledků simulace a z nich vytvořena co nejjednodušší grafická znázornění dané metriky.

Vzhledem k absenci grafického prostředí nemohla být provedena detailní analýza metod pro zpracování použitelnosti webové stránky na zadaný projekt. Z toho důvodu zůstala kapitola v rámci prvního zpracování analýzy pouze u návrhu a doporučení pro rozšíření v budoucnosti. Byly navrhnuty základní metriky hodnocení použitelnosti, které by měly být na webové rozhraní použity, a to metriky počítající s účastí koncových uživatelů tak i bez jejich účasti. Na základě již známých business požadavků byl proveden i jednoduchý návrh wireframu, pro část obrazovky, týkající se založení stavebního spoření.

6 Závěr

Hlavní cíl práce vycházel z projektu definovaného v úvodu praktické části. Testovaná část systému byla zaměřena na založení stavebního spoření elektronickou formou. Na základě požadavků byl sestaven postup testování v podobě testovacího plánu, a to včetně všech náležitostí. Druhým krokem bylo zpracování testovací analýzy s využitím nejrůznějších technik. Výsledkem jsou vytvořené testovací případy, které v dostatečné míře pokrývají celý testovaný systém. Součástí výstupu analýzy je také návrh na automatizaci testů za využití PHP a programu SoapUI v kombinaci s příkazovou řádkou Windows. Dále byly realizovány i dílčí cíle práce s následujícími výsledky:

Vytvoření reportu a vyhodnocení testování

Z důvodu, že projekt nedospěl do fáze realizace testů, byla provedena simulace testování, která vycházela z dvaceti sedmi testovacích případů definovaných v průběhu analýzy projektu. Testování bylo provedeno celkem pro tři release, které probíhaly napříč všemi testovacími fázemi. Pro věrohodnější výsledky byly vytvořeny různé chyby, které by v rámci skutečného testování mohly nastat, a to včetně definování priority a severity. V druhé části byla data zpracována formou kontingenčních tabulek za využití aplikace MS Excel. Výsledek byl znázorněn v podobě dvou grafů, pro každý zpracovaný release.

Analýza metod pro hodnocení kvality softwaru

Byl vytvořen přehled technik pro měření kvality a na jejich základě byly vybrány nejvhodnější metody, a to dle použitých dat a metod založených na testech. Obě vycházely z metrik specifikovaných v rámci testovacího plánu a podkladem jim byla simulace testování. Výstup metrik byl zpracován v co nejpřehlednější podobě, přesněji pomocí jednoduchého grafického znázornění a následného popisu.

Porovnání metod pro hodnocení použitelnosti

Vzhledem k absenci grafického prostředí, byla kapitola ponechána v rámci návrhu a doporučení pro rozšíření do budoucna. Byly navrženy základní metriky hodnocení použitelnosti, které by měly být na webovém rozhraní použity a jednoduchý návrh wireframe, pro část obrazovky, týkající se založení stavebního spoření.

Případová studie

Pro účely případové studie byla zvolena část systému, přesněji založení stavebního spoření přes webovou platformu. Na příkladu byly následně aplikovány všechny metody zpracovávané v rámci praktické části.

Formulování obecných a specifických závěrů

Testování je v dnešní době nezbytnou součástí každého vyvíjeného systému. Z tohoto důvodu byl připraven ukázkový model testování na reálném projektu. Všechny business požadavky byly transformovány na co nejsrozumitelnější zadání, z kterého byl sestaven testovací plán a testovací analýzy, včetně výběru nejvhodnějších metod. V návaznosti na to proběhlo vydefinování a popis automatizace testovacích případů a simulace jejich spuštění pro nastavení reportingu a hodnocení kvality.

7 Seznam použitých zdrojů

Acceptance testing. 2017. What is Acceptance testing. *ISTQB EXAM CERTIFICATION*. [Online] 2017. [Citace: 21. 1 2017.] http://istqbexamcertification.com/what-is-acceptance-testing/.

Automatizované testování. 2016. Automatizované testování. *Testování softwaru.* [Online] 2016. [Citace: 24. 1 2017.] http://testovanisoftwaru.cz/automatizovane-testovani/.

Bach, James. 2016. Satisfice Inc. [Online] 2016. [Citace: 26. 2 2017.] http://www.satisfice.com/tools.shtml.

