Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Návrh strategie testování softwaru

Bc. Lukáš Hrnčíř

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Lukáš Hrnčíř

Informatika

Název práce

Návrh strategie testování SW

Název anglicky

Design of software testing strategy

Cíle práce

Diplomová práce je zaměřena na problematiku tvorby strategie testování SW. Hlavním cílem je návrh testovací strategie pro vybranou aplikaci včetně samotné realizace testů a jejich následného vyhodnocení. Dílčím cílem je pak představit možné metody a přístupy k testování, přičemž vybraná metoda bude využita ve výše zmíněné strategii testování.

Metodika

Metodika práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných poznatků budou popsány metody a nástroje pro návrh testovací strategie.

Dále bude navrhnuta testovací strategie, která bude realizována v rámci testování vybrané aplikace. Na závěr bude provedeno vyhodnocení testování.

Doporučený rozsah práce

60-80 stran

Klíčová slova

software, test, strategie, implementace, testovací případ, chyba

---

Doporučené zdroje informací

BLACK, Rex. Managing the testing process: practical tools and techniques for managing software and hardware testing. 3rd ed. Indianapolis, MN: Wiley, c2009, xxxiv, 638 p. ISBN 9780470404157.

PATTON, Ron. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002, 313 s. Programování. ISBN 80-7226-636-5.

SOMMERVILLE, Ian. Softwarové inženýrství. 1. vyd. Brno: Computer Press, 2013, 680 s. ISBN 978-80-251-3826-7

STEPHENS, Matt a Doug ROSENBERG. Testování softwaru řízené návrhem. Vyd. 1. Brno: Computer Press, 2011, 336 s. ISBN 978-80-251-3607-2.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 28. 03. 2016

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Návrh strategie testování SW" 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 literatury 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. 2016 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Jiřímu Brožkovi, Ph. D. za velmi přínosné rady a konzultace při zpracování diplomové práce.

Návrh strategie testování SW

Design of software testing strategy

Souhrn

Práce je zaměřena na problematiku návrhu strategie testování ve formě test plánu. V teoretické části práce jsou popsány vybrané modely životního cyklus vývoje softwaru a metodiky vývoje softwaru a testování, přičemž hlavní pozornost je věnována významu testování a jeho pozici v rámci jednotlivých modelů a metodik. Praktická část práce je věnována samotné tvorbě návrhu strategie testování vybraného softwaru, včetně následného návrhu a provedení testovacích případů. Na závěr je uvedeno zhodnocení dosažených výsledků, které jsou doplněny diskuzí k možnosti volby jiného způsobu či přístupu testování.

Summary

The thesis is focused on issues of the design testing strategy in the form of the test plan. There are described selected models of the life cycle software development and methodologies of the software development and testing in the theoretical part. The main focus is dedicated to the significance testing and its position in the various models and methodologies. There is solved a creation of testing strategy of selected software in the practical part, including the subsequent design and implementation of the test cases. There is an assessment of the achieved results at the conclusion and there is discussed the possibility of using different type or approach of the testing.

Klíčová slova: Software, testování, testovací případ, strategie, test, implementace, chyba.

Keywords: Software, testing, test case, strategy, test, implementation, bug.

OBSAH

1 Úvod

Software je v dnešní době neoddělitelnou součástí života lidí, počínaje obchodními aplikacemi (např. elektronické bankovnictví), softwarem v lékařských přístrojích a konče softwarem aplikovaným do automobilů či letadel. Pokud software nepracuje správně, může způsobit mnoho problémů, jejichž následkem může dojít k finančním ztrátám, ke ztrátě obchodní reputace společnosti a v případě lékařského, automobilového či leteckého průmyslu může chyba softwaru způsobit dokonce zranění či smrt.

Donedávna byl proces testování poměrně opomíjenou činností v oblasti vývoje softwaru, nicméně s rostoucími požadavky trhu byly společnosti zaměřující se na vývoj jakéhokoli softwaru přinuceny změnit svůj postoj a začít věnovat testování větší pozornost. V současné době je možné pozorovat rostoucí počet společností, které buď již mají, nebo které budují testovací oddělení pro zajištění požadované úrovně kvality softwaru. Právě zmiňovaná úroveň kvality softwaru je v současné době klíčovým faktorem, který společnosti umožní nejen udržet si své současné zákazníky, ale také oslovit nové.

Úkolem testovacího týmu obvykle bývá provedení analýzy buď nově implementovaného systému nebo nově implementovaných funkcionalit v rámci jednoho či více systémů. Na základě zvolených cílů testování je testovacím týmem provedena analýza a návrh testovacích případů, které jsou poté prováděny, čímž dojde k otestování nové funkcionality nebo nově implementovaného systému.

Rozvoj v oblasti testování softwaru s sebou přináší zvýšenou potřebu a poptávku firem po odbornících, kteří budou působit v testovacím týmu jako test manažeři, test analytici či testeři a budou dohlížet na zajišťování požadované úrovně kvality softwarového produktu. V současnosti se IT trh potýká s nedostatkem kvalifikovaných odborníků z oblasti testování a to i přesto, že v posledních letech výrazně rostou mzdy odborníků působících v této oblasti.

Testování si v procesu vývoje softwaru buduje významnou pozici, která bude v budoucnu pravděpodobně ještě sílit, protože pokud bude dosaženo požadované úrovně kvality při vývoji softwaru, bude zajištěna jeho konkurenceschopnost na trhu.

2 Cíl a metodika práce

2.1 Cíl práce

Diplomová práce je svým tématem zaměřena na problematiku strategie testování softwaru. Hlavním cílem diplomové práce je provést návrh strategie testování pro vybraný systém včetně samotné realizace testů a jejich následného vyhodnocení.

Dílčími cíli práce jsou:

1) Představit možné metody a přístupy k testování softwaru.
2) Aplikovat vybraný přístup a metodu v rámci navržené testovací strategie.

2.2 Metodika práce

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

Praktická část práce bude věnována návrhu testovací strategie pro vybraný systém a bude realizována formou test plánu, na který bude následně navazovat test analýza. Dále bude provedena exekuce testovacích případů navržených v rámci test analýzy. Následně budou prezentovány a diskutovány dosažené výsledky. Na základě syntézy teoretických poznatků a výsledů dosažených v praktické části budou formulovány závěry diplomové práce.

Práce bude rozdělena do 3 základních kapitol:

3. Teoretická východiska
4. Praktická část
5. Zhodnocení výsledků a diskuze

Kapitola „*Teoretická východiska*“ se bude v úvodní části zabývat vybranými modely životního cyklu vývoje softwaru a metodikami vývoje softwaru a testování, přičemž bude kladen důraz na pozici a význam testování v rámci jednotlivých modelů a metodik. Další kapitoly budou věnovány technikám testování softwaru, rolím rozlišovaným v rámci testování, tvorbě testovací strategie ve formě test plánu, návrhu testovacích případů a zaznamenávání chyb. Tím dojde k naplnění prvního z dílčích cílů práce. V závěru kapitoly budou popsány možné způsoby vyhodnocení testování.

V úvodu kapitoly „*Praktická část*“ bude popsán testovaný systém včetně dokumentace k zaváděným změnám. Ta bude následně představovat zadání pro zbytek kapitoly. Dále bude navržena testovací strategie ve formě test plánu, následně bude provedena test analýza, při níž budou navrženy jednotlivé testovací případy, které budou v následující kapitole prováděny. Tím dojde ke splnění hlavního cíle diplomové práce. Součástí realizace bude aplikace vybrané metody a přístupu k testování, čímž dojde ke splnění druhého z dílčích cílů práce.

V kapitole „*Zhodnocení výsledků a diskuze*“ bude provedeno vyhodnocení konkrétních výsledků testování. Dále budou diskutovány další možné způsoby testování, přičemž bude vždy uvedeno doporučení, za jakých podmínek by bylo možné daný způsob testování aplikovat anebo naopak, proč by nebylo vhodné aplikovat daný způsob testování.

3 Teoretická východiska

3.1 Modely životního cyklu vývoje softwaru

„Model životního cyklu softwaru je proces, podle něhož se vytváří softwarový produkt, a to od jeho prvotního záměru až po uvedení na trh“. (PATTON, 2002)

Existuje celá řada metodik, které formalizují a detailně popisují postupy, jak efektivně řídit projekty vývoje softwaru, avšak žádnou z těchto metodik nelze označit za jedinou správnou. Příslušnou metodiku vývoje softwaru je nutné vybrat na základě jednotlivých aspektů konkrétního projektu. (PATTON, 2002)

3.1.1 Model velkého třesku

Jak již název napovídá, nejedná se o přesně strukturovaný model, který by explicitně definoval jednotlivé kroky při jeho aplikaci. Model nezahrnuje analýzu, plánování ani testování. Hlavní výhodou tohoto modelu je jeho jednoduchost. Veškeré úsilí je soustředěno na vývoj softwaru. Charakteristickými rysy tohoto modelu jsou ne zcela srozumitelné vstupní požadavky a neurčité datum dokončení vývoje softwaru. (SOMMERVILLE, 2013)

Jak je patrné, není tento přístup vhodný pro seriózní vývoj softwaru, u kterého je požadována určitá míra kvality, protože již hlavní myšlenka tohoto modelu takřka vylučuje jakékoliv smysluplné testování. (NAIK, a další, 2008)

3.1.2 Model „programuj a opravuj“

Přestože tento model není možné zařadit mezi efektivní modely vývoje softwaru, je možné ho označit za krok vpřed oproti předcházejícímu modelu. (VAN VLIET, 2008)

Proces vývoje softwaru začíná hrubou představou výsledného produktu, kdy je proveden jednoduchý návrh, ze kterého vývojový tým po zbytek projektu vychází. Dále následuje opakující se cyklus, kdy je naprogramována část kódu, která je testována a nalezené chyby jsou ihned opravovány a opětovně testovány. Po bezchybném otestování pokračuje znovu fáze programování další části kódu a celý proces se opakuje. (KOIRALA, a další, 2008)

Hlavní nevýhodou modelu je nedostatečný detail návrhu a stejně je tomu i v případě dokumentace. I přes tyto nevýhody je model vhodný pro malé projekty,

které jsou vytvořeny jako prototypy nebo demonstrační programy. Struktura tohoto modelu je natolik přirozená, že se často objevuje v nižších úrovních komplexnějších a složitějších modelů vývoje softwaru. (PATTON, 2002)

3.1.3 Model vodopádu

Model vodopádu je první z uvedených modelů, o kterém je možné říci, že je systematický a pokud je aplikován na vhodném projektu, funguje velmi dobře. Model se skládá z pěti kroků – nápad (specifikace požadavků, zadání) analýza, návrh, vývoj, testování. (PATTON, 2002)

Projekt vyvíjený pomocí tohoto modelu postupuje směrem dolů v posloupnosti jednotlivých kroků. Na konci každého kroku je hodnoceno, jestli je možné přejít do dalšího kroku. Pokud vývojový tým zjistí, že projekt není zralý přejít dále, zůstává na stejné úrovni do té doby, dokud není úroveň dokončena. (VAN VLIET, 2008)

Fáze vývoje se nachází v jediném bloku. Jednotlivé kroky se nepřekrývají. Ve vodopádovém modelu není možné vrátit se zpět. Jakmile se projekt nachází v určitém kroku, je zapotřebí tento krok dokončit a následně přejít do dalšího. (SOMMERVILLE, 2013)

Hlavní výhodou modelu je, že klade důraz na specifikaci výsledné podoby produktu. Jedná se o velmi obtížnou část, která se ne vždy podaří splnit tak, aby nevznikly ve specifikaci mezery. Při vývoji a testování pak nevznikají žádné dodatečné otázky, protože vše je předem dohodnuto, specifikováno a výsledek je velmi kvalitní. Nevýhodou modelu je, že fáze testování probíhá až na konci celého procesu vývoje softwaru. Testování může odhalit velké množství chyb, jejichž opravy pak mohou být finančně nákladné. (PATTON, 2002)

3.1.4 Spirálový model

Spirálový model byl zaveden v roce 1986 a postupem času se ukázalo, že se jedná o velmi efektivní model vývoje softwaru. (VAN VLIET, 2008)

Základní myšlenkou modelu je, že na začátku projektu není možné podrobně definovat všechny požadavky. Začíná se tedy definicí těch nejdůležitějších funkcionálních požadavků, které jsou následně implementovány, otestovány, zákazník k nim poskytne zpětnou vazbu a poté se pokračuje další úrovní vývoje softwaru. Takto se proces opakuje,

dokud není implementován kompletní výsledný produkt. Celý model vychází z posloupností následujících kroků, které jsou poté iterativně opakovány. (PATTON, 2002) rozlišuje tyto fáze:

- určení cílů, alternativ a omezení,
- rozpoznání a řešení rizik,
- vyhodnocení alternativ,
- vývoj a testování aktuální úrovně,
- plánování další úrovně,
- rozhodnutí o postupu na další úroveň.

Součástí spirálového modelu je částečně vodopádový model (také obsahuje kroky analýzy, návrhu, vývoje a testování) a částečně také model „programuj a opravuj“ (v každém průchodu spirálou). Oproti předchozím modelům je velkou výhodou spirálového modelu schopnost včasného odhalení chyb, což přispívá ke snížení nákladů na vývoj softwaru. (VAN VLIET, 2008)

Spirálový model je díky svým výhodám v praxi poměrně často využívaný a z hlediska testování softwaru se jedná o velmi efektivní model, kdy má testovací tým možnost ovlivňovat projekt od jeho začátku, protože je zapojen i do počátečních fází vývoje. Po každé iteraci je produkt testován, což zajišťuje kontrolu kvality vyvíjeného softwaru v průběhu celého projektu. (PATTON, 2002)

3.2 Metodiky vývoje softwaru a testování

Kvalita softwaru závisí na řadě faktorů, ale jedním z těch nejdůležitějších, který ovlivňuje jeho kvalitu, je proces vývoje softwaru. Tato kapitola se bude zabývat rolemi procesu testování v rámci jednotlivých metodik vývoje softwaru.

3.2.1 Extrémní programování (XP)

Metodika XP je určena pro software, jehož zadání není zcela jasné nebo v něm dochází k častým změnám. (BUCHALCELOVÁ, a další, 2008)

V metodice extrémního programování je proces testování součástí všech fází životního cyklu projektu. Testovací tým se podílí na odhadech pracnosti definovaných uživatelských scénářů. Ve fázích analýzy, návrhu a implementace probíhá testování průběžně (SOMMERVILLE, 2013)

Role procesu testování je v XP založena na přístupu Test-First Design, to znamená, že testy pro daný požadavek musí být vytvořeny před tím, než dojde k implementaci samotného požadavku. Aplikace přístupu TFD vede ke snížení rizika, že testy budou vytvořeny v závislosti na implementovaném požadavku, tj. tak, že sada testů bude sestavena, aby jimi implementovaný požadavek prošel. Zmíněné riziko hrozí, zejména pokud testování obstarávají sami programátoři, protože se v tomto případě může vytratit nezávislost a nezaujatost testování. (VAN VLIET, 2008)

Testování se v metodice XP týká rolí vývojář, tester a zákazník. Vývojář vytváří a vyhodnocuje jednotkové testy. Tester v metodice XP plní roli konzultanta (poradce), který pomáhá vývojáři a zákazníkovi s identifikací a sestavením testovacích případů, s návrhy na automatizaci či tvorbou testovacího prostředí. Dalšími úkoly testerů v metodice XP jsou tvorba odhadů rizik, definice požadavků na kvalitu a ověřování, že testy funkcionality skutečně testují definované uživatelské scénáře. (BUCHALCELOVÁ, a další, 2008)

3.2.2 Metodika RUP (Rational Unified Process)

Metodika RUP byla dříve řazena mezi rigorózní metodiky, ale v současnosti zahrnuje také agilní principy a je možné ji tedy využít k agilnímu vývoji softwaru. (NAIK, a další, 2008)

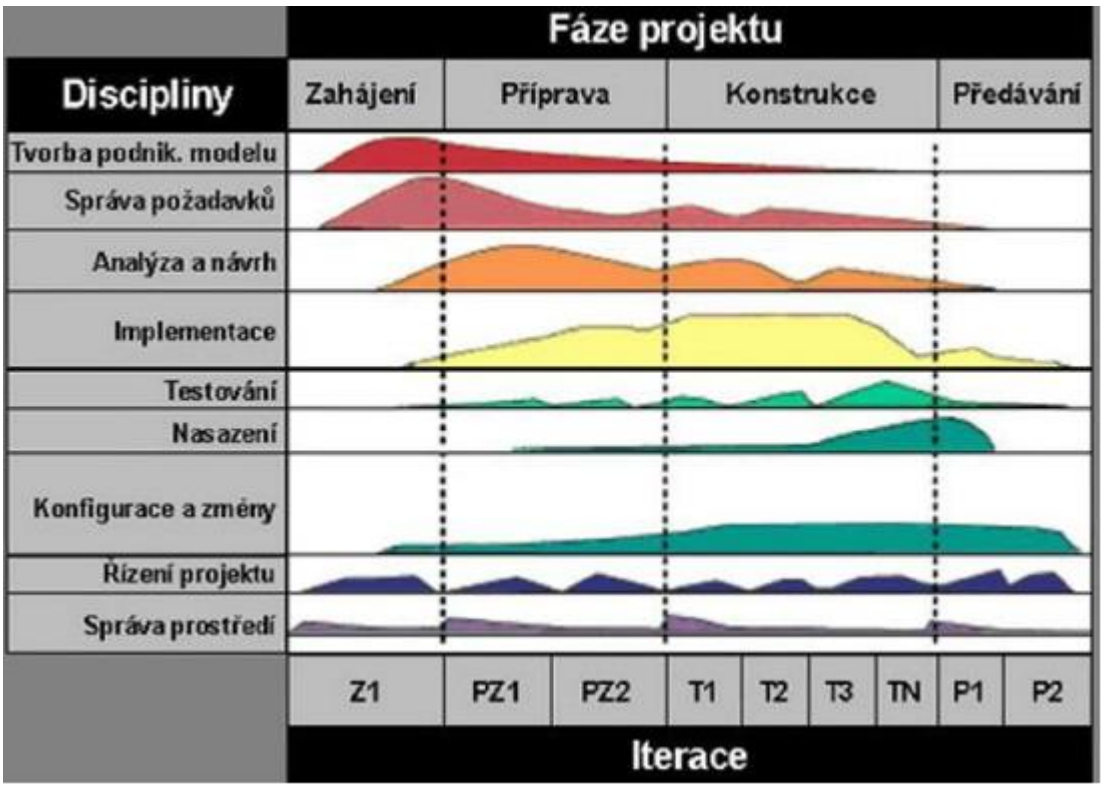
Prvním krokem testovacího procesu v metodice RUP je stanovení cílů testování. Poté následuje ověření základní stability a základních funkcí softwaru. V případě, že jsou nalezeny závažné chyby nebo je software nestabilní, následuje jeho vrácení zpět do vývoje. Pokud je software stabilní, je podroben kompletnímu testování. Současně s testováním probíhá tvorba zprávy o výsledcích testování určená manažerům projektu a dalším zainteresovaným osobám. Po úspěšném dokončení testování následuje fáze údržby, zkvalitňování testovací dokumentace, zkvalitňování vytvořených testovacích případů a dalších testovacích náležitostí, které bývají v metodice RUP souhrnně označovány jako Test Assets. Cílem těchto činností po testování je udržovat testovací náležitosti aktuální tak, aby tvořily základnu pro případné další testování. Celý proces může být realizován opakovaně v tzv. iteracích. Souběžně s testováním probíhá proces, jehož cílem je ověřit vhodnost zvoleného přístupu testování a posoudit kvalitu jeho výsledků. (CHRISPIN, a další, 2009)

V rámci testování se v metodice RUP rozlišují čtyři role: manažer testování, test analytik, návrhář testů a tester. Každé roli odpovídají specifické činnosti a pravomoci, s nimiž disponuje. Úkolem manažera testování je zastřešovat práci celého testovacího týmu, řídit proces testování a řešit problémy v něm vzniklé. Dále je manažer testování zodpovědný také za plánování zdrojů. Náplní role test analytika je identifikovat a popsat testovací případy, sledovat průběh testování, provést hodnocení kvality dosažené během testování a zvolit vhodnou prezentaci výsledků testování. Návrhář testů je zodpovědný za definici přístupu k testování a jeho úspěšnou realizaci, což zahrnuje doporučení testovacích nástrojů, časové odhady a identifikaci vhodných postupů. Úkolem testera je tvorba testovacích případů, jejich exekuce a vyhodnocení. (BUCHALCELOVÁ, a další, 2008)

V případě identifikace chyby je úkolem testera chybu zaznamenat a popsat vhodným způsobem do příslušného systému tak, aby mohla být co nejrychleji opravena. V metodice RUP probíhají také jednotkové testy, které jsou prováděny vývojáři příslušného softwaru. Za integraci a provedení integračních testů je poté zodpovědná role integrátora. Za fázi předání softwaru, řízení akceptačního a beta testování odpovídá tzv. Deployment Manager (manažer nasazení). Při akceptačním testování jsou obvykle přítomni zástupci ze strany zákazníka, kteří musí produkt schválit a podepsat akceptační

protokol. Během beta testování jsou potom prováděny testy v prostředí zákazníka jeho zaměstnanci. (LEFFINGWELL, 2011)

Jak již bylo zmíněno v přechozích kapitolách práce, testování by mělo probíhat v průběhu celého procesu vývoje softwaru. Na obrázku č. 1 je znázorněno zastoupení jednotlivých činností procesu testování v jednotlivých fázích vývoje softwaru v rámci metodiky RUP.



Obrázek 1 Testovací činnosti v metodice RUP, Zdroj: (Testování softwaru)

3.3 Úvod do oblasti testování softwaru

3.3.1 Charakteristika testování softwaru

Testování je obecně vnímáno jako proces, v rámci něhož jsou prováděny jednotlivé akce pro ověření funkčnosti softwaru. To je však pouze jedna z mnoha aktivit v rámci procesu testování.

Aktivity spojené s testováním existují nejen během samotného provádění testů, ale také před a po vykonání testů. Do těchto aktivit patří plánování a řízení, výběr testovacích podmínek, návrh testovacích případů, kontrola výsledků jednotlivých kroků testovacích případů, vyhodnocování výstupních kritérií, tvorba reportů o průběhu testovaní a aktivity probíhající po ukončení fáze testování. Proces testování zahrnuje také revizi dokumentů (např. technické dokumentace) a statickou analýzu. (MYERS, a další, 2012)

Cíle testování jsou závislé na úhlu pohledu, který může být různý. Vývojáři provádějí testování komponent (metod, tříd, modulů …) jehož cílem může být vyvolání co největšího počtu selhání s cílem identifikovat a opravit defekty. V rámci akceptačního testování může být hlavním cílem potvrzení požadovaných funkcí systému, tedy že systém pracuje podle očekávání. Někdy také může být hlavním cílem testování ohodnocení kvality softwaru (bez snahy opravovat chyby) nebo poskytnutí informací o rizicích, která přináší uvolnění systému v daném čase. (FALLEY-VINAY, 2008)

V souvislosti s testováním bývají často zaměňovány dva pojmy: ladění a testování. Ladění je vývojová aktivita, která analyzuje zdrojový kód, nachází a odstraňuje příčiny selhání. Testování slouží například ke snaze ověřit funkčnost softwaru. Z tohoto vyplývá, že vývojáři ladí a testeři testují. (ISTQB, 2013)

3.3.2 Základní pojmy z oblasti testování softwaru

Před samotnou definicí pojmu testování softwaru je nutné definovat některé základní pojmy, které se v oblasti testování softwaru často objevují.

Chyba

Za chybu v softwaru je možné označit jeden z následujících případů:

- software po vykonání kódu dělá, co by podle specifikace udělat neměl,
- software dělá něco, co by podle specifikace dělat neměl,

- software dělá něco, co není ve specifikaci zmíněno,
- software je nesrozumitelný, pomalý, obtížně se s ním pracuje nebo vykazuje jiné vlastnosti, které mohou bránit jeho efektivnímu použití. (ISTQB, 2013)

Člověk se může dopustit omylu. Jeho následkem je defekt (chyba) v kódu programu nebo v dokumentaci. V případě chyby v kódu programu, neudělá software po vykonání kódu to, co by udělat měl (buď může udělat něco jiného, nemusí ale udělat nic, nebo udělá něco, co není ve specifikaci zmíněno) a způsobí tak selhání. Ne vždy musí defekt způsobit selhání. (například špatné definování přístupových práv pro jednotlivé uživatelské role.) (PATTON, 2002)

K defektům dochází, protože jsou lidé omylní a pracují pod časovým tlakem, v komplikované infrastruktuře firmy, se složitým kódem, stále se měnícími technologiemi a také z důvodu vzájemného působení systémů. (FALLEY-VINAY, 2008)

Správnost a přesnost

Správnost a přesnost patří k jedněm z nejdůležitějších vlastností kvalitního softwaru, a proto je nutné obě tyto vlastnosti řádně otestovat. (BLACK, 2002)

Při testování správnosti je ověřováno, jestli program pracuje podle požadavků, tedy jestli po zadání vstupních údajů a volbě operací, udělá přesně to, co by udělat měl, bez ohledu na přesnost výstupu. Testování přesnosti ověřuje, že program poskytuje výsledky s požadovanou přesností. (ISTQB, 2013)

V praxi je testování těchto vlastností spojeno do jednoho procesu, kdy je nejprve testována správnost softwaru, a pokud software pracuje správně, potom je ověřováno, že výsledek odpovídá požadované přesnosti. (NAIK, a další, 2008)

Verifikace a validace

Verifikace je proces, jehož cílem je provést ověření, zda software vyhovuje zadané specifikaci. Validace je procesem kontroly, jestli software vyhovuje požadavkům uživatelů a dalších zájmových skupin. (TIAN, 2005)

Kvalita a spolehlivost

Software může být označen za spolehlivý, pokud je stabilní, důvěryhodný a neobsahuje žádné chyby technického charakteru. (ISTQB, 2013)

Spolehlivost je jedním z mnoha aspektů kvality. Dalšími aspekty kvality jsou funkčnost, použitelnost, výkon a podporovatelnost. (FALLEY-VINAY, 2008)

Pokud má být zajištěna vysoká kvalita a spolehlivost softwaru současně, musí být během celého procesu vývoje softwaru prováděna odpovídající verifikace a validace. (JORGENSEN, 2008)

Testování a zajišťování kvality

Pojmy testování a zajišťování kvality bývají často označovány zkratkou QA (quality assurance). QA popisuje proces, jehož úkolem je zajišťovat verifikaci a validaci testovaného softwaru. (TIAN, 2005)

Testování softwaru je možné definovat jako průběžný proces, jehož cílem je po celou dobu vývoje a testování nalezení chyb (přesněji selhání), provedení jejich identifikace a zajištění co nejrychlejší opravy. (MYERS, a další, 2012)

Zajišťování kvality je proces, jenž vytváří a aplikuje standardy a metody, které vedou ke zvýšení kvality softwaru. (BURNSTEIN, 2003)

3.3.3 Principy testování softwaru

Testování je velmi často spojeno s kompromisy a je specifické pro jednotlivé projekty. V testovacím procesu však existuje několik známých pravd tzv. axiomů, které by měly být dodržovány bez ohledu na charakter projektu nebo formu testování. (ISTQB, 2013)

Vyčerpávající testování je nemožné

Kompletní testování celého softwaru je nemožné s výjimkou triviálních případů. Úkolem procesu testování není otestovat vše, ale vybrat jednotlivé oblasti testovaného softwaru na základě analýzy rizik a stanovení priorit. (PATTON, 2002)

Testování ukazuje přítomnost defektů

Testování může ukázat přítomnost defektů, ale nemůže dokázat, že testovaný software neobsahuje žádné defekty. Testování tedy nevylučuje výskyt defektů, ale snižuje pravděpodobnost jejich výskytu. (BLACK, 2002)

Včasné testování

Testovací aktivity musí začít v rámci životního cyklus vývoje softwaru, co nejdříve a musí být zaměřeny na cíle. (ISTQB, 2013)

Shlukování defektů

Testování nemusí být vždy zaměřeno rovnoměrně na celý testovaný software. Malé množství modulů velmi často obsahuje většinu defektů nalezených v průběhu testování. (PATTON, 2002)

Pesticidní paradox

Pokud jsou opakovány stále stejné sady testovacích případů, postupem času soubor testovacích případů již nenalezne žádné defekty. K překonání „pesticidního paradoxu“ je zapotřebí provádět pravidelné revize a úpravy testovacích případů (FALLEY-VINAY, 2008)

Testování je závislé na kontextu

Způsob testování je závislý na jeho kontextu. Software u něhož budeme klást vysoké požadavky na bezpečnost, budeme testovat jiným způsobem než webovou aplikaci pro zákazníky mobilních operátorů. (ISTQB, 2013)

Falešná představa o neexistenci omylů

Vytvořený systém musí splňovat potřeby a očekávání zákazníků, jinak nalezení a oprava defektů nepomůže ke zvýšení jeho kvality. (PATTON, 2002)

3.3.4 Typy testů

Podle účelu testování je rozlišováno několik typů testů. Tato kapitola bude věnována jejich představení.

Testování funkcionality softwaru

Je zaměřeno na funkcionalitu, která má být systémem vykonávána. Tato funkcionalita může být popsána ve specifikacích požadavků, v případech užití nebo ve funkcionálních specifikacích. Funkcionální testování zkoumá externí chování systému a může být vykonáváno na všech úrovních testování. Pro jeho realizaci

se využívají techniky testování založené na specifikaci (viz kapitola 3.7.2). (NAIK, a další, 2008)

Testování nefunkcionálních charakteristik softwaru

Testování nefunkcionálních charakteristik zahrnuje testování výkonu, zátěžové testování, stres testování, testování použitelnosti, testování spolehlivosti, testování udržovatelnosti a testování přenositelnosti. Obecně se jedná o testování toho, „jakým způsobem“ systém pracuje. (ISTQB, 2013)

Vzhledem k tomu, že se jedná o nefunkcionální testování, jsou pro měření těchto testů vyžadovány měřitelné charakteristiky softwaru, které mohou být poté následně porovnány s vybraným stupnicemi měření. Nefunkcionální testování se může odvolávat na kvalitativní modely, jako například na model definovaný ve standardu Softwarové inženýrství – Kvalita softwarového produktu (ISO 9126). (NAIK, a další, 2008)

Testování struktury softwaru

Testování struktury bývá někdy také označováno jako testování bílé skříňky. Strukturální testování využívá technik založených na specifikaci s cílem měření důkladnosti testování pomocí pokrytí vybraných struktur (viz kapitola 3.7.2). (MYERS, a další, 2012)

Testování struktury bývá nejčastěji aplikováno při testování komponent a integračním testování, kde mohou být využity nástroje na měření pokrytí kódu nebo rozhodování.

Testování související se změnami a regresní testování

Po nalezení a opravě defektu (selhání) by měl být testovaný software znovu otestován s cílem potvrdit správnost opravy a odstranění defektu. Tento proces opětovného testování za účelem ověření opravy defektu bývá také označován jako konfirmační testování. (FALLEY-VINAY, 2008)

Regresní testování je opakované testování již dříve otestovaného softwaru po provedení úpravy s cílem nalézt všechny defekty, které mohly být provedením příslušné opravy zaneseny do softwaru. Regresní testování se také provádí, pokud dochází ke změnám prostředí, na němž byl daný systém před tím tzv. nasazen (systém fungoval

na daném prostředí). Rozsah regresního testování je závislý na míře rizika výskytu defektů v softwaru. (MYERS, a další, 2012)

3.4 Techniky testování softwaru

V současnosti existuje celá řada technik pro testování softwaru. Tyto techniky se člení podle několika hledisek a právě rozčleněním a vysvětlením rozdílu mezi jednotlivými technikami testování se bude tato kapitola zabývat.

Jak již bylo v práci zmíněno, testování není jednorázový proces, který probíhá před tím, než je software předán zákazníkovi nebo před uvedením do provozu, ale jedná se o průběžnou aktivitu, jejíž vybrané formy by měly být aplikovány v jednotlivých fázích vývoje softwaru. Obecně platí, že čím dříve je chyba odhalena, tím nižší jsou náklady na její odstranění. Jako další argument pro podporu průběžného testování je možné uvést fakt, že pokud je nalezeno příliš mnoho chyb najednou, může dojít k naprostému zastavení testování, které je zapříčiněno buď množstvím, nebo závažností nalezených chyb, a dojde tak ke zdržení celého projektu. (VAN VLIET, 2008)

3.4.1 Techniky testování software podle způsobu provedení

Obecně se techniky testování rozlišují na techniky černé skříňky (tzv. blackbox testování) a techniky bílé skříňky (tzv. whitebox testovaní).

Techniky testování černé skříňky (označované také jako techniky založené na specifikaci) slouží pro odvození a výběr testovacích podmínek, testovacích případů nebo testovacích dat. Techniky založené na specifikaci vycházejí z analýzy dokumentace a zahrnují jak funkcionální, tak nefunkcionální testování. Techniky testování černé skříňky nepoužívají žádnou informaci o vnitřní struktuře komponenty nebo systému, který je předmětem testování. (PERRY, 2006)

Techniky testování bílé skříňky (označované jako techniky založené na struktuře) jsou založené na analýze struktury komponenty nebo systému. V praxi bývají často obě techniky testování kombinovány s technikami založenými na zkušenostech, za účelem zužitkování zkušeností členů vývojového týmu, testerů i samotných uživatelů, pro stanovení předmětu testování. (VAN VLIET, 2008)

3.4.2 Techniky testování softwaru podle úrovně vývoje

Dalším kritériem, podle něhož je možné rozčlenit techniky testování softwaru je úroveň vývoje. Úrovní vývoje je v této kapitole myšleno, v jaké fázi vývoje se testovaný software nachází. (JORGENSEN, 2008)

Testování komponent

Testování komponent je také někdy označováno jako jednotkové testování nebo testování modulů. Jeho cílem je hledání defektů uvnitř softwarových komponent (modulů, tříd, objektů atd.), které jsou samostatně testovatelné. Může být vykonáváno odděleně od zbytku systému. Jednotkové testování provádí zpravidla programátor. (KANER, a další, 2002)

Součástí testování komponent může být testování funkcionality, testování specifických nefunkcionálních charakteristik (např. hledání přetečení paměti), anebo testování robustnosti, jakož i strukturální testování (např. pokrytí rozhodování). (PERRY, 2006)

Integrační testování

Cílem integračního testování je ověření rozhraní mezi komponentami nebo interakce s různými částmi systému. Existuje více úrovní integračního testování, které může být vykonáváno následujícími způsoby:

- integrační testování komponent prověřuje interakci mezi komponentami softwaru a je vykonáváno po testování komponent,
- systémové integrační testování prověřuje interakci mezi různými systémy nebo mezi softwarem a hardwarem a může být vykonáno po systémovém testování. (ISTQB, 2013)

Čím větší je rozsah integračního testování, tím složitější je lokalizovat defekty na určitý systém, což může mít za následek zvýšení rizika a časové náročnosti odstranění příslušného defektu. (BURNSTEIN, 2003)

Testovací tým by se měl v rámci integračního testování zaměřit pouze na samotnou integraci jednotlivých systémů. Pokud je předmětem testování například integrace systému A se systémem B, měl by se testovací tým zaměřit výhradně na komunikaci mezi těmito

dvěma systémy. V rámci uvedeného integračního testování by nemělo docházet k provádění testování funkcionality jednotlivých systémů, protože to není předmětem integračního testování. (WATKINS, 2001)

Systémové testování

Náplní systémového testování je chování celého systému (produktu). Rozsah a cíle testování musí být explicitně vymezeny v testovacím plánu. (ISTQB, 2013)

V případě systémového testování by se mělo testovací prostředí co nejvíce podobat produkčnímu prostředí tak, aby bylo minimalizováno riziko výskytu defektů vlivem odlišnosti testovacího a produkčního prostředí. (FALLEY-VINAY, 2008)

Systémové testy mohou být založeny na specifikaci požadavků, na specifikaci business procesů, na případech užití či na jiných popisech systému. (MILI, a další, 2015)

Systémové testování by mělo ověřit nejen funkcionální a nefunkcionální požadavky na testovaný systém, ale také kvalitu a správnost dat v systému. Systémové testování funkcionálních požadavku je prováděno pomocí vhodně zvolených technik založených na specifikaci (tzv. techniky černé skříňky), které při testování zohledňují povahu testovaného systému. Poté mohou být aplikovány techniky založené na struktuře (tzv. techniky bílé skříňky). (TIAN, 2005)

Akceptační testování

Cílem akceptačního testování je upevnění důvěry v testovaný systém. Při akceptačním testování není hlavním cílem nalezení defektů. Akceptační testování může sloužit pro vyhodnocení připravenosti systému pro nasazení do provozu. (ISTQB, 2013)

Akceptační testování se může vyskytovat v různých fázích životního cyklu vývoje softwaru:

- krabicový softwarový produkt může být akceptačně testován, když je instalován nebo integrován,
- akceptační testování použitelnosti komponenty může být vykonáváno v průběhu testování komponenty,
- akceptační testování nového funkcionálního rozšíření může být realizováno před systémovým testováním. (ISTQB, 2013)

V souvislosti s akceptačním testováním se často objevují dva pojmy, konkrétně je to alfa a beta testování. Alfa testování je vykonáváno v prostředí vyvíjející organizace, ne však týmem vývojářů. Beta testování (někdy označováno jako testování v terénu) je vykonáváno zákazníky v jejich vlastním prostředí (nebo zaměstnanci zákazníka v pracovním prostředí zákazníka). (SPILLNER, 2011)

3.5 Role v testování

Testování jako komplexní proces zahrnuje celou řadu činností od plánování testů, přes analýzu a návrh testů, po samotné vykonávání testů a reportování nalezených chyb. Z uvedených činností vyplývají role, které jsou rozlišovány v rámci testování. Jednotlivé metodiky a přístupy testování se mohou lišit nejen v přístupu k samotnému testování, ale také v rolích, které se v rámci tohoto procesu rozlišují. (FALLEY-VINAY, 2008)

Následující podkapitoly budou zaměřeny na definování rolí v rámci testování. (CHRISPIN, a další, 2009)

3.5.1 Manažer testování (Test manager)

Osoba pověřená rolí manažera testů zodpovídá za realizaci celého procesu testování. Mezi úkoly manažera testování patří správa testů, jejich plánování, řízení zdrojů a řešení problémů vzniklých během fáze testování (např. nefunkční testovací prostředí, nemožnost tvorby testovacích dat, atd.). (PATTON, 2002)

Osoba vykonávající roli manažera testování by měla mít následující dovednosti:

- znalost procesu vývoje softwaru,
- zkušenosti s testovacími technikami a nástroji,
- schopnost komunikovat a jednat s lidmi,
- schopnost dobrého plánování a řízení,
- znalost z oblasti testovaného softwaru. (BUCHALCELOVÁ, a další, 2008)

Osoba vykonávající roli manažera testování je zodpovědná za:

- znalost procesu vývoje softwaru,
- zkušenosti s testovacími technikami a nástroji,
- schopnost komunikovat a jednat s lidmi,
- schopnost dobrého plánování a řízení,
- znalost z oblasti testovaného softwaru. (BUCHALCELOVÁ, a další, 2008)

3.5.2 Analytik testování (Test analyst)

Hlavní náplní práce role analytik testování je identifikace a definice oblastí, které je zapotřebí otestovat. Mezi další aktivity této role patří sledování progresu v průběhu testování, kontrola výstupu jednotlivých testů a také hodnocení kvality softwaru. Analytik testování také v projektu reprezentuje tzv. stakeholders (zájmové skupiny), jinými slovy osoby, které jsou v projektu zainteresované, ale nejsou členy vývojového týmu. (BLACK, 2002)

Osoba vykonávající roli analytika testování by měla mít následující dovednosti:

- analytické myšlení,
- smysl pro detail,
- znalost testovaného systému,
- zkušenosti s testováním softwaru. (BUCHALCELOVÁ, a další, 2008)

Osoba vykonávající roli analytika testování je zodpovědná za:

- identifikaci oblastí softwaru, které budou předmětem testování,
- definici testovacích případů a testovacích dat,
- hodnocení výstupů testovacích případů. (BUCHALCELOVÁ, a další, 2008)

3.5.3 Návrhář testů (Test designer)

Návrhář testů zajišťuje definici přístupu k testování a jeho následnou implementaci. Náplní jeho práce je výběr vhodných technik a nástrojů testování s ohledem na dostupné technické a personální zdroje. (PATTON, 2002)

Osoba vykonávající roli návrháře testů by měla mít následující dovednosti:

- znalosti v oblasti hardware i software,
- schopnost identifikace a řešení problémů,
- zkušenosti s testováním softwaru,
- zkušenosti s nástrojem pro automatizované testování,
- detailní znalost testovaného softwaru. (BUCHALCELOVÁ, a další, 2008)

Osoba vykonávající roli návrháře testů je zodpovědná za:

- identifikaci a popis vybraných postupů testování,
- identifikaci a výběr nástrojů pro podporu testování,
- definici a údržbu architektury automatizovaných testů,
- specifikaci a kontrolu testovacího prostředí. (BUCHALCELOVÁ, a další, 2008)

3.5.4 Tester (Tester)

Úlohou testera je provádění jednotlivých testovacích případů, dokumentování výsledků jednotlivých kroků v rámci testovacího případu, reportování chyb a opětovné testování (tzv. retest) po opravách chyb. (FALLEY-VINAY, 2008)

Osoba vykonávající roli testera by měla mít následující dovednosti:

- znalost navržených postupů testování,
- schopnost rozpoznání a řešení problémů,
- zkušenosti s nástroji pro automatizované testování,
- schopnost lokalizace chyb,
- znalost testovaného softwaru,
- základy programování. (BUCHALCELOVÁ, a další, 2008)

Osoba vykonávající roli testera je zodpovědná za:

- implementaci jednotlivých testovacích případů,
- dokumentování výsledků testů a kontrolu jejich správného průběhu,
- analýzu chyb a jejich reportování. (BUCHALCELOVÁ, a další, 2008)

3.6 Plánování testů

Proces testování nemůže probíhat nahodile nebo neurčitě. Všechny úkoly a činnosti spojené s testováním musí být uspořádány. Bylo by velmi obtížné provádět kvalitní testování, pokud by programátoři nespecifikovali, jak by jimi implementované řešení mělo fungovat a kdy bude hotové. Také testovací tým by měl následně sdělit, co bude předmětem testování, jaké prostředky budou v rámci testů využity a jaký bude časový harmonogram testování. V případě, že by nebyly výše uvedené skutečnosti splněny, projekt by měl pouze velmi malou šanci na úspěch. Testovací plán je nezbytným a velmi důležitým prostředkem, v rámci něhož testovací tým definuje pro vývojový tým a ostatní členy projektu své záměry a vytváří tak testovací strategii. (BLACK, 2002)

Standard pro dokumentaci testů softwaru (Software Test Documentation) definuje, že smyslem testovacího plánu softwaru je:

„Předepsat rozsah, postup, prostředky a časový plán aktivit spojených s testováním. Identifikovat jednotlivé testované položky, testované funkce a úkoly prováděné při testování, konkrétní osoby odpovědné za každý z úkolu a rizika spojená s definovaným plánem.“ (ANSI/IEEE č. 829, 1998)

Plán testů je velmi důležitým dokumentem z pohledu jednotlivých činností a jejich načasování v rámci procesu testování. Nicméně je třeba si uvědomit, že se jedná pouze o vedlejší produkt procesu plánování, který je tím hlavním důvodem, proč je plán testů vytvářen. Výsledný dokument pouze vyjadřuje všechny důležité skutečnosti v písemné podobě. (PATTON, 2002)

„Konečným cílem procesu plánování testů je sdělit (nikoliv pouze zaznamenat) záměry testovacího týmu, jeho očekávání a celkové chápání prováděných testů.“ (PATTON, 2002)

Existují různé šablony, které definují podobu testovacího plánu, nicméně hlavní problém tohoto přístupu spočívá v soustředění hlavní pozornosti na podobu dokumentu a následné opomíjení samotného procesu plánování. V praxi se pak velmi často stává, že manažeři testování využívají jakousi univerzální šablonu, kdy pouze mění údaje o konkrétním projektu bez ohledu na jeho charakter a potřeby. To je však zásadní chybou, protože takto vytvořený plán testů nesplňuje svůj účel. Správně sestavený testovací plán

poskytuje informace o jednotlivých činnostech testovacího týmu ostatním členům projektového týmu. (SPILLNER, 2011)

Jak již bylo uvedeno v předchozím textu, plán testů by neměl být tvořen podle předem stanovené šablony, nicméně by měl vždy splňovat následující náležitosti. (FALLEY-VINAY, 2008)

3.6.1 Vymezení produktu a stanovení cílů testování

Na počátku tvorby testovacího plánu je nutné vymezit softwarový produkt, který bude předmětem testování. Dále je nutné definovat cíle uvedeného produktu z hlediska kvality a spolehlivosti.