Black, Rex. 2009. *Managing the Testing Process.* 3. Indianapolis : Wiley Publishing, 2009. 978-0-470-40415-7.

Bureš, Miroslav, a další. 2016. *Efektivní testování softwaru.* 1. Praha : Grada, 2016. 978-80-247-5594-6.

CaSTB. 2017. Czech and Slovak Testing Board. [Online] 2017. [Citace: 12. 2 2017.] http://castb.org/cz/certifikace/.

cncaps. 2016. cncaps. [Online] 2016. [Citace: 18. 2 2017.] https://www.cncaps.com/wp-content/uploads/2016/07/CNCAPS-Management.png?_t=1479174776.

Čermák, Miroslav. 2010a. Black box test. *Clever and smart.* [Online] 2010a. [Citace: 10. 1 2017.] http://www.cleverandsmart.cz/black-box-test/.

—. 2010b. Grey box test. *Clever and smart.* [Online] 2010b. [Citace: 10. 1 2017.] http://www.cleverandsmart.cz/grey-box-test/.

—. 2010c. White box test. *Clever and smart.* [Online] 2010c. [Citace: 10. 1 2017.] http://www.cleverandsmart.cz/white-box-test/.

Farrell-Vinay, Peter. 2008. *Manage Software Testing.* 1. Boca Raton : Auerbach Publications, 2008. 978-0849393839.

Integration testing. 2017. What is Integration testing? *ISTQB EXAM CERTIFICATION*. [Online] 2017. [Citace: 17. 1 2017.] http://istqbexamcertification.com/what-is-integration-testing/.

ISTQB. 2012. Advanced Level Syllabus - Technical Test Analyst. [Online] 19. 10 2012. [Citace: 20. 1 2017.] http://castb.org/wp-content/uploads/2013/09/advanced_syllabus_2012_technical_test_analyst_ga_release_20121019.pdf.

—. 2013. Certifikovaný tester: Učební osnovy pro základní stupeň. *Czech and Slovak Testing Board.* [Online] 15. 6 2013. [Citace: 20. 12 2016.] http://castb.org/wp-content/uploads/2013/11/ISTQB-CTFL-Syllabus-v2011-CZ-Beta1.pdf.

Jorgensen, Paul C. 2013. *Software Testing: A Craftsman's Approach, Fourth Edition.* 4th. Boca Raton : Auerbach Publications, 2013. 9781466560680.

Krug, Steve. 2010. *Nenuťte uživatele přemýšlet!* 1. Praha : COMPUTER PRESS, 2010. 978-80-251-2923-4.

Kutina, Rudolf. 2010. Quality Guru's – Home of QA/QE SW Testing Workshops. [Online] 2010. [Citace: 10. 1 2017.] https://qualityguru.files.wordpress.com/2010/08/spiral_model.gif?w=490.

Lacko, Branislav. 2006. [Online] 2006. [Citace: 10. 1 2017.] http://www.vns.wz.cz/NSR4_soubory/image010.gif.

Mili, Ali a Tchier, Fairouz. 2015. *Software Testing: Concepts and Operations.* [editor] 1st. New Jersey : Wiley, 2015. 1118662873.

Naik, Kshirasagar a Tripathy, Priyadarshi. 2008. *Software Testing and Quality Assurance: Theory and Practice.* 1. New Jersey : Wiley-Spektrum, 2008. 978-0471789116.

Non-functional testing. 2017. What is Non-functional testing? *ISTQB EXAM CERTIFICATION.* [Online] 2017. [Citace: 23. 1 2017.] http://istqbexamcertification.com/?s=non-functional.

Patton, Ron. 2002. *Testování softwaru.* 1. Praha : COMPUTER PRESS, 2002. 80-7226-636-5.

Progresní a regresní testy. 2016. Progresní a regresní testy. *Testování softwaru.* [Online] 2016. [Citace: 25. 1 2017.] http://testovanisoftwaru.cz/metodika-testovani/druhy-typy-a-kategorie-testu/progresni-a-regresni-testy/.

Roudenský, Petr a Havlíčková, Anna. 2013. *Řízení kvality softwaru.* Brno : Competer Press, 2013. stránky 102-116. 978-80-251-3816-8.

Severity and Priority. 2015. What is the difference between Severity and Priority? *ISTQB EXAM CERTIFICATION.* [Online] 2015. [Citace: 11. 2 2017.] http://istqbexamcertification.com/what-is-the-difference-between-severity-and-priority/.

SmartBear. 2017. What Is Unit Testing? [Online] 2017. [Citace: 15. 1 2017.] https://smartbear.com/learn/automated-testing/what-is-unit-testing/.

Spillner, Andreas, Linz, Tilo a Schaefer, Hans. 2014. *SOFTWARE TESTING FOUNDATIONS.* 4. Santa Barbara : Rocky Nook, 2014. 978-1-937538-42-2.

Spiral model. 2017. What is Spiral model? *ISTQB EXAM CERTIFICATION.* [Online] 2017. [Citace: 11. 1 2017.] http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/.

Testingfreak. 2017. Testingfreak. [Online] 2017. [Citace: 10. 1 2017.] http://testingfreak.com/wp-content/uploads/2015/02/waterfall.png.

Trnka, Andrej. 2008. Výber modelu testovania aplikácií dátových skladov. [Online] 3. 3 2008. [Citace: 5. 2 2017.] http://jit.efmk.sk/sites/default/files/01-trnka.pdf.

Tutorialspoint. 2017. SDLC - Waterfall Model. [Online] 2017. [Citace: 10. 1 2017.] https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.

Umel, VUT Brno. 2012. Vývojové modely. [Online] 2012. [Citace: 10. 1 2017.] http://www.umel.feec.vutbr.cz/bdts/index.php/embedded-systemy/vyvojove-modely.

Waterfall model. 2017. What is Waterfall model. *ISTQB EXAM CERTIFICATION.* [Online] 2017. [Citace: 10. 1 2017.] http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/.

8 Seznam obrázků

9 Seznam tabulek

10 Přílohy

Příloha 1

Výstup allpairs pro TC bez zákonného zástupce

TEST CASES

case	Věk	Zákonný zástupce		Státní příspěvek	Částka	Cizinec	Délka spoření	pairings
1	14 a méně	ANO	ANO	30k	ANO	6 let	15	
2	15	ANO	NE	35k	NE	10 let	15	
3	15-17	ANO	ANO	150k	NE	6 let	11	
4	14 a méně	ANO	NE	2m	ANO	10 let	9	
5	15	~ANO	ANO	30k	ANO	10 let	5	
6	15-17	~ANO	NE	35k	ANO	6 let	6	
7	14 a méně	~ANO	NE	150k	NE	10 let	4	
8	15	~ANO	ANO	2m	NE	6 let	5	
9	15-17	~ANO	NE	30k	NE	10 let	4	
10	14 a méně	~ANO	ANO	35k	~NE	~6 let	2	
11	15	~ANO	~NE	150k	ANO	~6 let	2	
12	15-17	~ANO	~ANO	2m	~ANO	~10 let	1	

PAIRING DETAILS

var1	var2	value1	value2	appearances	cases
Částka	Věk	30k	14 a méně	1	1
Částka	Věk	30k	15	1	5
Částka	Věk	30k	15-17	1	9
Částka	Věk	35k	14 a méně	1	10
Částka	Věk	35k	15	1	2
Částka	Věk	35k	15-17	1	6
Částka	Věk	150k	14 a méně	1	7
Částka	Věk	150k	15	1	11
Částka	Věk	150k	15-17	1	3
Částka	Věk	2m	14 a méně	1	4
Částka	Věk	2m	15	1	8
Částka	Věk	2m	15-17	1	12
Částka	Státní příspěvek	30k	ANO	2	1, 5
Částka	Státní příspěvek	30k	NE	1	9
Částka	Státní příspěvek	35k	ANO	1	10
Částka	Státní příspěvek	35k	NE	2	2, 6
Částka	Státní příspěvek	150k	ANO	1	3
Částka	Státní příspěvek	150k	NE	2	7, 11
Částka	Státní příspěvek	2m	ANO	2	8, 12
Částka	Státní příspěvek	2m	NE	1	4
Částka	Cizinec	30k	ANO	2	1, 5
Částka	Cizinec	30k	NE	1	9
Částka	Cizinec	35k	ANO	1	6
Částka	Cizinec	35k	NE	2	2, 10
Částka	Cizinec	150k	ANO	1	11
Částka	Cizinec	150k	NE	2	3, 7
Částka	Cizinec	2m	ANO	2	4, 12
Částka	Cizinec	2m	NE	1	8
Částka	Délka spoření	30k	6 let	1	1
Částka	Délka spoření	30k	10 let	2	5, 9