Tato část testovacího plánu by měla odpovídat na otázky:

- Jaký je účel a význam testování?

Je nutné, aby byly všichni členové projektového týmu seznámení s důvodem testování příslušného softwarového produktu. Často se může stát, že je testovací tým negativně vnímán analytiky nebo programátory, proto je nezbytné, aby všichni členové projektového týmu znali důvod testování a chápali testování jako činnost vedoucí k zajištění stanovené míry kvality softwarového produktu. (SPILLNER, 2011)

- Jaký produkt bude testován?

Testování nemůže být nikdy úspěšné, pokud nebude jasně stanovena podoba projektu, jeho velikost a rozsah. Z těchto důvodů je nutné uvést, jestli se bude jednat o testování nového softwarového produktu, nebo o testování nové verze stávajícího produktu, nebo bude předmětem testování integrace příslušného softwaru s jiným softwarovým produktem. (PATTON, 2002)

- Jaké jsou cíle v oblasti kvality a spolehlivosti produktu?

Zde nastává velký prostor pro diskuzi jednotlivých členů projektového týmu, nicméně konečná podoba cílů v oblasti kvality a spolehlivosti produktu musí být odsouhlasena všemi členy vývojového týmu. V tomto případě dochází ke střetu zájmů jednotlivých členů projektového týmu. Například obchodní zástupce zastává názor, že software by měl být co nejrychlejší. Programátor zastává názor, že by pro implementaci

daného řešení měly být využity moderní a efektivní technologie. Vedoucí produktové podpory trvá na tom, že produkt nesmí obsahovat chyby. (FALLEY-VINAY, 2008)

Výsledkem procesu plánování musí být tedy zcela jasná a všemi stranami projektového týmu odsouhlasená definice cílů kvality a spolehlivosti daného softwarového produktu. Cíle musí být uvedeny explicitně, aby nebylo pochyb o jejich splnění či nesplnění. (SPILLNER, 2011)

3.6.2 Lidé, místa a věci

Dalším krokem při tvorbě plánu testů je určení osob, které budou členy testovacího týmu, stanovení jejich úkolů v rámci týmu a způsobu jejich případného kontaktování. Velmi žádoucí je uvést tento popis nejen u jednotlivých členů testovacího týmu, ale u všech členů projektového týmu, tedy i u analytiků, vývojářů, manažerů, atd. (PATTON, 2002)

Obdobným způsobem je zapotřebí popsat místo uložení jednotlivých dokumentů, místo pro možné stažení nové verze softwaru, umístění testovacích nástrojů atd. (JORGENSEN, 2008)

Pokud je zapotřebí k provádění testů určitý hardware, je nutné uvést, kde se nachází a kdy je možné jej využívat. (SPILLNER, 2011)

3.6.3 Definice názvosloví

Klíčovým faktorem úspěchu celého projektu jsou jasné definice a názvosloví pro všechny oblasti, které jsou předmětem testování. Každý člen projektového týmu musí být s příslušnými definicemi a názvoslovím seznámen. Je nepřípustné, aby během realizace projektu vznikaly problémy zapříčiněné vzájemným nepochopením členů projektového týmu. Takto vzniklé problémy s sebou nesou vysokou pravděpodobnost zanesení chyb do softwaru. (FALLEY-VINAY, 2008)

3.6.4 Definice odpovědností a povinností v rámci skupin

Na projektu obvykle pracuje několik skupin – analytici, vývojáři, management, testeři a další. Klíčovou rolí při vytváření jakéhokoliv plánu je stanovení odpovědností. V této fázi plánu ještě není možné stanovit odpovědnosti jednotlivých členů projektového týmu, ale je možné a žádoucí, stanovit odpovědnost jednotlivých skupin projektového

týmu, tedy za co nese odpovědnost tým vývojářů, tým testerů atd. Je nutné zaměřit se zejména na ty činnosti, které jsou na pomyslných hranicích pravomocí jednotlivých týmů a mohly by být řešeny různými skupinami v rámci projektového týmu. (ISTQB, 2013)

Efektivní a přehledné řešení je znázorněno pomocí tabulky č. 1, která poskytuje informace o činnostech, které budou realizovány v rámci projektu. Součástí tabulky jsou také informace o skupinách, které jsou za realizaci těchto činností zodpovědné. (PATTON, 2002)

Činnost	Vedení projektu	Vývojový tým	Testovací tým	Tvůrci dokumentace	Marketing	Podpora produktu
Sepsání vize produktu						x
Sepsání smluv	x					
Návrh produktu a jeho funkcí	x					
Návrh produktu a jeho specifikací	x					
Interní architektura produktu		x				
Návrh a kódování produktu		x				
Plánování testů			x			
Revize plánu testů			x			
Testování jednotek		x				
Obecné testování			x			
Revize tištěných materiálů				x		
Vytvoření seznamu konfigurací					x	

Tabulka 1 Odpovědnost skupin za jednotlivé činnosti v rámci projektu, Zdroj: (PATTON, 2002), str. 216

3.6.5 Co bude předmětem testování

V této fázi tvorby test plánu je nutné rozhodnout, které části softwarového produktu budou testovány a které části budou z testování vyjmuty. V případě, že se některé komponenty softwarového produktu nebudou testovat, mělo by být toto rozhodnutí odůvodněno. Mezi důvody vyjmutí komponent z testování může patřit například

implementace některých již hotových a otestovaných komponent, kdy již není nutné jejich funkcionalitu znovu testovat. (FALLEY-VINAY, 2008)

3.6.6 Fáze testování

Fáze testování závisí na zvoleném modelu či metodice vývoje příslušného softwarového produktu. Metodiky obvykle obsahují testování a jeho role v nich je jednoznačně stanovena. V rámci jednotlivých fází je nutné explicitně stanovit kritéria, podle nichž bude možné rozhodnout o dokončení příslušné fáze a zahájení následující. (JORGENSEN, 2008)

3.6.7 Strategie testování

V rámci strategie testování je nutné definovat postupy, které budou využity při testování, a to jak v rámci celého procesu testování, tak v rámci jednotlivých fází testování. Při rozhodování o volbě strategie testování je zapotřebí udělat následující rozhodnutí (BLACK, 2002):

- testovat vlastními silami, nebo využít služeb externí firmy?
- na jaké komponenty aplikovat white-box testování a na jaké black-box testování?
- pro jaké komponenty využít manuální a pro jaké automatické testy?
- v případě automatického testování použít vlastní řešení nebo zakoupit komerční nástroj – pokud ano, tak jaký? (KOIRALA, a další, 2008)

3.6.8 Požadavky na prostředky

Při plánování požadavků na prostředky je nutné zvážit jaké zdroje a jaké množství těchto zdrojů bude zapotřebí pro realizaci zvolené strategie testování. Je nutné zvážit veškeré zdroje, které by mohly být v průběhu realizace projektu k testování využity. Mezi tyto zdroje patří např.:

- osoby – Kolik lidí bude zapotřebí? S jakými zkušenostmi a s jakou odborností?
- vybavení – Počítače, hardware pro testování a další.
- prostory a laboratoře – Kde se budou prostory nacházet? Jak velké prostory budou zapotřebí a jak budou uspořádány?

- software – textové procesory, databáze, a další nástroje. Výběr licencí pro zakoupení.
- externí společnosti – Využijeme outsourcing? Jaká budou kritéria výběru? Jak nákladná bude spolupráce s externí firmou?
- další potřeby – Mobilní telefony, školící materiály, atd. (FALLEY-VINAY, 2008)

3.6.9 Pověření testerů

V této fázi tvorby plánu testů je nutné provést další rozdělení odpovědností, tentokrát již ale konkrétním pracovníkům v rámci jednotlivých skupin. Každý člen projektového týmu musí znát svoji odpovědnost za realizaci konkrétních úkolů v rámci dané skupiny podílející se na projektu. Příslušný pracovník musí mít všechny dostupné informace o jemu přidělených úkolech tak, aby je mohl co nejlépe splnit. Například v případě testování musí mít člen testovacího týmu k dispozici kompletní dokumentaci komponenty, za jejíž testování je zodpovědný.

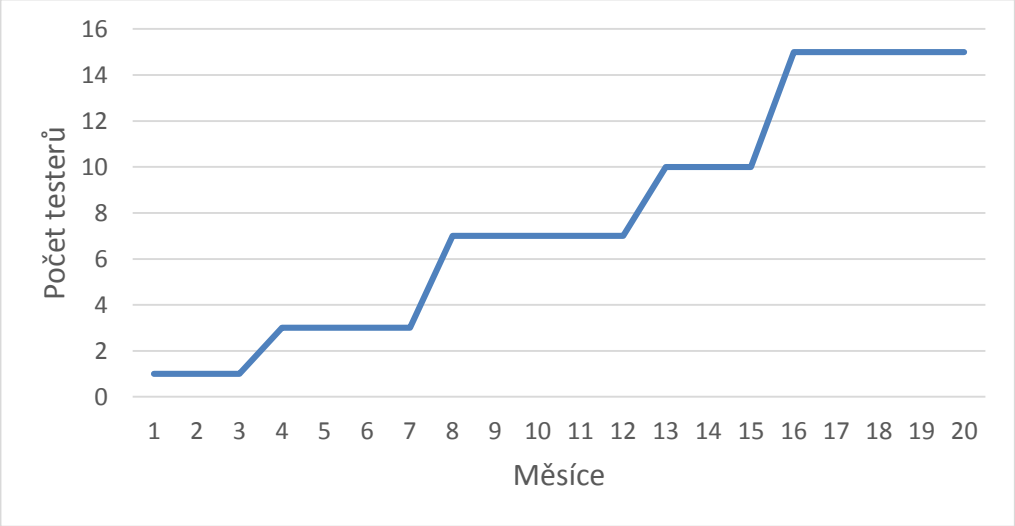
Na závěr této fáze by měla být provedena kontrola, že neexistuje oblast testování, za kterou není nikdo zodpovědný. (WATKINS, 2001)

3.6.10 Časový harmonogram

V tomto okamžiku je zapotřebí všechny doposud zjištěné informace sestavit a promítnout do celkového plánu. Jedná se o klíčovou fázi z hlediska plánování testů, protože se může v průběhu testování stát, že se vyskytne problém, jehož řešení zabere velké množství času, přičemž při odhadech a tvorbě plánu se tato oblast zdála být zcela jasná a bezproblémová. V případě, že je v rámci plánování testů sestaven také jejich časový rozvrh, jsou tímto krokem zcela jasně vymezeny činnosti testovacího týmu a jsou tím také poskytnuty informace pro produktový tým i vedoucí projektu. Ti na základě těchto informací, mohou lépe plánovat jednotlivé činnosti v rámci příslušného projektu. Na základě dobře sestaveného plánu testů mohou vedoucí projektu rozhodnout o „vypuštění“ nebo přesunu realizace vybraných funkcí do pozdější verze softwarového produktu. (SPILLNER, 2011)

Dalším podstatným faktorem z hlediska plánování testů je rozložení objemu testovacích prací. Objem práce v rámci testování není rovnoměrně rozložen v průběhu celé

délky realizace projektu. Určité testovací aktivity probíhají již na začátku projektu v podobě například revizí specifikací a kódu. V průběhu projektu dochází ke zvyšování množství úkolů testovacího týmu, zvyšování množství času stráveného nad testováním a ke zvyšování počtu členů testovacího týmu, zapojených do testování. Nejvyšší vytížení testovacího týmu nastává krátce před konečným uvolněním produktu do provozu. Příklad vývoje zapojení jednotlivých členů testovacího týmu do testování je znázorněn na obrázku č. 2. (MYERS, a další, 2012)



Obrázek 2 Vývoj počtu testerů zapojených v průběhu projektu Zdroj: (PATTON, 2002), str. 219

Sestavení časového plánu testů je jedna věc, ovšem dodržet tento plán je věc druhá. Testovací tým nemůže ovlivnit zpoždění, která mohou nastat v počátečních fázích projektu. Pokud se dostane projekt do zpoždění na začátku své realizace, je pro testovací tým velmi obtížné, často i nemožné splnit svůj úkol do stanoveného termínu. Proto se doporučuje v rámci sestavování plánu testu uvést místo absolutně stanovených údajů relativně zadaná data. Například sestavení plánu testů bude trvat 4 týdny a jeho sestavování bude zahájeno 7 dní po dokončení specifikace softwarového produktu. Tímto krokem se manažer testů vyhne velkému tlaku jak na sebe, tak testovací tým a zajistí pro testování dostatek času tak, aby byl splněn cíl testování a byla zajištěna požadovaná úroveň kvality vyvíjeného softwaru. (PATTON, 2002)

3.6.11 Metriky a statistiky

Metriky a statistiky slouží pro měření a sledování vývoje projektu včetně procesu testování.

Pro testování je možné například využít níže uvedené testové metriky:

- počet chyb nalezených v jednotlivých dnech realizace projektu,
- seznam chyb čekajících na opravení,
- seznam chyb rozdělených podle závažnosti,
- počet chyb nalezených v jednotlivých komponentách. (ISTQB, 2013)

3.6.12 Rizika a problémy

Jednou z dalších součástí plánování testů je identifikace potenciálních problémů, které mohou nastat. Je nutné soustředit se zejména na rizika, která by měla dopad na průběh testování. (FALLEY-VINAY, 2008)

Jsou to právě členové testovacího týmu, kteří jsou zodpovědní za identifikaci a oznámení rizik manažerovi testování, který by tato rizika měl řešit s vedoucím projektu. Rizika je nutné nejen zahrnout do plánu testů, ale také je zapotřebí promítnout je do časového rozvrhu jednotlivých testovacích činností. Ne všechna identifikovaná rizika se musí projevit, některá mohou být planým poplachem, jiná se mohou v průběhu testování naplno projevit. Je nezbytné tedy identifikovat všechna potenciální rizika, protože pouze tak je možné vyhnout se nepříjemným překvapením. (BLACK, 2002)

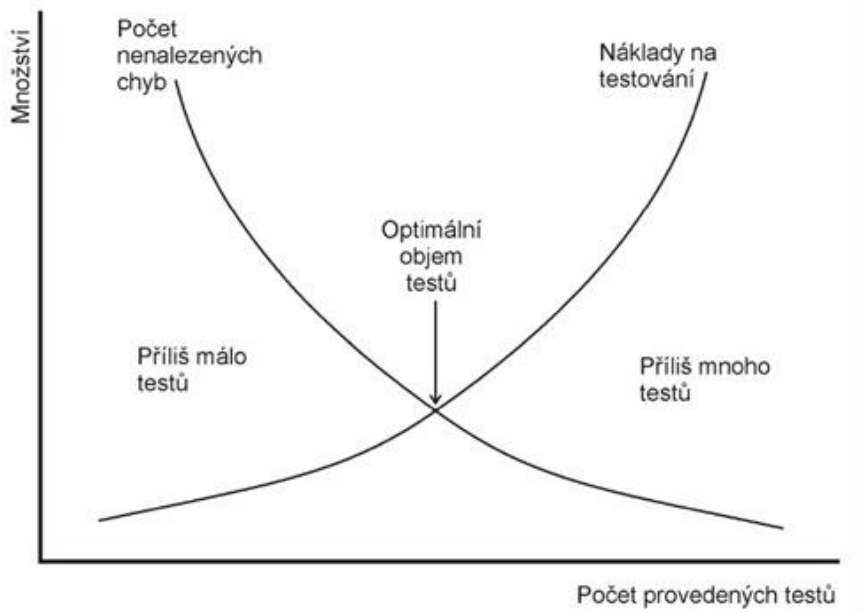
3.7 Návrh testovacích případů

Software nelze nikdy kompletně otestovat (s výjimkou triviálních případů, které v této práci pomineme). Obvykle existuje mnoho vstupů a různých variant, které mohou při běhu softwaru nastat. Pokud neprovedeme kompletní testování, nemůžeme říci, že software neobsahuje chyby. Testování je tedy spojeno s určitou mírou rizika, že otestovaný software může obsahovat chyby, nicméně jak je uvedeno v kapitole 3.3.3.1, vyčerpávající testování je nemožné, a proto je nutné stanovit cíl, kterého chceme v rámci testování dosáhnout. (SPILLNER, 2011)

3.7.1 Optimální objem testovacích případů

Čím větší množství testovacích případů provedeme v rámci testování softwarového produktu, tím bude nižší pravděpodobnost výskytu chyby. S rostoucím počtem prováděných testovacích případů roste nejen časová náročnost testování, ale také dochází ke zvyšování nákladů projektu. Cílem vedení projektu je vytvořit kvalitní software při splnění stanoveného časového a finančního plánu. (PATTON, 2002)

Rozhodování o objemu testovacích případů by mělo vycházet z požadavků na výsledný produkt a současně z rozpočtového a časového plánu. Obrázek č. 3 znázorňuje příklad stanovení optimálního objemu testovacích případů. (FALLEY-VINAY, 2008)



Obrázek 3 Volba optimálního objemu testovacích případů, Zdroj: (MICHÁLEK, 2006)

3.7.2 Techniky návrhu testovacích případů založené na specifikaci

Rozdělení tříd ekvivalence

Vstupy softwaru jsou rozděleny do jednotlivých skupin. Pro každou skupinu vstupů je očekáváno stejné chování a předpokládá se, že jsou pravděpodobně zpracovány stejným způsobem. Třídy ekvivalence mohou být stanoveny jak pro platné, tak pro neplatné hodnoty. Dále mohou být třídy identifikovány pro výstupy, vnitřní hodnoty, časově související hodnoty a pro parametry rozhraní. Rozdělení tříd ekvivalence je možné aplikovat na všech úrovních testování. (STEPHENS, a další, 2011)

Analýza hraničních hodnot

Testy tříd ekvivalence se zabývají identifikací jednotlivých skupin, které jsou následně testovány. Existují však hraniční oblasti (maximální a minimální hodnoty tříd), ve kterých se vyskytují defekty s větší pravděpodobností než v jiných oblastech. (ISTQB, 2013)

Hraniční hodnota pro platnou třídu je platná hraniční hodnota a naopak hranice neplatné třídy je neplatná hraniční hodnota. Testy mohou být navrženy s cílem pokrýt buď obě hodnoty, nebo pouze platnou hodnotu. Pokud jsou testy navrženy s cílem pokrýt obě hodnoty, je v průběhu návrhu testovacích případů vybrán test pro každou hraniční hodnotu.

Tato technika je často považována za rozšíření rozdělení tříd ekvivalence nebo jiných technik návrhu testů černé skříňky. (SPILLNER, 2011)

Testování rozhodovacích tabulek

Rozhodovací tabulky jsou užitečným způsobem pro zachycení požadavků obsahujících logické podmínky. Dále mohou být také využity pro zachycení složitých business pravidel, která mají být v testovaném systému implementovány. Při tvorbě rozhodovací tabulky je nutné nejprve provést analýzu specifikace a identifikovat podmínky a činnosti. Poté je možné sestavit rozhodovací tabulku, přičemž vstupní podmínky nabývají nejčastěji hodnot Pravda nebo Nepravda (hodnoty typu Boolean). (FALLEY-VINAY, 2008)

Výsledná rozhodovací tabulka potom obsahuje spouštěcí podmínky pro všechny vstupní podmínky a výsledné činnosti pro každou kombinaci podmínek. Jednotlivé sloupce odpovídají specifickému pravidlu, které je definováno unikátní kombinací podmínek, která vede k provedení odpovídající činnosti. Obvykle se vytváří na každý sloupec jeden testovací případ, což zajistí pokrytí všech kombinací vstupních podmínek. (PATTON, 2002)

Výhodou testování pomocí rozhodovacích tabulek je vytvoření kombinací podmínek, které by jinak nemusely být otestovány. Technika může být aplikována v případě, kdy je činnost softwaru závislá na několika logických rozhodnutích. (SPILLNER, 2011)

Testování přechodů stavů

Základem testování přechodů stavů je stavový diagram, který umožňuje testerovi pochopit software z hlediska stavů, přechodů mezi stavy, vstupů a událostí, které spouštějí jednotlivé přechody (změny stavů). (WATKINS, 2001)

Stavy objektu nebo systému jsou samostatné, identifikovatelné a je jich konečné množství. Sestavená stavová tabulka potom může pomoci odhalit neplatné přechody. Testy mohou být navrženy s cílem pokrýt nejobvyklejší sekvence stavů, pro každý stav, pro každý přechod, pro vykonání specifických sekvencí přechodů nebo pro testování neplatných přechodů. (BLACK, 2002)

Testování přechodu stavů je používáno v odvětví vestavěných softwarových produktů a v technické automatizaci obecně, nicméně tato technika je taktéž vhodná pro modelování uživatelských objektů, které nabývají specifických stavů. (ISTQB, 2013)

Testování případů užití

Dalším způsobem, pomocí něhož může být software testován, je testování případů užití (use case). Případ užití popisuje interakce mezi uživateli nebo systémy, které vytvářejí výsledek, jež má hodnotu pro uživatele systému nebo zákazníka. Případy užití mohou být popsány například na úrovni obchodního procesu (tedy bez použití technologie) nebo na úrovni systému (systémový případ užití znázorňující konkrétní funkcionalitu systému). Bez ohledu na úroveň, musí každý případ užití splňovat (BLACK, 2002):

- každý příklad užití musí mít předpoklady, které musí být splněny, proto aby fungoval,
- každý případ užití musí být ohraničen výstupními podmínkami a musí mít definován koncový (finální) stav,
- případ užití musí mít základní (nejpravděpodobnější) scénář a alternativní scénáře, kterých může být více. (PATTON, 2002)