Částka	Délka spoření	35k	6 let	2	6, 10	
Částka	Délka spoření	35k	10 let	1	2	
Částka	Délka spoření	150k	6 let	2	3, 11	
Částka	Délka spoření	150k	10 let	1	7	
Částka	Délka spoření	2m	6 let	1	8	
Částka	Délka spoření	2m	10 let	2	4, 12	
Částka	Zákonný zástupce		30k	ANO	3	1, 5, 9
Částka	Zákonný zástupce		35k	ANO	3	2, 6, 10
Částka	Zákonný zástupce		150k	ANO	3	3, 7, 11
Částka	Zákonný zástupce		2m	ANO	3	4, 8, 12
Věk	Státní příspěvek	14 a méně		ANO	2	1, 10
Věk	Státní příspěvek	14 a méně		NE	2	4, 7
Věk	Státní příspěvek	15	ANO	2	5, 8	
Věk	Státní příspěvek	15	NE	2	2, 11	
Věk	Státní příspěvek	15-17	ANO	2	3, 12	
Věk	Státní příspěvek	15-17	NE	2	6, 9	
Věk	Cizinec	14 a méně		ANO	2	1, 4
Věk	Cizinec	14 a méně		NE	2	7, 10
Věk	Cizinec	15	ANO	2	5, 11	
Věk	Cizinec	15	NE	2	2, 8	
Věk	Cizinec	15-17	ANO	2	6, 12	
Věk	Cizinec	15-17	NE	2	3, 9	
Věk	Délka spoření	14 a méně		6 let	2	1, 10
Věk	Délka spoření	14 a méně		10 let	2	4, 7
Věk	Délka spoření	15	6 let	2	8, 11	
Věk	Délka spoření	15	10 let	2	2, 5	
Věk	Délka spoření	15-17	6 let	2	3, 6	
Věk	Délka spoření	15-17	10 let	2	9, 12	
Věk	Zákonný zástupce		14 a méně		ANO	4 1, 4, 7, 10
Věk	Zákonný zástupce		15	ANO	4	2, 5, 8, 11
Věk	Zákonný zástupce		15-17	ANO	4	3, 6, 9, 12
Státní příspěvek	Cizinec	ANO	ANO	3	1, 5, 12	
Státní příspěvek	Cizinec	ANO	NE	3	3, 8, 10	
Státní příspěvek	Cizinec	NE	ANO	3	4, 6, 11	
Státní příspěvek	Cizinec	NE	NE	3	2, 7, 9	
Státní příspěvek	Délka spoření		ANO	6 let	4	1, 3, 8, 10
Státní příspěvek	Délka spoření		ANO	10 let	2	5, 12
Státní příspěvek	Délka spoření		NE	6 let	2	6, 11
Státní příspěvek	Délka spoření		NE	10 let	4	2, 4, 7, 9
Státní příspěvek	Zákonný zástupce			ANO	ANO	6 1, 3, 5, 8, 10, 12
Státní příspěvek	Zákonný zástupce			NE	ANO	6 2, 4, 6, 7, 9, 11
Cizinec	Délka spoření	ANO	6 let	3	1, 6, 11	
Cizinec	Délka spoření	ANO	10 let	3	4, 5, 12	
Cizinec	Délka spoření	NE	6 let	3	3, 8, 10	
Cizinec	Délka spoření	NE	10 let	3	2, 7, 9	
Cizinec	Zákonný zástupce		ANO	ANO	6	1, 4, 5, 6, 11, 12
Cizinec	Zákonný zástupce		NE	ANO	6	2, 3, 7, 8, 9, 10
Délka spoření	Zákonný zástupce			6 let	ANO	6 1, 3, 6, 8, 10, 11
Délka spoření	Zákonný zástupce			10 let	ANO	6 2, 4, 5, 7, 9, 12

Příloha 2

Výstup allpairs pro TC se zákonným zástupcem

TEST CASES

case	Věk	Zákonný zástupce			Státní příspěvek	Částka	Cizinec	Délka spoření	pairings
1	18	NE	ANO	30k	ANO	6 let	15		
2	18-20	NE	NE	35k	NE	10 let	15		
3	21	NE	ANO	150k	NE	6 let	11		
4	21-50	NE	NE	2m	ANO	10 let	11		
5	50 a víc	NE	NE	30k	NE	6 let	8		
6	18	~NE	ANO	35k	ANO	10 let	5		
7	18-20	~NE	ANO	150k	ANO	6 let	5		
8	21	~NE	NE	2m	NE	10 let	4		
9	21-50	~NE	NE	150k	NE	6 let	4		
10	50 a víc	~NE	ANO	2m	ANO	10 let	5		
11	18	~NE	NE	150k	NE	10 let	4		
12	21	~NE	~ANO	30k	ANO	10 let	3		
13	21-50	~NE	ANO	35k	~NE	6 let	3		
14	18-20	~NE	~NE	2m	~ANO	6 let	2		
15	50 a víc	~NE	~NE	35k	~ANO	~6 let	1		
16	18	~NE	~ANO	2m	~NE	~6 let	1		
17	18-20	~NE	~ANO	30k	~NE	~10 let	1		
18	21	~NE	~NE	35k	~ANO	~6 let	1		
19	21-50	~NE	~NE	30k	~ANO	~10 let	1		
20	50 a víc	~NE	~ANO	150k	~NE	~10 let	1		

PAIRING DETAILS

var1	var2	value1	value2	appearances	cases
Věk	Částka	18	30k	1	1
Věk	Částka	18	35k	1	6
Věk	Částka	18	150k	1	11
Věk	Částka	18	2m	1	16
Věk	Částka	18-20	30k	1	17
Věk	Částka	18-20	35k	1	2
Věk	Částka	18-20	150k	1	7
Věk	Částka	18-20	2m	1	14
Věk	Částka	21	30k	1	12
Věk	Částka	21	35k	1	18
Věk	Částka	21	150k	1	3
Věk	Částka	21	2m	1	8
Věk	Částka	21-50	30k	1	19
Věk	Částka	21-50	35k	1	13
Věk	Částka	21-50	150k	1	9
Věk	Částka	21-50	2m	1	4
Věk	Částka	50 a víc	30k	1	5
Věk	Částka	50 a víc	35k	1	15
Věk	Částka	50 a víc	150k	1	20
Věk	Částka	50 a víc	2m	1	10
Věk	Státní příspěvek	18	ANO	3	1, 6, 16
Věk	Státní příspěvek	18	NE	1	11
Věk	Státní příspěvek	18-20	ANO	2	7, 17
Věk	Státní příspěvek	18-20	NE	2	2, 14
Věk	Státní příspěvek	21	ANO	2	3, 12
Věk	Státní příspěvek	21	NE	2	8, 18
Věk	Státní příspěvek	21-50	ANO	1	13
Věk	Státní příspěvek	21-50	NE	3	4, 9, 19
Věk	Státní příspěvek	50 a víc	ANO	2	10, 20
Věk	Státní příspěvek	50 a víc	NE	2	5, 15
Věk	Cizinec	18	ANO	2	1, 6