Případy užití popisují "procesní toky" v rámci systému. Vycházejí tedy z jeho reálného použití, a proto testovací případy, které jsou odvozeny z případů užití, patří k nejpřínosnějším při odhalování defektů v procesních tocích v průběhu skutečného používání systému. Případy užití jsou často využívány například pro integrační testování

komponent či celých systémů nebo také pro návrh akceptační testů prováděných uživatelem nebo zákazníkem. (SPILLNER, 2011)

3.7.3 Techniky návrhu testovacích případů založené na struktuře

Testování pokrytím 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. (ISTQB, 2013)

Pokrytí příkazů je určené podílem počtu vykonatelných příkazů, které jsou pokryté (navrženými nebo vykonanými) testovacími případy a počtem všech vykonatelných příkazů v testovaném kódu. (FALLEY-VINAY, 2008)

Testování pokrytím rozhodování

Pokrytí rozhodování se zaměřuje na testování výsledků rozhodování, jež byly vykonány sadou testovacích případů. Technika definuje testovací případy s cílem dosažení specifických výsledků rozhodování. Větve rozhodování odpovídají rozhodovacím bodům v kódu programu a zobrazují přenos řízení na jinou oblast v programovém kódu. (ISTQB, 2013)

Pokrytí rozhodování určíme jako podíl všech výsledků dosažených příslušnou sadou testovacích případů, a počtem všech možných výsledků rozhodování obsažených v testovaném kódu programu. (PATTON, 2002)

Testování rozhodování je také označováno jako testování řídicích toků, protože sleduje specifický tok řízení prostřednictvím rozhodovacích bodů. Pokrytí rozhodování „je silnější“ než pokrytí příkazů (100% pokrytí rozhodování garantuje 100% pokrytí příkazů, ale ne opačně). (BLACK, 2002)

3.8 Zaznamenání chyb

Náplní práce testera je nejen plánování a provádění jednotlivých testů, ale také oznamování nalezených chyb. Tato činnosti patří k vůbec nejdůležitějším aktivitám testera v rámci celého procesu testování. Každé oznámení chyby musí být efektivní a splňovat

určitá kritéria. Tato kapitola se bude věnovat ohlašování chyb a jejich následným řešením členy vývojového a testovacího týmu. (ISTQB, 2013)

3.8.1 Oprava chyb

Jak již vyplývá z definice testování, úkolem testovacího týmu není pouze nalézt chyby, ale také zajistit jejich opravu a tím dohlížet na dosažení požadované úrovně kvality. (PATTON, 2002)

Řada chyb, které jsou v softwaru nalezeny, zůstane neopravena. Děje se tomu z mnoha důvodu, jako je například nedostatek času, chyba není programátory nebo vedením projektu považována za důležitou nebo by oprava vyžadovala riskantní zásah do softwaru. Tyto příčiny neopravení nalezených chyb tester nemůže ovlivnit. Často se však stává, že nedojde k opravení chyby, protože chyba není testerem správně zadána a programátor z popisu nepochopí, o jakou chybu se jedná. V tomto případě má tester přímý vliv na to, jestli bude chyba opravena či nikoliv. (BLACK, 2002)

Při ohlašování (zadávání) chyb by měla být dodržena následující pravidla:

- **Chyby je třeba oznamovat co nejdříve** – čím dříve je chyba nalezena, tím bude její odstranění levnější a vývojáři budou mít na její opravu více času. Není tedy možné, aby tester střádal chyby např. celý den a na konci dne zadal (oznámil) všechny najednou. Chyba musí být oznámena neprodleně po jejím nalezení. (ISTQB, 2013)
- **Chyby je nutné účinně popsat** – chybu je nutné popsat takovým způsobem, aby jejímu popisu vývojář rozuměl. Tématem jak efektivně zadat chybu, se bude zabývat kapitola 3.8.2. (PATTON, 2002)
- **Oznámení chyby by mělo být nestranné** – testeři a vývojáři se mohou velmi snadno dostat do konfliktu, stačí pouze pár špatně zvolených slov v zadání chyby. Tester musí být vždy nestranný, neosobní a neutrální. Měl by pečlivě vážit slova, jakými chybu popisuje, tak aby nedocházelo ke sporům nebo ke vzniku antipatií mezi ním a ostatními členy projektového týmu. Pro testera je tedy důležité, aby při zaznamenávání chyby využíval určitou dávku taktu a diplomacie. Zpráva o chybě by měla obsahovat fakta a měla by být směřována na vyvíjený produkt nikoliv na vývojáře. (BLACK, 2002)

- **Sledování řešení chyby** – tester je zodpovědný nejen za nalézání a zadávání chyb, ale také za předání chyby odpovědné osobě, která provede opravu chyby. (ISTQB, 2013)

3.8.2 Jak zaznamenat chyby

Základním předpokladem pro řádné a rychle opravení chyby je její efektivní zaznamenání. Efektivně zadaný popisy chyby by měl být:

- **Minimální** – popis chyby musí obsahovat pouze fakta a informace, které jsou potřebné pro její demonstraci a případné opětovné nasimulování. Je nutné uvést konkrétní posloupnost kroků, kterými byla chyba vyvolána. V případě, že chybu vyvolává více než jedna kombinace vstupů, musí tester uvést všechny tyto kombinace. Vývojář potom může v těchto kombinacích nalézt společný model chování, což mu pomůže v hledání příčiny chyby. (ISTQB, 2013; FALLEY-VINAY, 2008)
- **Jednotlivý** – každý popis chyby by se měl zabývat pouze jednou chybou. Někdy může být pro testera obtížné rozeznat, jestli se jedná např. o dva projevy jedné chyby, nebo o dvě odlišné chyby. Je však nutné tyto věci správně identifikovat, protože pokud tester zadá jednu chybu, ve které se bude skrývat několik dalších chyb, je pravděpodobné, že vývojář opraví první chybu a na ostatní zapomene nebo je přehlédne. Z toho důvodu je důležité správně identifikovat počet nalezených chyb. (ISTQB, 2013)
- **Jasný** – v případě popisu chyby je nutné definovat podmínky, při kterých k chybě došlo. Tester musí chybu popsat ve zcela jasných, stručných a zřetelných krocích, aby bylo patrné, že je chyba obecná a projeví se u uživatele. Může se stát, že se chyby vyskytují nahodile a potom je velmi obtížné je popsat. Identifikace kroků, které vedly k vyvolání příslušné chyby, je však velmi důležitá. Nahodile vyskytující se chyby se může tester pokusit identifikovat např. otestováním hraničních podmínek nebo za pomoci spolupráce s vývojářem. (PATTON, 2002)
- **Reprodukovatelný** – pokud má být chyba opravena je nutné, aby byla popsána opakovatelným způsobem, to znamená, že v popisu chyby musí být uvedeny přesné kroky a hodnoty, tak aby se daný případ mohl přesně zopakovat a chybu bylo možné reprodukovat. (MYERS, a další, 2012)

3.8.3 Priorita a severita chyb

V silách vývojářů není opravit všechny zaznamenané chyby současně, obvykle se nestihnou opravit ani všechny nalezené chyby. Proto je nutné nalezené chyby třídit. Každá chyba má dvě základní vlastnosti a těmi jsou priorita a severita (závažnost). (PATTON, 2002)

Priorita chyby značí, jak je důležité chybu odstranit a za jak dlouho od nalezení musí být chyba odstraněna. Závažnost chyby (severita) označuje riziko vzniklé uživateli v případě, že se chyba vyskytne. (ISTQB, 2013)

Součástí každé firmy vyvíjející software je definice stupňů priority a severity včetně pravidel, podle nichž se vytváří pořadí, v němž jsou chyby opravovány. Kategorie priority a severity podle Patton, 2002:

Priorita

1. Havárie, ztráta či poškození dat.
2. Funkční chyba, nesprávný výsledek.
3. Kosmetické chyby jako např. překlep, špatné rozložení formuláře.
4. Připomínka.

Závažnost (Severita)

1. Okamžitě opravit – chyba znemožňuje další testování.
2. Opravit před uvedením do běžného provozu.
3. Opravit pokud na to zbude čas.
4. Oprava může počkat na další verzi produktu. (PATTON, 2002)

3.9 Hodnocení testování

V průběhu testování softwaru je nutné neustále provádět kontrolu a hodnocení aktuálního stavu projektu. Důvodů, proč provádět kontrolu a hodnocení je celá řada ovšem jedním z nejčastějších důvodů bývá kontrola kapacit testerů a vývojářů podílejících se na testování. Může se stát, že se v rámci testování vyskytlo mnoho chyb a v dané chvíli na projektu nepracuje dostatečný počet testerů a je třeba testovací tým doplnit o dalšího

testera či více testerů. Nastat může také opačná situace, kdy je nutné přesunout testera nebo více testerů na jiný projekt z důvodu nevyužití jejich kapacity. (BLACK, 2002)

K průběžnému hodnocení testování se využívají tři základní metriky.

3.9.1 Metriky založené na chybách

V případě, že chce manažer testování nebo vedoucí projektu znát aktuální stav testování, může vycházet z aktuálního počtu nalezených chyb, nicméně pouze číselný údaj o množství chyb mu nic neřekne a sám o sobě nemá správnou vypovídací hodnotu.

Metriky založené na chybách využívají kromě informace o počtu nalezených chyb také další informace, jako je například závažnost chyby nebo stav její opravy. Tyto informace doplňují údaj o počtu nalezených chyb a pro manažera testování slouží jako sada užitečných měřítek, na jejichž základě je schopen provést odpovídající hodnocení testů. (FALLEY-VINAY, 2008)

3.9.2 Metriky založené na sadě testovacích případů

Metriky založené na testech vychází z předpokladu, že testovaný software projde sadou testovacích případů a neobsahuje žádné chyby (po ukončení testování byly všechny nalezené chyby opraveny) a software je připraven k nasazení do běžného provozu. Tvrzení, že software neobsahuje žádné chyby je však zavádějící, pokud by mělo být tvrzení zcela přesné, tak je nutné říci, že v rámci stanovené sady testovacích případů software neobsahuje žádnou chybu. Nikdy (s výjimkou triviálních případů) není možné otestovat všechny varianty, které mohou v příslušném softwaru nastat. (ISTQB, 2013)

V případě, že známe počet testovacích případů, pomocí nichž bude příslušný software testován, lze snadno určit, v jaké fázi se testování nachází a kolik procent testovacích případů již bylo provedeno, kolik procent čeká na opravení chyb a kolik procent testovacích případů čeká na provedení. (FALLEY-VINAY, 2008)

3.9.3 Metriky založené na kódu

Metriky založené na kódu vyjadřují, kolik procent kódu testovaného softwaru již bylo otestováno a kolik procent ještě zbývá otestovat. Metriky založené na kódu se často označují jako pokrytí kódu (viz kapitola 3.7.3). (BLACK, 2002)

4 Praktická část

V praktické části diplomové práce bude řešen skutečný projekt z oblasti telekomunikací. Přestože se jedná o reálnou funkcionalitu, jsou všechny názvy uvedené v této kapitole vymyšlené a společnost, v rámci níž testování probíhalo, bude anonymní.

4.1 Popis testovaného systému

Systém pro simulaci datového provozu je jedním z nejdůležitějších systémů společnosti působící v oblasti telekomunikací. Úkolem systému je realizovat procesy aktivací, deaktivací a změn roamingových balíčků a následné čerpání datových limitů. Systém je integrován s řadou dalších interních systémů a aplikací, které od něho přebírají, a naopak systému poskytují, informace o uživatelích, tarifech, datových balíčcích a jednotlivých konfiguracích. Systém pro simulaci datového provozu je tzv. backend tedy aplikace, která je v pozadí a s níž nepřijdou do kontaktu koncoví zákazníci ani zaměstnanci na jednotlivých pobočkách. Součástí systému je tzv. produktový katalog, v němž jsou uloženy jednotlivé tarify, balíčky, jejich konfigurace a další nezbytné informace. Další část funckionality systému tvoří sada skriptů, které slouží k simulaci jednotlivých akcí tak, jako by je prováděl skutečný zákazník na provozním prostředí.

Systém pracuje v pozadí za frontend systémy a aplikacemi, od nichž přebírá požadavky, které následně provádí. V praxi jsou akce v rámci popisovaného systému iniciovány buď samotným zákazníkem, který si přes webové rozhraní nebo mobilní aplikaci může zvolit roamingový balíček v rámci svého mobilního tarifu nebo zaměstnancem na pobočce, který provede na přání zákazníka aktivaci jím vybraného roamingového balíčku. Následně jsou tyto hodnoty zapsány do databáze popisovaného systému a odeslány do ostatních systémů. Hodnoty týkající se zákazníka, jeho roamingových balíčků nebo hodnoty datového limitu jsou uchovávány ve zmiňované databázi systému. Pokud se poté zákazník připojí k internetu v zahraničí, je mu měřena hodnota čerpaného objemu dat. Zákazník je vždy informován o vyčerpání datového limitu balíčku a jsou mu nabídnuty další možnosti, mezi nimiž si může vybrat. Zákazník má možnost si buď prostřednictvím tzv. landing page dokoupit balíček, nebo může využívat roaming se sníženou rychlostí, anebo mu může být účtována pevná sazba za 1 MB dat dle jeho tarifu. Konkrétní možnost závisí na tarifu a službách, které má zákazník aktivovány.

4.2 Dokumentace ke změnám testovaného systému

Součástí nové verze produktové katalogu systému pro simulaci datového provozu jsou nové položky v produktovém katalogu. Definice jednotlivých typů a konfigurací nových položek je následující:

- **Tariffs**

V rámci změn byly do produktového katalogu přidány nové tarify. Tabulka č. 2 popisuje nově vzniklé roamingové tarify včetně jejich konfigurací.

ID	Tariff group	Tarif code	Tariff name
200275189297	Postpaid zero rate national	512	Cestovatel 1
200275189299	Postpaid zero rate national	513	Cestovatel 2
10000092	Postpaid zero rate national	DTTP0001	Cestovatel 3

Tabulka 2 Přehled nových tarifů, Zdroj: Vlastní zpracování

- **Tariff Add-ons Mapping Tariffs**

Položky Tariff Add-ons Mapping Tariffs mapují definované tarify na tzv. Ad-ons, jejichž prostřednictvím je k danému tarifu přiřazen odpovídající roamingový balíček.

ID	Service ID	Add-on-id	Description
200275189325	TECH512	empty	Cestovatel 1 addon
200275189327	TECH513	empty	Cestovatel 2 addon
10000096	DTAP0001	1	Cestovatel 3 addon

Tabulka 3 Přehled nových položek Tariff Add-ons Mapping Tariffs, Zdroj: Vlastní zpracování

- **Manage Data Pass Type**

Položky Manage Data Pass Type představují konkrétní datové baličky, které si budou moci zákazníci zakoupit pro využívání internetu v zahraničí, konkrétně v rámci Evropské unie.

ID	Name	Type Code
200275189309	Cestovatel 1, 2 addon	RP1BC1_7
200275184889	Cestovatel 3 addon	RP1BC1_5

Tabulka 4 Nové položky Manage Data Pass Type – část 1, Zdroj: Vlastní zpracování

Validity	Segment	Restriction	Hard Limit	Soft Limit	HL STATE
1	POSTPAID	ALLOWANCE	104857600	80%	FALLBACK
1	POSTPAID	ALLOWANCE	104857600	80%	FALLBACK

Tabulka 5 Nové položky Manage Data Pass Type - část 2, Zdroj: Vlastní zpracování

4.3 Test plán

Kapitola se bude zabývat návrhem testovacího plánu v rámci testování systému pro simulaci datového provozu.

4.3.1 Vymezení produktu, stanovení cílů a předmětu testování

V rámci tohoto projektu bude testován systém pro simulaci datového provozu, u něhož došlo k nasazení nové verze produktového katalogu, který obsahuje nové položky. Z tohoto důvodu bude nutné provést systémové testy za účelem ověření správné funkcionality nově implementovaných položek. Cílem testování je kompletní otestování aktivačních procesů nově implementovaných položek a následná simulace datového provozu, při němž by mělo dojít k vyčerpání datového balíčku.

4.3.2 Lidé, místa a věci

Tým	Členové týmu	Alokace
Projektový manažer	Pavel Nový	0,25 úvazku
Analytický tým	Petr Kvapil (Hlavní analytik) Jan Zelený (Analytik)	Každý 0,2 úvazku
Vývojový tým	Václav Malina (Hlavní programátor) Ondřej Malý (Programátor)	Každý 0,2 úvazku
Testovací tým	Lukáš Hrnčíř	Plný úvazek

Tabulka 6 Přehled skupin a členů projektového týmu, Zdroj: Vlastní zpracování

Všechny potřebné dokumenty k projektu budou dostupné na adrese: intranet.cz/R012016/SPSDP[1]

4.3.3 Definice názvosloví

- Defekt – nesoulad s dokumentací nebo chyba nalezená při provádění navržených testovacích případů.
- Report – dokument, který je vyplňován na denní bázi všemi aktivně se na projektu podílejícími členy týmu.
- Change request – opravná verze, kterou vytváří programátor za účelem odstranění nalezené chyby.
- Status – projektová schůzka, na které se řeší aktuální problémy či nejasnosti při realizaci projektu
- Retest – opětovné provedení testu po nasazení opravy programátorem.

[1] Jedná se o fiktivní adresu pro ukázku. V praxi by se jednalo o odkaz na interní web společnosti, kde by byly umístěny dokumenty k danému projektu.

4.3.4 Definice odpovědností a povinností v rámci skupin

Zodpovědný tým	Úkoly
Projektový manažer	Řízení projektu, zajišťování prostředků pro potřeby projektu, komunikace s ostatními členy projektového týmu.
Analytický tým	Návrh řešení, tvorba technické dokumentace, komunikace s ostatními členy projektového týmu.
Vývojový tým	Implementace navrženého řešení, oprava nalezených chyb, komunikace s ostatními členy projektového týmu.
Testovací tým	Vytvoření test plánu, sady testovacích případů, provedení testů, reporting nalezených chyb, opětovné testování opravených chyb, komunikace s ostatními členy projektového týmu.

Tabulka 7 Rozdělení odpovědností dle skupin v projektovém týmu, Zdroj: Vlastní zpracování

4.3.5 Fáze testování

Pro otestování nově implementovaných změn systému pro simulaci datového provozu bylo zvoleno systémové testování.

4.3.6 Strategie testování

Testování bude prováděno členem interního testovacího týmu. Na celý systém pro simulaci datového provozu bude aplikováno black-box testování. Celý proces testování bude prováděn manuálně a testovací případy budou navrženy za pomoci techniky Testování přechodů stavů.

4.3.7 Požadavky na prostředky

Pro testování je nutné mít k dispozici jednoho testera s plnou alokací, který má zkušenosti s testováním systému pro simulaci datového provozu. Dále je nutné mít plně funkční a dostupné testovací prostředí s nasazenými změnami v rámci testovaného systému.

4.3.8 Časový harmonogram

Harmonogram projektu	
Tvorba technické dokumentace	02. 02. 2016 - 05. 02. 2016
Implementace	08. 02. 2016 - 14. 02. 2016
Test analýza	08. 02. 2016 - 14. 02. 2016
Instalace na testovací prostředí	15. 02. 2016 – 16. 02. 2016
Testování	17. 02. 2016 - 29. 02. 2016
Instalace na produkční prostředí	10. 03. 2016

Tabulka 8 Harmonogram projektu, Zdroj: Vlastní zpracování

4.3.9 Rizika a problémy

Ze strany testera byla identifikována následující možná rizika:

- nedodržení termínu předání k testování,
- chybné nasazení na testovací prostředí,
- chybná konfigurace testovacího prostředí,
- neaktuální dokumentace.

4.4 Test analýza

Testování bude rozděleno do dvou fází. Úkolem první fáze bude ověřit konfiguraci nových položek produktového katalogu. Pro otestování konfigurací bude vytvořena sada testovacích případů označených jako ECH (Entry Check). Tato sada testů ověřuje, zda došlo k nasazení příslušných změn i ke konfiguraci nově implementovaných položek. Tyto testovací případy se provádějí přednostně, protože představují prerekvizity pro provedení zbylých testů. Pokud by v rámci testování systému pro simulaci datového provozu nebyly přidány všechny nové položky do produktového katalogu, nebylo by možné provést operaci jako je například aktivace zákazníka, aktivace balíčku příslušnému zákazníkovi nebo simulace datového provozu na příslušném zákazníkovi s aktivním roamingovým balíčkem. Proto je nutné nejprve provést všechny testovací případy označené jako ECH a vyřešit případné chyby, které byly při testování nalezeny.

Ve druhé fázi bude provedena simulace datového provozu tak, aby bylo zajištěno, že jsou všechny nové položky správně nastaveny, jsou plně funkční a je možné přistoupit k nasazení na produkční prostředí. Předmětem druhé fáze testování budou testovací případy označované jako ST (systémové testy). V rámci této sady testovacích případů bude

ověřena správná funkce aktivačních procesů, včetně následné změny stavů nově implementovaných datových balíčků při jejich vyčerpání. Dále budou provedeny nezbytné kontroly v lozích a databázi, která uchovává informaci o limitech datových balíčků.

4.4.1 Entry check

Tato kapitola je věnována návrhu konkrétní podoby testovacích případů (neboli konkrétních testů) označovaných jako ECH. Z dokumentace (kapitola 4.2) je patrné, že v rámci nové verze produktového katalogu vznikly 3 nové tarify, 3 nové položky Tariff Addons Mapping Tariffs a 2 nové roamingové balíčky. Je tedy žádoucí, aby byla zkontrolována „přítomnost“ a konfigurace každé z nových položek produktového katalogu.

Může se nabízet otázka, proč nebyla konfigurace položek ověřena v jednom testovacím případu. Pokud by tomu tak bylo, jednalo by se o zásadní chybu Test designera. Při návrhu testovacích případů musí být jednotlivé testovací případy navrženy tak, aby byla jejich následná exekuce efektivně reportovatelná. V případě, že by byla konfigurace všech položek ověřována v jednotlivých krocích jednoho testovacího případu a například u třetí položky by došlo k nalezení chyby, v reportu projektu pro testovací případy typu ECH by existoval pouze jeden testovací případ, který by byl ve stavu failed. Druhou, v tomto případě správnou a žádoucí možností je rozdělit kontrolu jednotlivých položek tak, že při nalezení chyby v konfiguraci jedné položky, bude odpovídající testovací případ ve stavu failed a zbylé testovací případy budou ve stavu passed (za předpokladu, že bude jejich konfigurace správná). Pro následně vytvářený report to znamená zásadní změnu. Místo 0% úspěšně otestovaných ECH bude otestováno 87,5% (7 z 8) testovacích případů typu ECH. Z uvedeného textu vyplývá, že je zapotřebí, aby Test designer zvolil správný detail testovacích případů tak, aby každodenní reporty odrážely skutečný progres, jehož bylo dosaženo v rámci tetování.