Věk	Cizinec	18	NE	2	11, 16		
Věk	Cizinec	18-20	ANO	2	7, 14		
Věk	Cizinec	18-20	NE	2	2, 17		
Věk	Cizinec	21	ANO	2	12, 18		
Věk	Cizinec	21	NE	2	3, 8		
Věk	Cizinec	21-50	ANO	2	4, 19		
Věk	Cizinec	21-50	NE	2	9, 13		
Věk	Cizinec	50 a víc	ANO	2	10, 15		
Věk	Cizinec	50 a víc	NE	2	5, 20		
Věk	Délka spoření	18	6 let	2	1, 16		
Věk	Délka spoření	18	10 let	2	6, 11		
Věk	Délka spoření	18-20	6 let	2	7, 14		
Věk	Délka spoření	18-20	10 let	2	2, 17		
Věk	Délka spoření	21	6 let	2	3, 18		
Věk	Délka spoření	21	10 let	2	8, 12		
Věk	Délka spoření	21-50	6 let	2	9, 13		
Věk	Délka spoření	21-50	10 let	2	4, 19		
Věk	Délka spoření	50 a víc	6 let	2	5, 15		
Věk	Délka spoření	50 a víc	10 let	2	10, 20		
Věk	Zákonný zástupce		18	NE	4	1, 6, 11, 16	
Věk	Zákonný zástupce		18-20	NE	4	2, 7, 14, 17	
Věk	Zákonný zástupce		21	NE	4	3, 8, 12, 18	
Věk	Zákonný zástupce		21-50	NE	4	4, 9, 13, 19	
Věk	Zákonný zástupce		50 a víc	NE	4	5, 10, 15, 20	
Částka	Státní příspěvek	30k	ANO	3	1, 12, 17		
Částka	Státní příspěvek	30k	NE	2	5, 19		
Částka	Státní příspěvek	35k	ANO	2	6, 13		
Částka	Státní příspěvek	35k	NE	3	2, 15, 18		
Částka	Státní příspěvek	150k	ANO	3	3, 7, 20		
Částka	Státní příspěvek	150k	NE	2	9, 11		
Částka	Státní příspěvek	2m	ANO	2	10, 16		
Částka	Státní příspěvek	2m	NE	3	4, 8, 14		
Částka	Cizinec	30k	ANO	3	1, 12, 19		
Částka	Cizinec	30k	NE	2	5, 17		
Částka	Cizinec	35k	ANO	3	6, 15, 18		
Částka	Cizinec	35k	NE	2	2, 13		
Částka	Cizinec	150k	ANO	1	7		
Částka	Cizinec	150k	NE	4	3, 9, 11, 20		
Částka	Cizinec	2m	ANO	3	4, 10, 14		
Částka	Cizinec	2m	NE	2	8, 16		
Částka	Délka spoření	30k	6 let	2	1, 5		
Částka	Délka spoření	30k	10 let	3	12, 17, 19		
Částka	Délka spoření	35k	6 let	3	13, 15, 18		
Částka	Délka spoření	35k	10 let	2	2, 6		
Částka	Délka spoření	150k	6 let	3	3, 7, 9		
Částka	Délka spoření	150k	10 let	2	11, 20		
Částka	Délka spoření	2m	6 let	2	14, 16		
Částka	Délka spoření	2m	10 let	3	4, 8, 10		
Částka	Zákonný zástupce		30k	NE	5	1, 5, 12, 17, 19	
Částka	Zákonný zástupce		35k	NE	5	2, 6, 13, 15, 18	
Částka	Zákonný zástupce		150k	NE	5	3, 7, 9, 11, 20	
Částka	Zákonný zástupce		2m	NE	5	4, 8, 10, 14, 16	
Státní příspěvek	Cizinec	ANO	ANO	5	1, 6, 7, 10, 12		
Státní příspěvek	Cizinec	ANO	NE	5	3, 13, 16, 17, 20		
Státní příspěvek	Cizinec	NE	ANO	5	4, 14, 15, 18, 19		
Státní příspěvek	Cizinec	NE	NE	5	2, 5, 8, 9, 11		
Státní příspěvek	Délka spoření		ANO	6 let	5	1, 3, 7, 13, 16	
Státní příspěvek	Délka spoření		ANO	10 let	5	6, 10, 12, 17, 20	
Státní příspěvek	Délka spoření		NE	6 let	5	5, 9, 14, 15, 18	
Státní příspěvek	Délka spoření		NE	10 let	5	2, 4, 8, 11, 19	

Státní příspěvek	Zákonný zástupce			ANO	NE	10	1, 3, 6, 7, 10, 12, 13, 16, 17, 20
Státní příspěvek	Zákonný zástupce			NE	NE	10	2, 4, 5, 8, 9, 11, 14, 15, 18, 19
Cizinec	Délka spoření	ANO	6 let	5	1, 7, 14, 15, 18		
Cizinec	Délka spoření	ANO	10 let	5	4, 6, 10, 12, 19		
Cizinec	Délka spoření	NE	6 let	5	3, 5, 9, 13, 16		
Cizinec	Délka spoření	NE	10 let	5	2, 8, 11, 17, 20		
Cizinec	Zákonný zástupce		ANO	NE	10	1, 4, 6, 7, 10, 12, 14, 15, 18, 19	
Cizinec	Zákonný zástupce		NE	NE	10	2, 3, 5, 8, 9, 11, 13, 16, 17, 20	
Délka spoření	Zákonný zástupce			6 let	NE	10	1, 3, 5, 7, 9, 13, 14, 15, 16, 18
Délka spoření	Zákonný zástupce			10 let	NE	10	2, 4, 6, 8, 10, 11, 12, 17, 19, 20