Níže uvedené testovací případy kontrolují v prvním kroku, jestli je nová položka v produktovém katalogu skutečně implementována a ve druhém kroku je provedena kontrola konkrétní konfigurace dané položky. Pro každou novou položku byl vytvořen jeden testovací případ, který ověřuje výše uvedené skutečnosti.

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola tarifu ID 200275189297	ECH	1	1	Kontrola, jestli produktový katalog obsahuje tarif ID 200275189297	Tarif byl nalezen v produktovém katalogu
			2	Kontrola konfigurace tarifu dle zadání	Tarif má správnou konfiguraci dle zadání

Tabulka 9 Testovací případ entry check č. 1, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola tarifu ID 200275189299	ECH	1	1	Kontrola, jestli produktový katalog obsahuje tarif ID 200275189299	Tarif byl nalezen v produktovém katalogu
			2	Kontrola konfigurace tarifu dle zadání	Tarif má správnou konfiguraci dle zadání

Tabulka 10 Testovací případ entry check č. 2, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola tarifu ID 10000092	ECH	1	1	Kontrola, jestli produktový katalog obsahuje tarif ID 10000092	Tarif byl nalezen v produktovém katalogu
			2	Kontrola konfigurace tarifu dle zadání	Tarif má správnou konfiguraci dle zadání

Tabulka 11 Testovací případ entry check č. 3, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola Tariff Add-ons Mapping Tariffs ID 200275189325	ECH	1	1	Kontrola, jestli produktový katalog obsahuje položku s ID 200275189325	Položka byla nalezena v produktovém katalogu
			2	Kontrola konfigurace položky dle zadání	Položka má správnou konfiguraci dle zadání

Tabulka 12 Testovací případ entry check č. 4, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola Tariff Add-ons Mapping Tariffs ID 200275189327	ECH	1	1	Kontrola, jestli produktový katalog obsahuje položku s ID 200275189327	Položka byla nalezena v produktovém katalogu
			2	Kontrola konfigurace položky dle zadání	Položka má správnou konfiguraci dle zadání

Tabulka 13 Testovací případ entry check č. 5, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola Tariff Add-ons Mapping Tariffs ID 10000096	ECH	1	1	Kontrola, jestli produktový katalog obsahuje položku s ID 10000096	Položka byla nalezena v produktovém katalogu
			2	Kontrola konfigurace položky dle zadání	Položka má správnou konfiguraci dle zadání

Tabulka 14 Testovací případ entry check č. 6, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola Manage Data Pass Types ID 200275184889	ECH	1	1	Kontrola, jestli produktový katalog obsahuje roamingový balíček s ID 200275184889	Položka byla nalezena v produktovém katalogu
			2	Kontrola konfigurace roamingového balíčku	Položka má správnou konfiguraci dle zadání

Tabulka 15 Testovací případ entry check č. 7, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Kontrola Manage Data Pass Types ID 200275189309	ECH	1	1	Kontrola, jestli produktový katalog obsahuje roamingový balíček s ID 200275189309	Položka byla nalezena v produktovém katalogu
			2	Kontrola konfigurace roamingového balíčku	Položka má správnou konfiguraci dle zadání

Tabulka 16 Testovací případ entry check č. 8, Zdroj: Vlastní zpracování

4.4.2 Systémové testy

Tato kapitola je věnována návrhu konkrétní podoby testovacích případů (konkrétních testovacích scénářů) označovaných jako ST. Za povšimnutí stojí, že tyto testovací případy mají prioritu 2, tzn., k jejich exekuci dojde až při otestovaní všech testovacích případů označených prioritou 1 (v tomto případě všech ECH). Z dokumentace (kapitola 4.2) je patrné, že je nutné ověřit správnou funkcionalitu jednotlivých komponent (Tariff, Tariff add-ons Mapping tariffs a Manage Data Pass Type). V rámci test analýzy byla navržena sada testovacích případů, jejichž cílem je ověřit požadovanou funkcionalitu. Testovací sada obsahuje celkem 6 testovacích případů, nicméně tyto testovací případy je možné rozdělit do dvou skupin, podle procesu, na který se příslušné testovací případy zaměřují.

První skupina testovacích případů je zaměřena na proces aktivací, kde je nutné ověřit, že lze zákazníkovi aktivovat příslušný tarif a následně balíček odpovídající tomuto tarifu.

Cílem druhé skupiny testovacích případů je ověřit správnou funkci procesu simulace datového provozu. Z dokumentace vyplývá, že je nutné ověřit správné chování systému při vyčerpání 100% datového limitu (FUP) balíčku.

V následující části kapitoly je uvedena konkrétní podoba jednotlivých testovacích případů vycházejících z předchozího textu uvedeného v této kapitole.

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Aktivace zákazníka s tarifem ID 200275189297, datovým balíčkem ID 200275189309 a Tariff Add-ons Mapping Tariffs ID 200275189325	SIT	2	1	Aktivace tarifu ID 200275189297 vybranému zákazníkovi	Tarif je aktivní
			2	Aktivace datového balíčku ID 200275189309 vybranému zákazníkovi	Balíček je aktivní
			3	Přiřazení správného Tariff Add-ons Mapping Tariffs ID 200275189325	Add-on je správný
			4	Kontrola v databázi	V databázi se nachází správné údaje k danému zákazníkovi

Tabulka 17 Systémový testovací případ č. 1, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
100% vyčerpání datového limitu (FUP) - zákazník s tarifem ID 200275189297 a balíčkem ID 200275189309	SIT	2	1	100% vyčerpání datového limitu (FUP) balíčku ID 200275189309	100% FUP vyčerpáno
			2	Kontrola v databázi	V databázi došlo k načtení vyčerpaného FUP

Tabulka 18 Systémový testovací případ č. 2, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Aktivace zákazníka s tarifem ID 200275189299, datovým balíčkem ID 200275189309 a Tariff Add-ons Mapping Tariffs ID 200275189327	SIT	2	1	Aktivace tarifu ID 200275189299 vybranému zákazníkovi	Tarif je aktivní
			2	Aktivace datového balíčku ID 200275189309 vybranému zákazníkovi	Balíček je aktivní
			3	Přiřazení správného Tariff Add-ons Mapping Tariffs ID 200275189327	Add-on je správný
			4	Kontrola v databázi	V databázi se nachází správné údaje k danému zákazníkovi

Tabulka 19 Systémový testovací případ č. 3, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
100% vyčerpání datového limitu (FUP) - zákazník s tarifem ID 200275189299 a balíčkem ID 200275189309	SIT	2	1	100% vyčerpání datového limitu (FUP) balíčku ID 200275189309	100% FUP vyčerpáno
			2	Kontrola v databázi	V databázi došlo k načtení vyčerpaného FUP

Tabulka 20 Systémový testovací případ č. 4, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
Aktivace zákazníka s tarifem ID 10000092, datovým balíčkem ID 200275184889 a Tariff Add-ons Mapping Tariffs ID 10000096	SIT	2	1	Aktivace tarifu ID 10000092 vybranému zákazníkovi	Tarif je aktivní
			2	Aktivace datového balíčku ID 200275184889 vybranému zákazníkovi	Balíček je aktivní
			3	Přiřazení správného Tariff Add-ons Mapping 10000096	Add-on je správný
			4	Kontrola v databázi	V databázi se nachází správné údaje k danému zákazníkovi

Tabulka 21 Systémový testovací případ č. 5, Zdroj: Vlastní zpracování

Název testu	Typ	Priorita	Krok	Popis	Očekávaný výsledek
100% vyčerpání datového limitu (FUP) - zákazník s tarifem ID 10000092 a balíčkem ID 200275184889	SIT	2	1	100% vyčerpání datového limitu (FUP) balíčku ID 200275184889	100% FUP vyčerpáno
			2	Kontrola v databázi	V databázi došlo k načtení vyčerpaného FUP

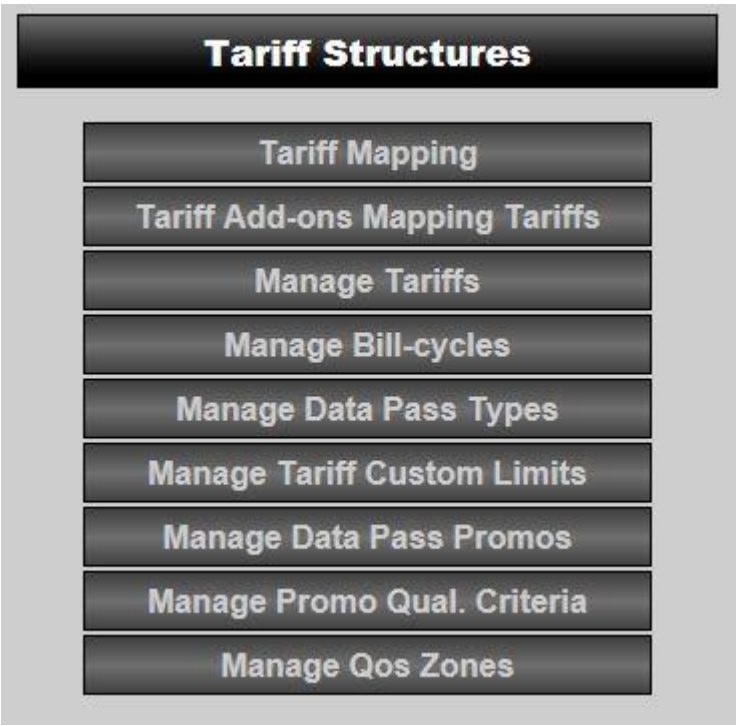
Tabulka 22 Systémový testovací případ č. 6, Zdroj: Vlastní zpracování

4.5 Testování

V této kapitole bude řešena realizace testovacích případů navržených v kapitole 4.4. Jednotlivé testovací případy budou postupně prováděny podle stanovené priority. Po úspěšném provedení daného testovacího scénáře bude příslušnému scénáři přiřazen stav Passed, v případě nalezení defektu, bude scénáři přiřazen stav Failed a bude jednoznačně označen krok testovacího případu, v němž byl defekt nalezen.

4.5.1 Exekuce testovacích případů Entry Check

Na obrázku č. 4 je znázorněna úvodní strana produktového katalogu, ve kterém budou prováděny jednotlivé kontroly konfigurací tarifů, addons a balíčků, které jsou popsány v kapitole 4.2. Na obrázku č. 4 jsou identifikovatelné skupiny, v nichž se budou kontrolované položky nacházet.



Obrázek 4 Hlavní menu produktového katalogu, Zdroj: Vlastní zpracování

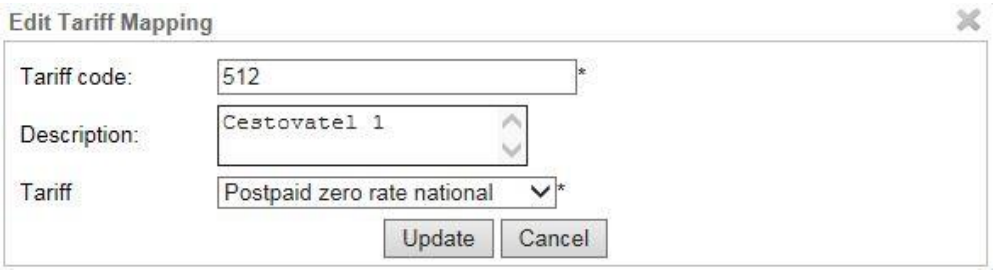
Exekuce entry check testovacího případu č. 1

První testovací případ (tzv. scénář) obsahuje dva kroky. V prvním kroku je nutné ověřit, že seznam tarifů v produktovém katalogu (v menu produktového katalogu položka Tariff Mapping) obsahuje nově zaváděný tarif Cestovatel 1. Vyhledání položky v seznamu

je provedeno dle ID tarifu, které je uvedeno v testovacím případu č. 1 v kapitole 4.4.1. Příslušný scénář byl vytvořen na základě dokumentace z kapitoly 4.2.

Ve druhém kroku scénáře je ověřena konfigurace výše uvedeného tarifu tak, že je zobrazen detail příslušné položky (tarif s ID 200275189297) a jsou porovnávány parametry tarifu z produktového katalogu s hodnotami uvedenými v kapitole 4.2.

Obrázek č. 5 zobrazuje konfiguraci tarifu Cestovatel 1 v produktovém katalogu. Všechny parametry, tedy Tariff code[2], Description[3] i Tariff[4] odpovídají hodnotám uvedeným v kapitole 4.2. Vzhledem k tomu, že jsou hodnoty shodné, je možné přiřadit testovacímu případu č. 1 z kapitoly 4.4.1 stav Passed[5].



Obrázek 5 Konfigurace tarifu Cestovatel 1, Zdroj: Vlastní zpracování

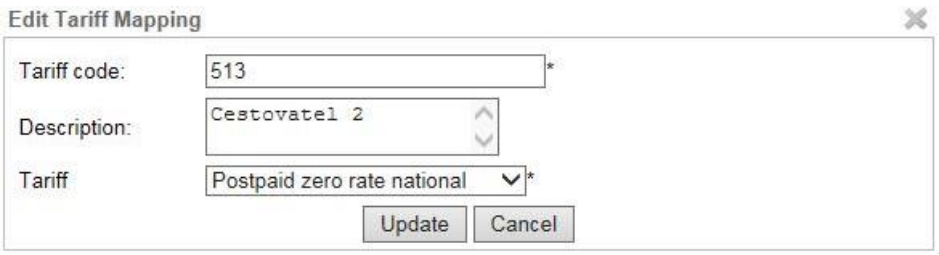
Exekuce entry check testovacího případu č. 2

Druhý testovací scénář obsahuje také dva kroky. V prvním kroku je zapotřebí ověřit, že seznam tarifů v produktovém katalogu (v menu produktového katalogu položka Tariff Mapping) obsahuje druhý z nově zaváděných tarifů (tarif Cestovatel 2). Při vyhledávání položky v produktovém katalogu bylo postupováno stejně jako v případě předchozího scénáře. Scénář byl navržen na základě dokumentace z kapitoly 4.2.

Ve druhém kroku testovacího případu je ověřena konfigurace tarifu tak, že je zobrazen detail příslušné položky (v tomto scénáři tarifu s ID 200275189299) a opět jsou porovnávány parametry tarifu z produktového katalogu s hodnotami uvedenými v kapitole 4.2.

[2] Označuje kód tarifu
[3] Popis, v němž je uveden skutečný název tarifu, pod kterým si ho mohou koupit zákazníci
[4] Interní název tarifu v produktovém katalogu
[5] Stav označuje, že byl příslušný scénář otestován a nebyl nalezen defekt

Obrázek č. 6 zobrazuje konfiguraci tarifu Cestovatel 2 v produktovém katalogu. Vzhledem k tomu, že jsou hodnoty shodné, je možné přiřadit testovacímu případu č. 2 z kapitoly 4.4.1 stav Passed.



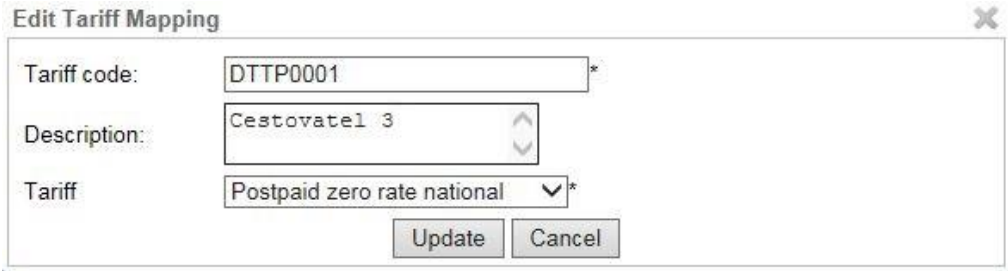
Obrázek 6 Konfigurace tarifu Cestovatel 2, Zdroj: Vlastní zpracování

Exekuce entry check testovacího případu č. 3

Stejně jako tomu bylo i u dvou předchozích scénářů i tento obsahuje dva kroky. V první kroku je opět ověřováno, že seznam tarifů v produktovém katalogu obsahuje třetí z nově zaváděných tarifů (tarif Cestovatel 3). Při vyhledání položky v produktovém katalogu bylo postupováno stejně jako v předchozích dvou testovacích případech. Scénář byl opět navržen na základě dokumentace z kapitoly 4.2.

Ve druhém kroku testovacího případu je ověřována konfigurace tarifu s využitím ID tarifu (záznam je vyhledán pomocí ID 200275189299) a opět jsou porovnávány parametry tarifu z produktového katalogu s hodnotami uvedenými v kapitole 4.2.

Obrázek č. 7 zobrazuje konfiguraci tarifu Cestovatel 3 v produktovém katalogu. Vzhledem k tomu, že jsou hodnoty shodné, je možné přiřadit testovacímu případu č. 3 z kapitoly 4.4.1 stav Passed.



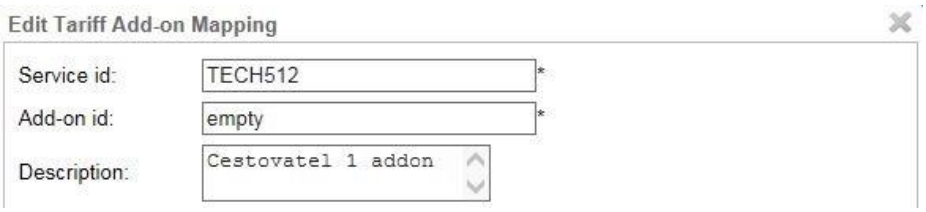
Obrázek 7 Konfigurace tarifu Cestovatel 3, Zdroj: Vlastní zpracování

Exekuce entry check testovacího případu č. 4

Cílem testovacího případu je ověřit správné mapování tarifu na příslušné balíčky pomocí tzv. addons. Scénář obsahuje dva kroky. V prvním kroku je ověřováno, zda se daný záznam nachází v seznamu Tariff Add-ons Mapping Tariffs v produktovém katalogu. Záznam je, stejně jako tarify v předchozích testovacích případech, vyhledán za pomoci ID (v tomto případě ID 200275189325).

Ve druhém kroku testovacího případu je ověřována konfigurace příslušné položky (tzv. addon), přičemž záznam z produktového katalogu je porovnán s hodnotou uvedenou v kapitole 4.2.

Obrázek č. 8 zobrazuje konfiguraci položky s ID 200275189325 v seznamu Tariff Add-ons Mapping Tariffs v produktovém katalogu. Vzhledem k tomu, že jsou hodnoty shodné, je možné přiřadit testovacímu případu č. 4 z kapitoly 4.4.1 stav Passed.



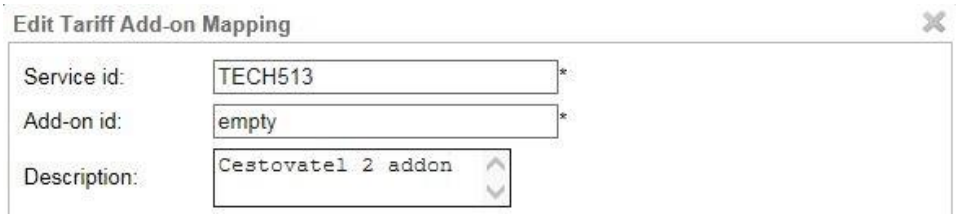
Obrázek 8 Konfigurace položky Cestovatel 1 addon, Zdroj: Vlastní zpracování

Exekuce entry check testovacího případu č. 5

Scénář obsahuje dva kroky, v rámci nichž je ověřováno, jestli se daný záznam nachází v seznamu Tariff Add-ons Mapping Tariffs v produktovém katalogu. Záznam je opět jako v předchozím scénáři vyhledán za pomoci ID (v tomto případě ID 200275189327).

Cílem druhého kroku tohoto testovacího případu je ověření konfigurace příslušné položky (tzv. addon), kdy je záznam z produktového katalogu porovnán s hodnotou uvedenou v kapitole 4.2.

Obrázek č. 9 zobrazuje konfiguraci položky s ID 200275189327 v seznamu Tariff Add-ons Mapping Tariffs v produktovém katalogu. Hodnoty z produktového katalogu a hodnoty z kapitoly 4.2 jsou shodné. Je tedy možné přiřadit testovacímu případu č. 5 z kapitoly 4.4.1 stav Passed.



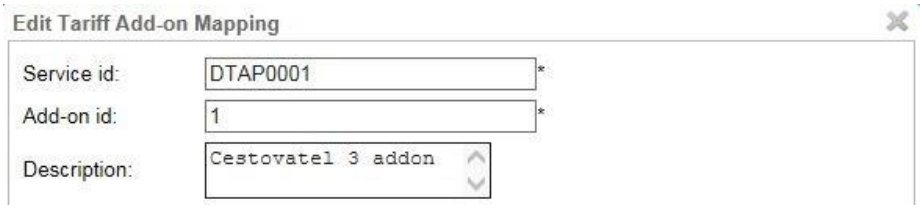
Obrázek 9 Konfigurace položky Cestovatel 2 addon, Zdroj: Vlastní zpracování

Exekuce entry check testovacího případu č. 6

Scénář se opět skládá ze dvou kroků, přičemž v prvním kroku je ověřováno, jestli se daný záznam nachází v seznamu Tariff Add-ons Mapping Tariffs v produktovém katalogu. Záznam je stejně jako v předchozích scénářích vyhledán pomocí ID (v tomto případě ID 10000096).

Ve druhém kroku testovacího případu je ověřována konfigurace příslušné položky (tzv. addon), přičemž záznam z produktového katalogu je porovnán s hodnotou uvedenou v kapitole 4. 2.

Obrázek č. 10 zobrazuje konfiguraci položky s ID 10000096 v seznamu Tariff Add-ons Mapping Tariffs v produktovém katalogu. Vzhledem k tomu, že jsou hodnoty shodné, je možné přiřadit testovacímu případu č. 4 z kapitoly 4.4.1 stav Passed.



Obrázek 10 Konfigurace položky Cestovatel 3 addon, Zdroj: Vlastní zpracování

Exekuce entry check testovacího případu č. 7

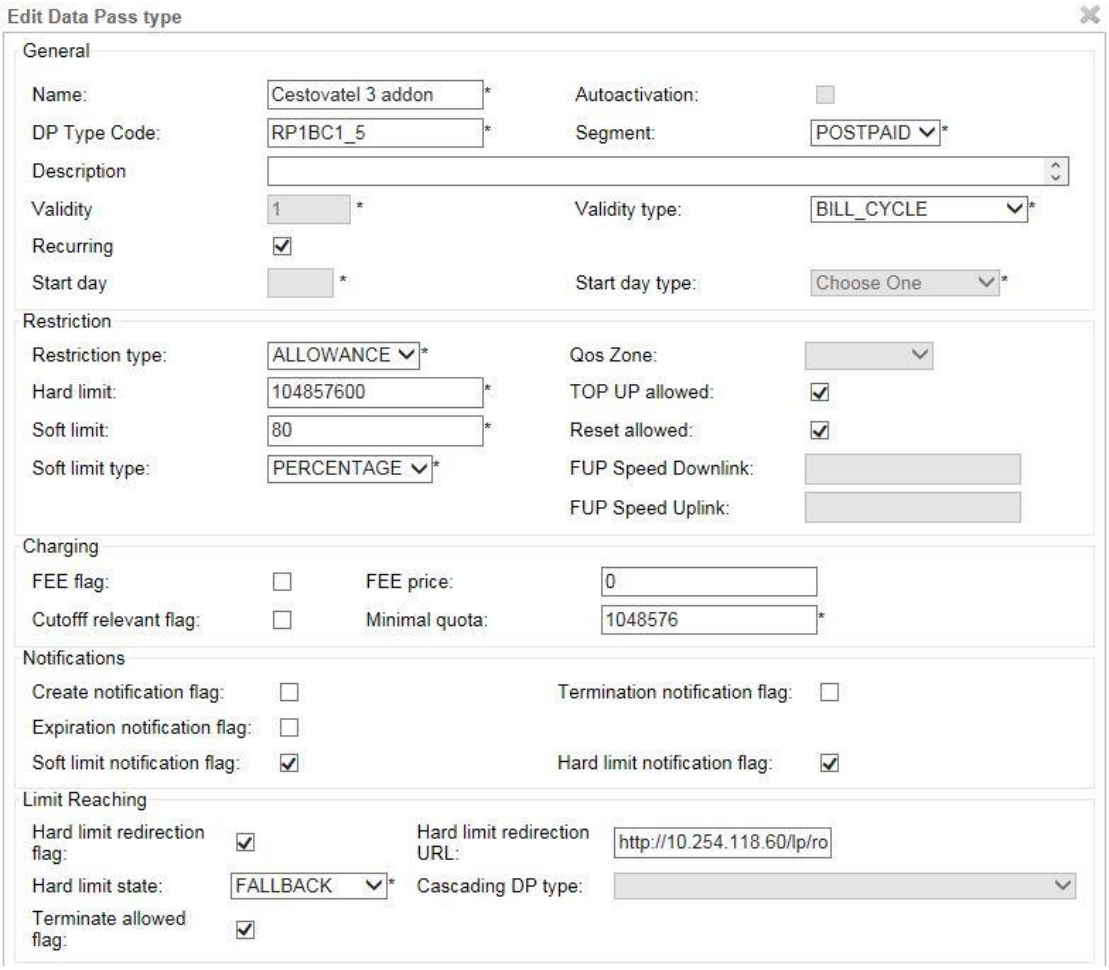
Položky Manage Data Pass Type představují konkrétní balíčky roamingu, které si budou moci zákazníci zakoupit pro využívání internetu v zahraničí.

Testovací případ obsahuje dva kroky, přičemž jeho cílem je ověřit konfiguraci příslušného roamingového balíčku. V prvním kroku je nutné ověřit, že seznam datových balíčků v produktovém katalogu (v menu produktového katalogu položka Manage Data

Pass Types) obsahuje nový roamingový balíček Cestovatel 3 addon. Vyhledání položky v seznamu je provedeno dle ID datového balíčku (ID 200275184889).

Ve druhém kroku testovacího případu je ověřována konfigurace příslušného datového balíčku, přičemž je nutné provést kontrolu následujících parametrů: Name, Type Code, Validity, Segment, Hard limit, Soft limit a HL State. Hodnoty jednotlivých parametrů v produktovém katalogu je nutné porovnat s hodnotami uvedenými v kapitole 4.2.

Obrázek č. 11 zachycuje konfiguraci datového balíčku s ID 200275184889. Je patrné, že jednotlivé hodnoty parametrů v produktovém katalogu odpovídají hodnotám uvedeným v kapitole 4. 2, a proto je možné přiřadit tomuto scénáři také stav Passed.



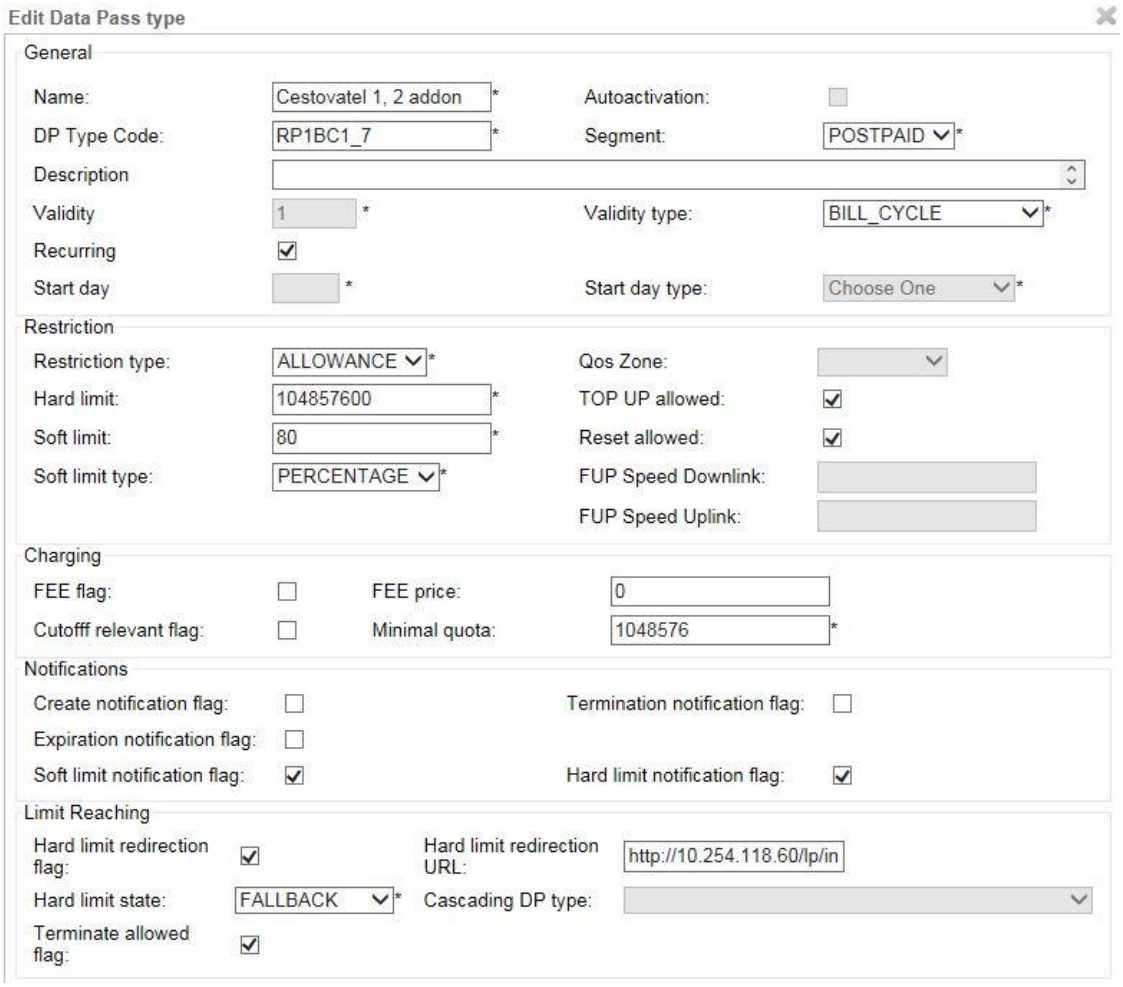
Obrázek 11 Konfigurace datového balíčku Cestovatel 3 addon, Zdroj: Vlastní zpracování

Exekuce entry check testovacího případu č. 8

Scénář se skládá ze dvou kroků, přičemž jeho cílem je ověřit konfiguraci příslušného roamingového balíčku. V prvním kroku je ověřováno, že seznam datových balíčků v produktovém katalogu (v menu produktového katalogu položka Manage Data Pass Types) obsahuje nový roamingový balíček Cestovatel 1, 2 addon. Vyhledání položky v seznamu je provedeno dle ID datového balíčku (ID 200275189309).

Ve druhém kroku testovacího případu je stejným způsobem jako v přechozím scénáři ověřována konfigurace příslušného datového balíčku, přičemž je nutné provést kontrolu hodnot stejných parametrů jako v předcházejícím testovacím případě. Hodnoty jednotlivých parametrů v produktovém katalogu je nutné opět porovnat s hodnotami uvedenými v kapitole 4.2.

Obrázek č. 12 zachycuje konfiguraci datového balíčku s ID 200275189309. Je patrné, že jednotlivé hodnoty parametrů v produktovém katalogu odpovídají hodnotám uvedeným v kapitole 4.2, a proto je možné přiřadit tomuto scénáři také stav Passed.



Obrázek 12 Konfigurace datového balíčku Cestovatel 1, 2 addon, Zdroj: Vlastní zpracování

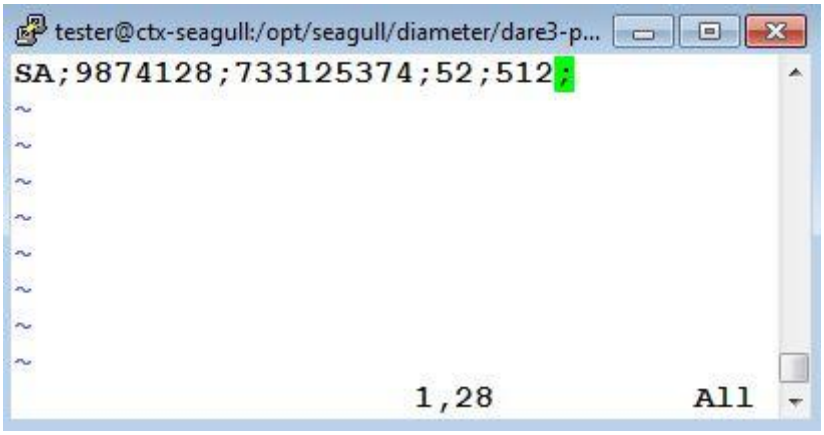
4.5.2 Exekuce systémových testovacích případů

Exekuce systémového testovacího případu č. 1

Cílem tohoto testovacího scénáře je otestovat dva základní procesy aktivací, a to založení zákazníka. V tomto testovacím scénáři se bude jednat o zákazníka s MSIDDN 733125374 a aktivaci tarifu. Zákazníkovi bude aktivován tarif Cestovatel 1 (ID 200275189297).

V prvním kroku je nutné provést založení zákazníka. Tento krok je realizován prostřednictvím souboru a tzv. aktivačního skriptu. (simulační skript pro testovací prostředí). Soubor je určený jako vstup pro aktivační skript a musí mít následující strukturu: SA;extIdSu;MSISDN;billCycle;tariffID;. Zkratka SA označuje operaci aktivace

zákazníka (Subscriber Activation), parametr extIDSu představuje jednoznačný interní identifikátor zákazníka, parametr MSISDN označuje mobilní telefonní číslo, které patří zákazníkovi s příslušným extIdSu. Parametry billCycle a tariffID se vztahují ke konfiguraci příslušného mobilního tarifu, který je zákazníkovi aktivován. Obrázek č. 13 znázorňuje vstupní soubor, který bude využit pro realizaci prvního kroku tohoto testovacího případu.



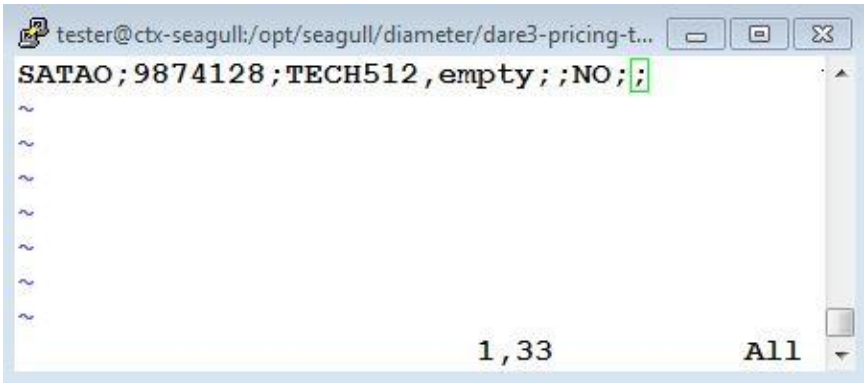
Obrázek 13 Soubor pro aktivaci tarifu Cestovatel 1, Zdroj: Vlastní zpracování

Následný aktivační skript zpracuje výše uvedený soubor s popisovanými parametry. Výsledkem tohoto procesu bude zákazník, který bude mít aktivován příslušný tarif. Na obrázku č. 14 je zvýrazněna část logu, která obsahuje request s parametry, které byly uvedeny do vstupního souboru. Dále log obsahuje response, ve které se nachází velmi důležité pole status = OK, které je na obrázku zvýrazněno zeleným rámečkem. Podle hodnoty OK parametru status je možné identifikovat, že proces aktivace proběhl správně. Za polem status následují parametry, které jsou v response obsaženy, a které se shodují s parametry uvedenými ve vstupním souboru.

```
2016-03-01 17:03:22,433 INFO  c.h.d.p.PricingToolProcessor - Invoking 4 sender(s).
2016-03-01 17:03:22,435 DEBUG com.hp.dare.pricingtool.Sender - processing: csv='SA;98
74128;733125374;52;512;'
2016-03-01 17:03:22,436 DEBUG com.hp.dare.pricingtool.Sender - subscriberActivate req
uest: extIdSu=9874128, msisdn=733125374, billCycle=52, tariffCode=512
2016-03-01 17:03:22,501 DEBUG com.hp.dare.pricingtool.Sender - subscriberActivate res
ponse: status=OK, code=null, desc=null, extIdSu=9874128, responseTime=65
2016-03-01 17:03:22,502 INFO  c.h.d.p.PricingToolProcessor - Collecting results.
2016-03-01 17:03:22,505 INFO  c.h.d.p.PricingToolProcessor - Statistics: Total reques
ts: 1, Total time [s] 0, Request per second: 15, Min/Max/Avg responseTime [ms]: 65/65
/65, OK/FAIL/ERROR count: 1/0/0, Exceptions caught count: 0
2016-03-01 17:03:22,505 INFO  c.h.d.p.PricingToolProcessor - Statistics: LEGEND|Input
File;TotalRequest;TotalTime;RequestRate;ResponseTimeMin;ResponseTimeMax;ResponseTimeA
vg;ResulCodeOK;ResulCodeFAIL;ResulCodeERROR;
2016-03-01 17:03:22,505 INFO  c.h.d.p.PricingToolProcessor - Statistics: DATA|/opt/se
agull/diameter/dare3-pricing-tool-test/provisioning/data/prov_SA.csv;1;0;15;65;65;65;
1;0;0;
2016-03-01 17:03:22,506 INFO  c.h.d.p.PricingToolApplication - Done
```

Obrázek 14 Log úspěšné aktivace tarifu Cestovatel 1, Zdroj: Vlastní zpracování

Ve druhém kroku tohoto testovacího scénáře je nutné provést aktivaci roamingového balíčku (ID 200275189309) příslušnému zákazníkovi. Nejprve je nutné připravit samotný soubor pro aktivaci balíčku danému zákazníkovi a po té bude soubor zpracován aktivačním skriptem. Soubor pro aktivaci roamingového balíčku musí mít následující strukturu: SATAO; extIdSu;tariffAddOn;;. Zkratka SATAO označuje proces aktivace balíčku příslušnému zákazníkovi (Subscriber Activate Tariff Add-on). Parametr extIdSu označuje, stejně jako tomu bylo v předchozím případě, jednoznačný interní identifikátor zákazníka. Dále soubor obsahuje parametr tariffAddOn, který se skládá ze dvou parametrů, konkrétně ze serviceId a addOnId. Hodnoty obou těchto parametrů jsou uvedeny v produktovém katalogu. Obrázek č. 15 znázorňuje soubor pro aktivaci roamingového balíčku.



Obrázek 15 Soubor pro aktivaci roamingového balíčku Cestovatel 1, 2 addon, Zdroj: Vlastní zpracování

Aktivační skript zpracuje výše uvedený soubor (soubor SATAO) s popisovanými parametry. Výsledkem tohoto procesu bude aktivaci balíčku zákazníkovi s příslušným tarifem. Na obrázku č. 16 je zvýrazněna část logu, která obsahuje request s parametry, které byly uvedeny do vstupního souboru. Dále log obsahuje response, ve které se nachází zmiňovaný parametr status = OK, který je na obrázku zvýrazněn zeleným rámečkem. Podle hodnoty OK parametru status je možné identifikovat, že proces aktivace proběhl správně. Za polem status následují parametry, které jsou v response obsaženy, a které se shodují s parametry uvedenými ve vstupním souboru.

```
2016-03-01 17:18:43,858 INFO  c.h.d.p.PricingToolProcessor - Invoking 4 sen
(s).
2016-03-01 17:18:43,860 DEBUG com.hp.dare.pricingtool.Sender - processing:
='SATAO;9874128;TECH512,empty;;NO;;'
2016-03-01 17:18:43,862 DEBUG com.hp.dare.pricingtool.Sender - subscriberAc
ateTariffAddOn request: extIdSu=9874128, tariff add-on[serviceId,addOnId]=[
H512,empty], schedulingDateTime=[], forceTerminate=false, instantDataPassPr
=null, instantCustomLimit=null
2016-03-01 17:18:43,932 DEBUG com.hp.dare.pricingtool.Sender - subscriberAc
ateTariffAddOn response: status=OK, code=null, desc=null, extIdSu=9874128,
ponseTime=69
2016-03-01 17:18:43,933 INFO  c.h.d.p.PricingToolProcessor - Collecting res
s.
2016-03-01 17:18:43,936 INFO  c.h.d.p.PricingToolProcessor - Statistics: To
 requests: 1, Total time [s] 0, Request per second: 13, Min/Max/Avg respons
me [ms]: 70/70/70, OK/FAIL/ERROR count: 1/0/0, Exceptions caught count: 0
2016-03-01 17:18:43,937 INFO  c.h.d.p.PricingToolProcessor - Statistics: LE
D|InputFile;TotalRequest;TotalTime;RequestRate;ResponseTimeMin;ResponseTime
;ResponseTimeAvg;ResulCodeOK;ResulCodeFAIL;ResulCodeERROR;
2016-03-01 17:18:43,937 INFO  c.h.d.p.PricingToolProcessor - Statistics: DA
/opt/seagull/diameter/dare3-pricing-tool-test/provisioning/data/prov_SATAO.
;1;0;13;70;70;70;1;0;0;
2016-03-01 17:18:43,937 INFO  c.h.d.p.PricingToolApplication - Done
```

Obrázek 16 Log úspěšné aktivace roamingového balíčku Cestovatel 1, 2 addon, Zdroj: Vlastní zpracování

Na obrázku č. 17 je znázorněn výpis zákazníka s MSISDN 733125374, jemuž byl v přechozích krocích aktivován tarif ID 200275189297 a následně roamingový balíček ID 200275189309.

První zvýrazněná oblast obsahuje telefonní číslo zákazníka (MSISDN), jednoznačný interní identifikátor zákazníka (extIdSu) a parametr status, který nabývá hodnoty active, tzn., že zákazníkovi je přiřazen aktivní stav. Hodnoty všech tří zmiňovaných parametrů lze označit za správné, protože hodnoty prvních dvou parametrů se shodují s hodnotami z přechozích kroků tohoto testovacího případu. Hodnotu parametru

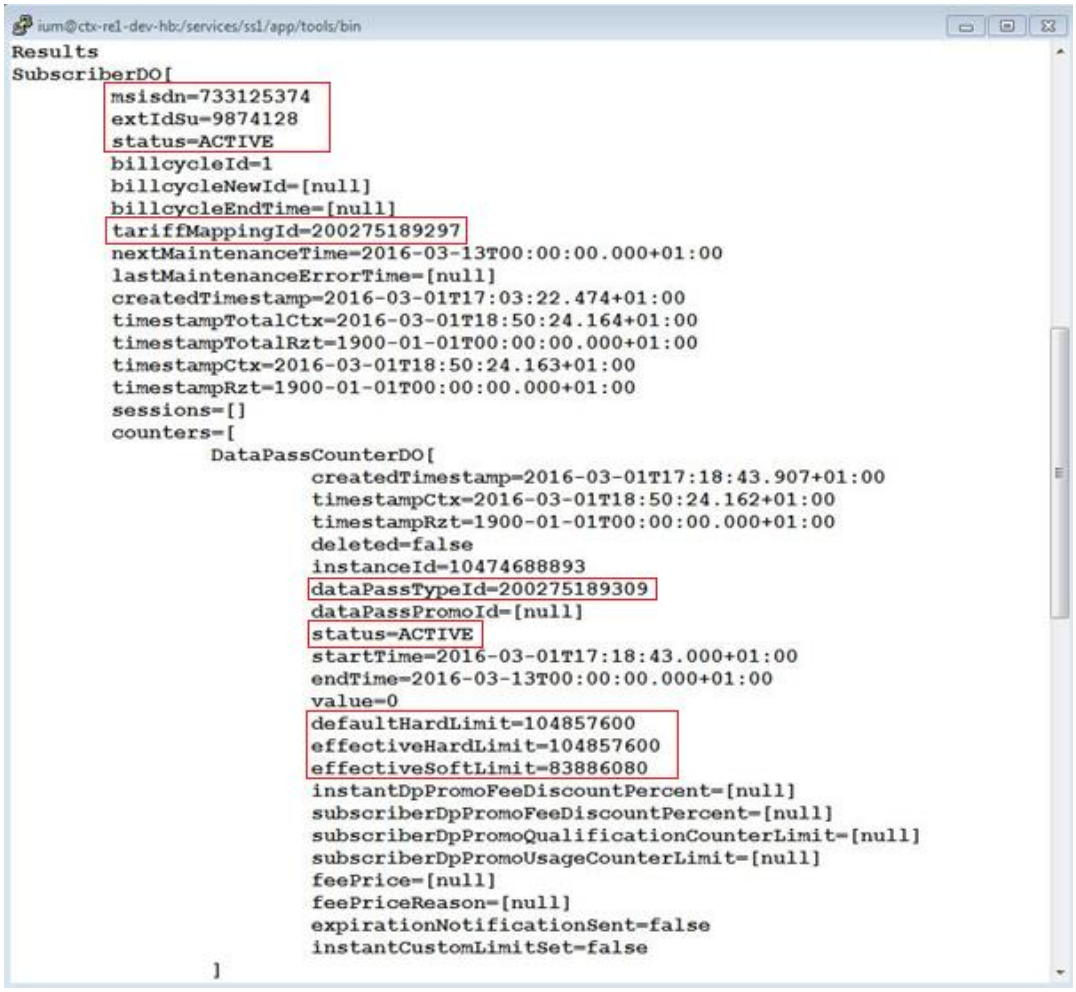
status lze také označit za správnou, protože zákazník byl v prvním kroku založen s požadovaným tarifem, a proto musí mít status aktivní.

Druhá zvýrazněná oblast obsahuje parametr tariffMappingId, jehož hodnota je hodnota ID tarifu, který má zákazník aktivován (v tomto případě je ID tarifu 200275189297). Z obrázku je patrné, že se jedná o správný ID tarif, protože tento tarif byl danému zákazníkovi aktivován v prvním kroku tohoto testovacího případu.

Sekce DataPassCounterDo obsahuje DataPassTypeId (v tomto případe je ID datového balíčku 200275189309), které souhlasí s aktivovaným balíčkem. Dále obsahuje parametr status, který nabývá správně hodnoty aktivní.

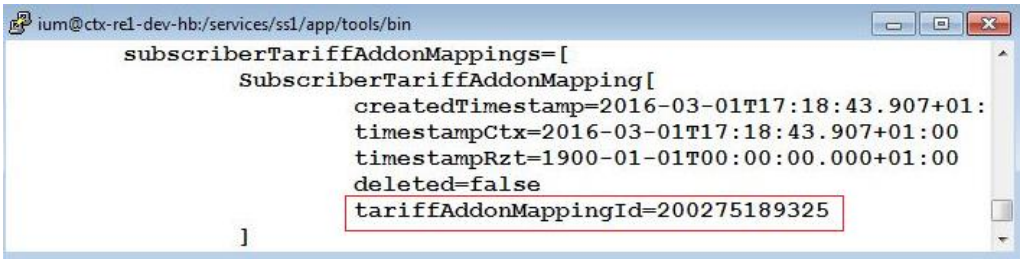
V následující zvýrazněné sekci se nachází parametr defaultHardLimit, který označuje datový limit příslušného balíčku v bajtech (jedná se o 100 MB datový balíček). Posledním důležitým parametrem je parametr efectiveSoftLimit, který označuje 80% vyčerpaného limitu (v tomto případě 80 MB).

Na základě údajů z kapitoly 4.2 a výsledků testovacích případů typu ECH, je možné označit všechny hodnoty parametrů na níže uvedeném obrázku za správné.



Obrázek 17 Výpis údajů k zákazníkovi s MSISDN 733125374 před datovým provozem, Zdroj: Vlastní zpracování

Dále je nutné provést kontrolu správné hodnoty parametru tariffAddonMapingId. Jak je patrné z obrázku č. 18, hodnota parametru tariffAddonMapingId odpovídá příslušné hodnotě ID uvedené v testovacím případu č. 1 v kapitole 4.4.2.



Obrázek 18 Výpis mapovací položky ID 200275189325, Zdroj: Vlastní zpracování

Na základě všech výše uvedených kroků, které byly provedeny za účelem exekuce prvního systémového testovacího případu, je množné testovacímu případu přiřadit stav Passed.

Exekuce systémového testovacího případu č. 2

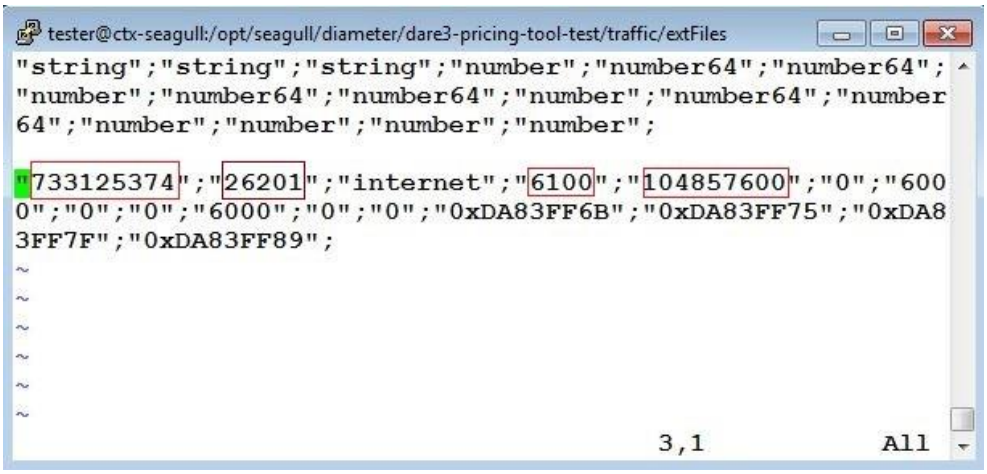
Cílem tohoto testovacího scénáře je otestovat vyčerpání datového limitu (tzv. FUP). Datový provoz bude generován na zákazníkovi s MSIDDN 733125374, který má tarif cestovatel 1 (ID 200275189297) a datový balíček Cestovatel 1, 2 addon (ID 200275189309).

První krok, tedy vyčerpání datového limitu, se skládá z několika dílčích kroků. Obrázek č. 19 znázorňuje strukturu souboru, jehož prostřednictvím je vytvořena tzv. datovací věta. Na obrázku je opět zvýrazněno několik parametrů, které jsou důležité pro čerpání datového limitu. Prvním parametrem je MSISDN (telefonní číslo) s hodnotou 733125374. Mezi další parametry patří network ID usage type, usage in MB a output. Parametr network ID definuje zónu, ve které je datový provoz simulován (v tomto případě hodnota 26201 odpovídá zemím EU) Parametr usage type udává zónu, v níž je datový provoz generován, v tomto případě se jedná o hodnotu 6100 roaming charged, která odpovídá datovému provozu v zahraničí (roamingový datový provoz). Parametr usage in MB definuje velikost datového provozu v MB (v datovací větě je tato hodnota přepočítána do bajtů, bude se tedy jednat o hodnotu 104857600). Posledním velmi důležitým parametrem je output, který na základě zadaných hodnot sestavuje tzv. datovací větu.

MSISDN	733125374				
IS TRAFFIC FREE ?	N	Y/ N			
NETWORK ID	26201	23001	23106	Zona 1 - 26201 - Německo	Zona 2 - 22801 - Švýcarsko
USAGE TYPE	6100	1100 local charged	6100 roaming charged	4100 - Mobil CZ free	
USAGE IN MB	100,00	104857600	bytes		
TIME	4.3.2016 13:52:07	4.3.2016 13:52:17	4.3.2016 13:52:27	4.3.2016 13:52:37	
čas num (dny)	42433,577857060200000	42433,577972800900000	42433,578088541700000	42433,578204282400000	
čas num (s) opravený	3666084727	3666084737	3666084747	3666084757	
čas hex	DA840377	DA840381	DA84038B	DA840395	
čas hex	D7B80948	D7B80952	D7B8095C	D7B80966	
čas num (s) opravený	3619338840	3619338850	3619338860	3619338870	
čas num (dny)	41890,495833333300000	41890,495949074100000	41890,496064814800000	41890,496180555600000	
čas	08.09.2014 11:54:00	08.09.2014 11:54:10	08.09.2014 11:54:20	08.09.2014 11:54:30	
OUTPUT	"733125374";"26201";"	nternet";"6100";"104857600";"0";"6000";"0";"0";"6000";"0";"0";"0xDA840377";"0xDA840381";"0x			

Obrázek 19 Soubor pro tvorbu datovací věty - zákazník s MSISDN 733125374, Zdroj: Vlastní zpracování

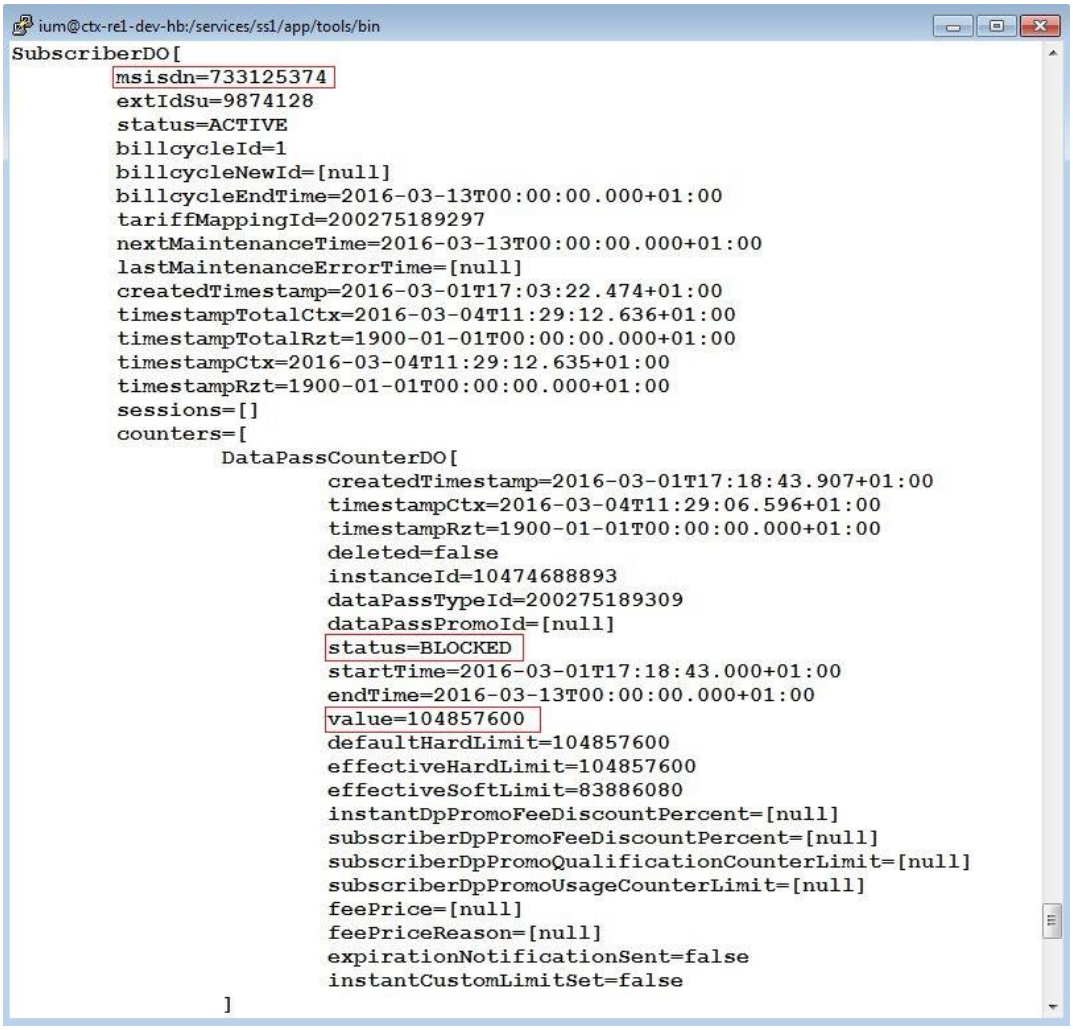
Na základě zadaných hodnot je sestavena datovací věta, která je následně vložena do souboru s připravenou hlavičkou. Výsledkem tohoto dílčího kroku je vytvoření datovacího souboru, jehož strukturu znázorňuje obrázek č. 20. Vytvořený soubor je nutné zkontrolovat. V první označené oblasti se nachází telefonní číslo (MSISDN), následující označená oblast obsahuje lokalitu, v níž dochází ke generování datového provozu. Další označená oblast obsahuje typ generování datového provozu a poslední oblast obsahuje hodnotu datového provozu v bajtech. Všechny hodnoty datovací věty v souboru odpovídají hodnotám, které byly zadány při tvorbě datovací věty.



Obrázek 20 Datovací soubor – zákazník s MSISDN 733125374, Zdroj: Vlastní zpracování

Po vytvoření datovacího souboru je nutné spustit tzv. datovací skript, který tento soubor zpracuje a tím bude provedena simulace datového provozu.

Na závěr je nutné provést kontrolu celého procesu v databázi. Obrázek č. 21 znázorňuje údaje k zákazníkovi s MSISDN 733125374. V sekci DataPassCounterDo se nachází parametr status. Při aktivaci, jejíž realizaci se věnoval předcházející testovací případ, nabýval parametr status hodnoty active. Z níže uvedeného obrázku je patrné, že se hodnota parametru status změnila z hodnoty active na hodnotu blocked. Cílem tohoto testovacího scénáře bylo provést vyčerpání datového limitu balíčku, a proto musí dojít i ke změně hodnoty. Parametr status nabývá hodnoty blocked právě tehdy, když dojde k jeho vyčerpání. Tuto skutečnost dokládá parametr value, jehož hodnota je 104857600 (100 MB). Zmiňovaná hodnota je limitem, při němž má dojít k vyčerpání datového balíčku.



Obrázek 21 Výpis údajů k zákazníkovi s MSISDN 733125374 po datovém provozu, Zdroj: Vlastní zpracování

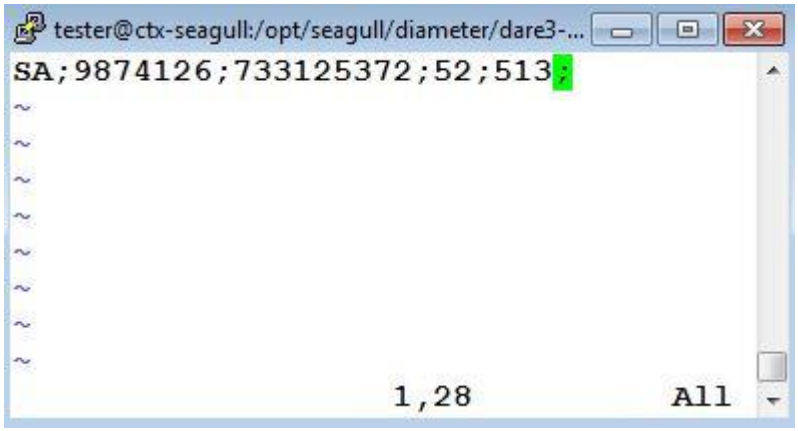
Na základě všech výše uvedených kroků, které byly provedeny za účelem exekuce druhého systémového testovacího případu, je množné testovacímu případu přiřadit stav Passed.

Exekuce systémového testovacího případu č. 3

Cílem tohoto testovacího případu je otestovat procesy aktivací. Bude se jednat o založení zákazníka (v tomto testovacím scénáři se bude jednat o zákazníka s MSIDDN 733125372) a aktivaci tarifu (v tomto případě bude zákazníkovi aktivován tarif cestovatel 2 s ID 200275189299).

V prvním kroku je nutné provést založení zákazníka. Tento krok je opět realizován za pomoci souboru a tzv. aktivačního skriptu. Soubor je vstupem pro aktivační skript a

jeho struktura musí být následující: SA;extIdSu;MSISDN;billCycle;tariffID;. Význam jednotlivých parametrů souboru SA (Subscriber Activation) byl již v této kapitole popsán při exekuci testovacího případu č. 1. Obrázek č. 22 znázorňuje vstupní soubor, který bude využit pro realizaci prvního kroku tohoto testovacího případu.



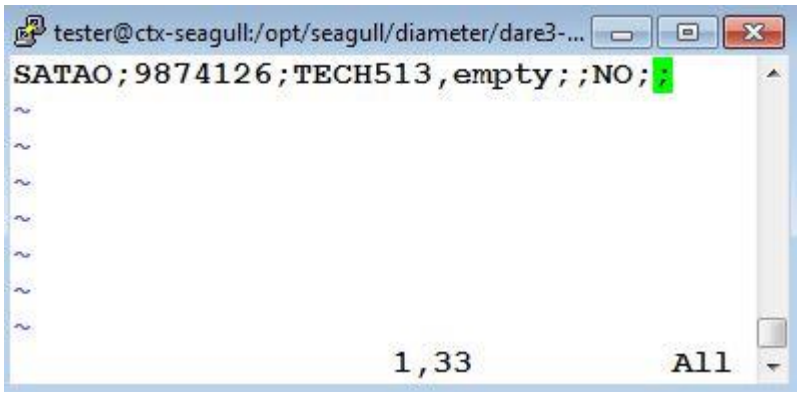
Obrázek 22 Soubor pro aktivaci tarifu Cestovatel 2, Zdroj: Vlastní zpracování

Následný aktivační skript opět provede zpracování výše uvedeného souboru a výsledkem tohoto procesu bude zákazník, který bude mít aktivován tarif s ID 200275189299. Na obrázku č. 23 je zvýrazněna část logu, která obsahuje request s parametry, které byly uvedeny do vstupního souboru. Dále log obsahuje response, ve které se nachází velmi důležité pole status = OK, které je na obrázku zvýrazněno zeleným rámečkem. Podle hodnoty OK parametru status je možné identifikovat, že proces aktivace proběhl správně. Za polem status následují parametry, které jsou v response obsaženy, a které odpovídají parametrům uvedeným ve vstupním souboru, jež byl následně zpracován aktivačním skriptem.

```
2016-03-01 16:50:07,758 INFO  c.h.d.p.PricingToolProcessor - Invoking 4 sender(s).
2016-03-01 16:50:07,760 DEBUG com.hp.dare.pricingtool.Sender - processing: csv='SA;98
74126;733125372;52;513;'
2016-03-01 16:50:07,761 DEBUG com.hp.dare.pricingtool.Sender - subscriberActivate req
uest: extIdSu=9874126, msisdn=733125372, billCycle=52, tariffCode=513
2016-03-01 16:50:07,825 DEBUG com.hp.dare.pricingtool.Sender - subscriberActivate res
ponse: status=OK, code=null, desc=null, extIdSu=9874126, responseTime=63
2016-03-01 16:50:07,825 INFO  c.h.d.p.PricingToolProcessor - Collecting results.
2016-03-01 16:50:07,828 INFO  c.h.d.p.PricingToolProcessor - Statistics: Total reques
ts: 1, Total time [s] 0, Request per second: 15, Min/Max/Avg responseTime [ms]: 64/64
/64, OK/FAIL/ERROR count: 1/0/0, Exceptions caught count: 0
2016-03-01 16:50:07,828 INFO  c.h.d.p.PricingToolProcessor - Statistics: LEGEND|Input
File;TotalRequest;TotalTime;RequestRate;ResponseTimeMin;ResponseTimeMax;ResponseTimeA
vg;ResulCodeOK;ResulCodeFAIL;ResulCodeERROR;
2016-03-01 16:50:07,829 INFO  c.h.d.p.PricingToolProcessor - Statistics: DATA|/opt/se
agull/diameter/dare3-pricing-tool-test/provisioning/data/prov_SA_LH.csv;1;0;15;64;64;
64;1;0;0;
2016-03-01 16:50:07,829 INFO  c.h.d.p.PricingToolApplication - Done
```

Obrázek 23 Log úspěšné aktivace tarifu Cestovatel 2, Zdroj: Vlastní zpracování

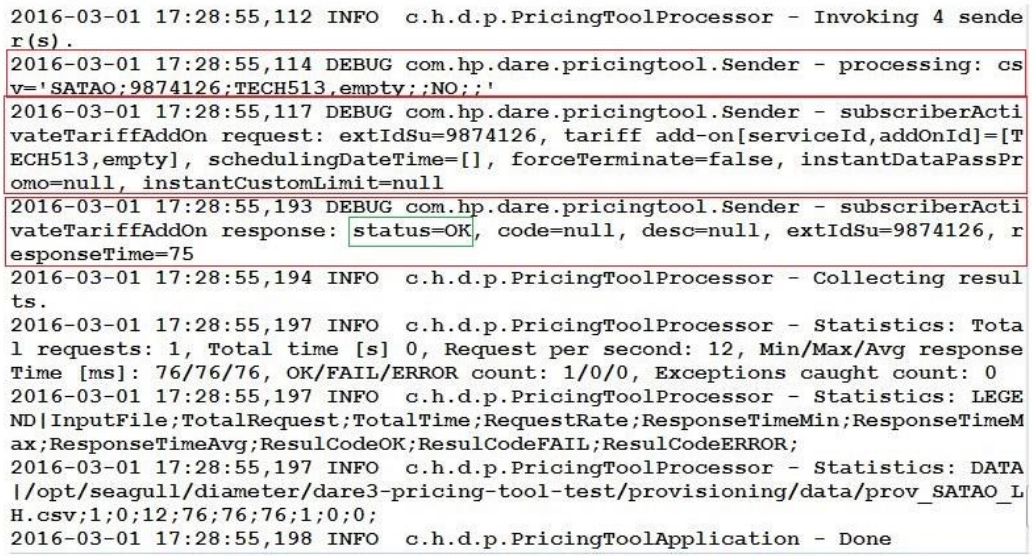
Cílem druhého kroku tohoto testovacího případu je provést aktivaci roamingového balíčku (ID 200275189309) příslušnému zákazníkovi. Nejprve je nutné, jako tomu bylo při exekuci testovacího případu č. 1, připravit samotný soubor pro aktivaci balíčku (ID 200275189309) danému zákazníkovi a po té bude soubor opět zpracován aktivačním skriptem. Soubor pro aktivaci roamingového balíčku musí mít následující strukturu: SATAO; extIdSu;tariffAddOn;;. Význam jednotlivých parametrů souboru SATAO (Subscriber Activate Tariff Add-on) byl již v této kapitole popsán při exekuci testovacího případu č. 1. Obrázek č. 24 znázorňuje vstupní soubor, který bude využit pro realizaci druhého kroku tohoto testovacího případu.



Obrázek 24 Soubor pro aktivaci roamingového balíčku Cestovatel 1, 2 addon, Zdroj: Vlastní zpracování

Aktivační skript opět provede zpracování výše uvedeného souboru (SATAO) a výsledkem tohoto procesu bude aktivaci balíčku s ID 200275189309 zákazníkovi s příslušným tarifem. Na obrázku 25 je zvýrazněna část logu, která obsahuje request

s parametry, které byly uvedeny do vstupního souboru. Dále log obsahuje response, ve které se nachází velmi důležité pole status = OK, které je na obrázku zvýrazněno zeleným rámečkem. Podle hodnoty OK parametru status je možné identifikovat, že proces aktivace proběhl správně. Za polem status následují opět parametry, které obsahuje response a které odpovídají parametrům uvedeným ve vstupním souboru



Obrázek 25 Log úspěšné aktivace roamingového balíčku Cestovatel 1, 2 addon, Zdroj: Vlastní zpracování

Na obrázku 26 je znázorněn výpis zákazníka s MSISDN 733125372, jemuž byl v přechozích krocích aktivován tarif ID 200275189299 a následně roamingový balíček ID 200275189309.

První zvýrazněná oblast obsahuje telefonní číslo zákazníka (MSISDN), jednoznačný interní identifikátor zákazníka (extIdSu) a parametr status, který nabývá hodnoty active, což znamená, že zákazníkovi je přiřazen aktivní stav. Hodnoty všech tří zmiňovaných parametrů lze označit za správné, protože hodnoty prvních dvou parametrů se shodují s hodnotami z přechozích kroků tohoto testovacího případu. Hodnotu parametru status lze také označit za správnou, protože zákazník byl v prvním kroku založen s požadovaným tarifem, a proto musí mít status aktivní.

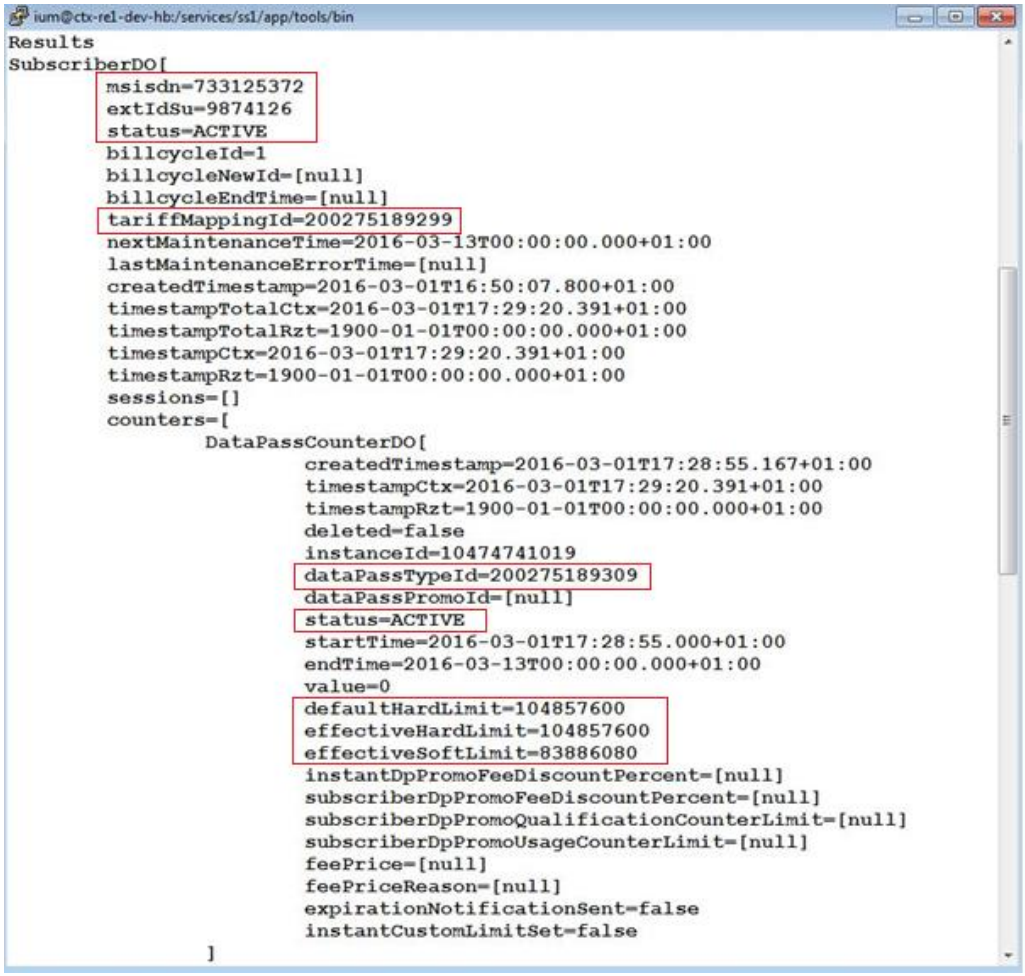
Druhá označená oblast obsahuje parametr tariffMappingId, jehož hodnota je ID tarifu, který má zákazník aktivován (v tomto případě se jedná o ID tarifu 200275189299).

Z obrázku je patrné, že se jedná o správné ID tarifu, protože tento tarif byl danému zákazníkovi aktivován v prvním kroku tohoto testovacího případu.

Sekce DataPassCounterDo obsahuje DataPassTypeId (v tomto případě je ID datového balíčku 200275189309), které souhlasí s aktivovaným balíčkem, dále obsahuje parametr status, který nabývá správně hodnoty aktivní.

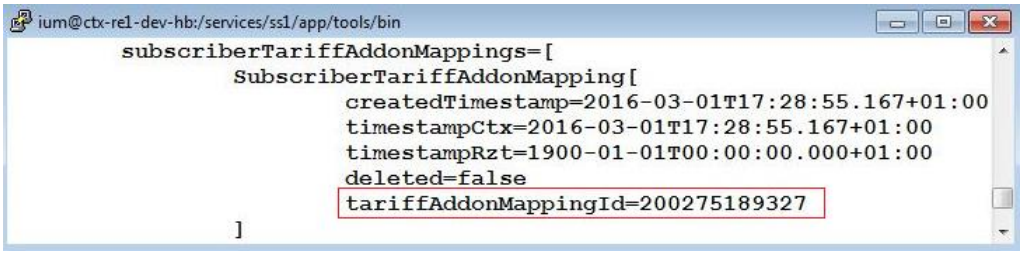
V další zvýrazněné oblasti se nachází parametr defaultHardLimit, který označuje datovým limit příslušného balíčku v bajtech (jedná se o 100 MB datový balíček). Posledním důležitým parametrem znázorněným na níže uvedeném obrázku, je parametr efectiveSoftLimit, který označuje 80% vyčerpaného limitu (v tomto případě 80 MB).

Na základě údajů z kapitoly 4.2 a výsledků testovacích případů typu ECH, je možné označit všechny hodnoty parametrů za správné.



Obrázek 26 Výpis údajů k zákazníkovi s MSISDN 733125372 před datovým provozem, Zdroj: Vlastní zpracování

Na závěr exekuce tohoto testovacího případu je nutné provést kontrolu správné hodnoty parametru tarifAddonMapingId. Jak je patrné z obrázku č. 27, hodnota parametru tarifAddonMapingId odpovídá příslušné hodnotě ID uvedené v testovacím případu č. 3 v kapitole 4.4.2.



Obrázek 27 Výpis mapovací položky ID 200275189327, Zdroj: Vlastní zpracování

V návaznosti na výše popsané kroky, které byly provedeny za účelem exekuce třetího systémového testovacího případu, je množné testovacímu případu přiřadit stav Passed.

Exekuce systémového testovacího případu č. 4

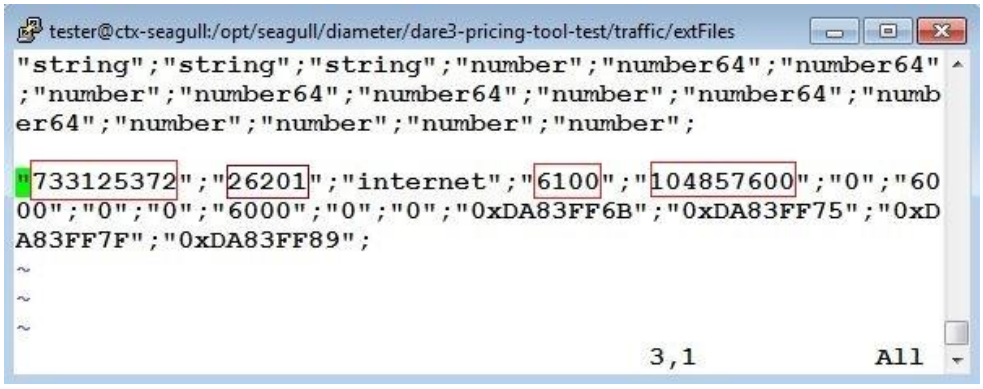
Cílem testovacího případu je otestovat vyčerpání datového limitu. Datový provoz bude generován na zákazníkovi s MSIDDN 733125372, který má tarif Cestovatel 2 (ID 200275189299) a datový balíček Cestovatel 1, 2 addon (ID 200275189309).

První krok tohoto testovacího scénáře se skládá z několika dílčích kroků. Obrázek č. 28 zobrazuje soubor, s jehož pomocí je vytvořena tzv. datovací věta. Na obrázku jsou zvýrazněny parametry, které jsou důležité pro čerpání datového limitu. Prvním parametrem je MSISDN (telefonní číslo) s hodnotou 733125372. Dalšími podstatnými parametry jsou network ID usage type, usage in MB a output. Význam a hodnoty jednotlivých parametrů jsou shodné jako v případě exekuce testovacího případu č. 2.

MSISDN	733125372				
IS TRAFFIC FREE ?	N	Y/N			
NETWORK ID	26201	23001	23106	Zona 1 - 26201 - Německo	Zona 2 - 22801 - Švýcarsko
USAGE TYPE	6100	1100 local charged	6100 roaming charged	4100 - Mobil CZ free	
USAGE IN MB	100,00	104857600	bytes		
TIME	4.3.2016 13:34:51	4.3.2016 13:35:01	4.3.2016 13:35:11	4.3.2016 13:35:21	
čas num (dny)	42433,565866203700000	42433,565981944400000	42433,566097685200000	42433,566213425900000	
čas num (s) opravený	3666083691	3666083701	3666083711	3666083721	
čas hex	DA83FF6B	DA83FF75	DA83FF7F	DA83FF89	
čas hex	D7B80948	D7B80952	D7B8095C	D7B80966	
čas num (s) opravený	3619338840	3619338850	3619338860	3619338870	
čas num (dny)	41890,495833333300000	41890,495949074100000	41890,496064814800000	41890,496180555600000	
čas	08.09.2014 11:54:00	08.09.2014 11:54:10	08.09.2014 11:54:20	08.09.2014 11:54:30	
OUTPUT	"733125372";"26201";"internet";"6100";"104857600";"0";"6000";"0";"0";"6000";"0";"0";"0xDA83FF6B";"0xDA83FF75";"0xD				

Obrázek 28 Soubor pro tvorbu datovací věty - zákazník s MSISDN 733125372, Zdroj: Vlastní zpracování

Sestavená datovací věta je následně vložena do souboru s připravenou hlavičkou. Výsledkem tohoto dílčího kroku je vytvoření datovacího souboru, jehož strukturu znázorňuje obrázek č. 29. Vytvořený soubor je opět nutné zkontrolovat. Označené oblasti obsahují telefonní číslo (MSISDN), lokalitu, v níž dochází ke generování datového provozu, typ generování datového provozu a poslední zvýrazněná oblast obsahuje hodnotu datového provozu v bajtech. Všechny hodnoty datovací věty v souboru odpovídají hodnotám, které byly zadány při tvorbě datovací věty.

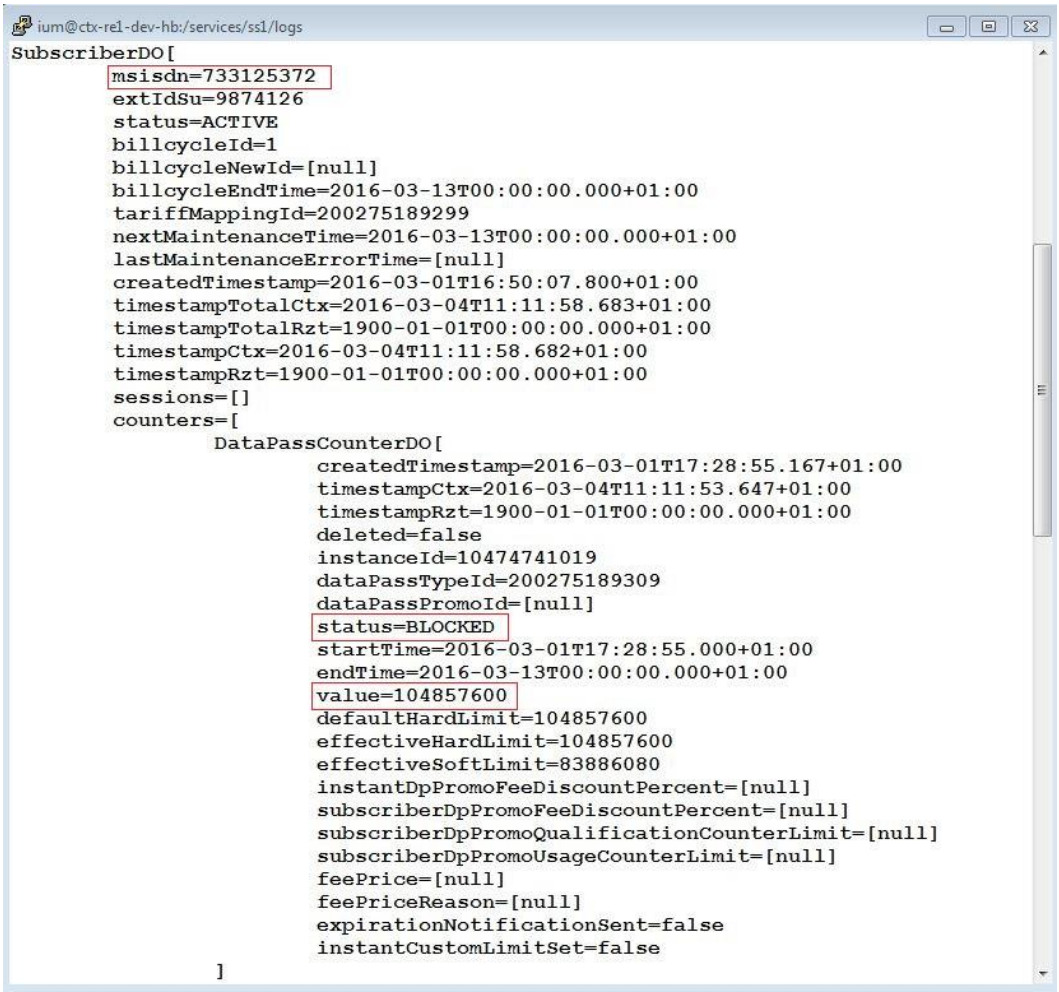


Obrázek 29 Datovací soubor – zákazník s MSISDN 733125372, Zdroj: Vlastní zpracování

Po vytvoření datovacího souboru je nutné spustit tzv. datovací skript, který tento soubor zpracuje, a tím bude provedena simulace datového provozu.

Na závěr je nutné provést kontrolu celého procesu v databázi. Obrázek č. 30 znázorňuje údaje k zákazníkovi s MSISDN 733125372. V sekci DataPassCounterDo

se nachází parametr status. Při aktivaci, jejíž realizaci se věnoval přecházející testovací případ, nabýval parametr status hodnoty active. Z níže uvedeného obrázku je patrné, že se hodnota parametru status změnila z hodnoty active na hodnotu blocked. Cílem tohoto testovacího scénáře bylo provést vyčerpání datového limitu balíčku, a proto musí dojít i ke změně hodnoty. Parametr status nabývá hodnoty blocked právě tehdy, když dojde k jeho vyčerpání. Této skutečnosti odpovídá také hodnota 104857600 (100 MB) parametru value. Zmiňovaná hodnota je limitem, při němž má dojít k vyčerpání datového balíčku a tedy i ke změně stavu parametru status.



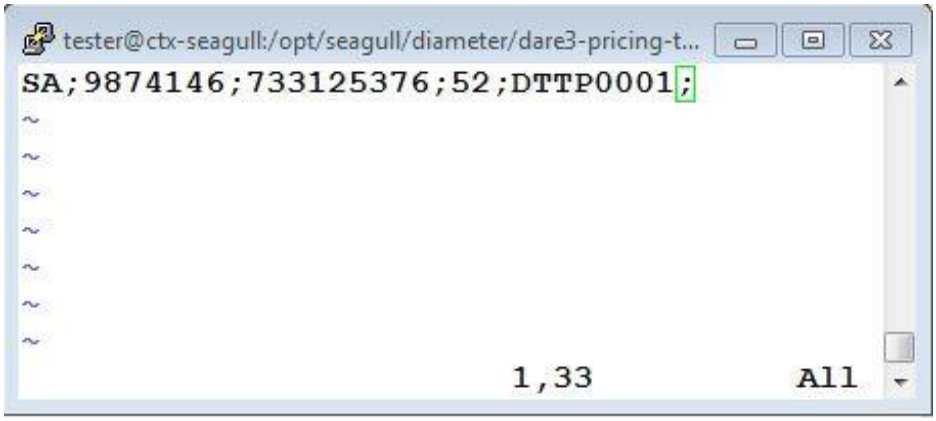
Obrázek 30 Výpis údajů k zákazníkovi s MSISDN 733125372 po datovém provozu, Zdroj: Vlastní zpracování

Na základě všech výše uvedených kroků, které byly provedeny za účelem exekuce čtvrtého systémového testovacího případu, je množné testovacímu případu přiřadit stav Passed.

Exekuce systémového testovacího případu č. 5

Cílem tohoto testovacího případu je opět otestovat procesy aktivací. Bude se jednat o založení zákazníka s telefonním číslem (MSIDDN) 733125376 a aktivaci tarifu Cestovatel 2 (ID tarifu 10000092).

Stejně jako tomu bylo v případě testovacích scénářů č. 1 a 3, i v tomto testovacím scénáři je nutné v prvním kroku provést založení zákazníka. Tento krok je opět realizován za pomoci souboru a tzv. aktivačního skriptu. Soubor je vstupem pro aktivační skript a jeho struktura musí být následující: SA;extIdSu;MSISDN;billCycle;tariffID;. Význam jednotlivých parametrů souboru SA (Subscriber Activation) byl již v této kapitole popsán při exekuci testovacího případu č. 1. Obrázek č. 31 znázorňuje vstupní soubor, který bude využit pro realizaci prvního kroku tohoto testovacího případu.



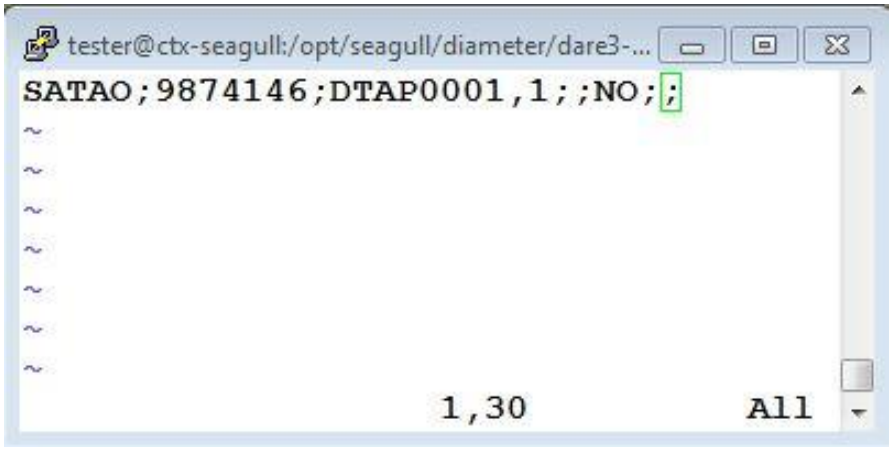
Obrázek 31 Soubor pro aktivaci tarifu Cestovatel 3, Zdroj: Vlastní zpracování

Následně bude opět za pomoci aktivačního skriptu provedeno zpracování výše uvedeného souboru a výsledkem tohoto procesu bude zákazník s aktivním tarifem s ID 10000092. Na obrázku č. 32 je zvýrazněna část logu, která obsahuje request s parametry, které byly uvedeny do vstupního souboru. Stejně jako v předešlých testovacích případech zaměřených na aktivaci tarifů a datových balíčků, obsahuje log response, ve které se nachází velmi důležité pole status = OK, které je na obrázku zvýrazněno zeleným rámečkem. Podle hodnoty OK parametru status je možné identifikovat, že proces aktivace proběhl správně. Za polem status následují parametry, které jsou v response obsaženy, a které odpovídají parametrům uvedeným ve vstupním souboru, jež byl následně zpracován aktivačním skriptem.

```
2016-03-01 21:12:11,772 INFO  c.h.d.p.PricingToolProcessor - Invoking 4 sende
r(s).
2016-03-01 21:12:11,774 DEBUG com.hp.dare.pricingtool.Sender - processing: cs
v='SA;9874146;733125376;52;DTTP0001;'
2016-03-01 21:12:11,774 DEBUG com.hp.dare.pricingtool.Sender - subscriberActi
vate request: extIdSu=9874146, msisdn=733125376, billCycle=52, tariffCode=DTT
P0001
2016-03-01 21:12:11,834 DEBUG com.hp.dare.pricingtool.Sender - subscriberActi
vate response: status=OK, code=null, desc=null, extIdSu=9874146, responseTime
=59
2016-03-01 21:12:11,834 INFO  c.h.d.p.PricingToolProcessor - Collecting resul
ts.
2016-03-01 21:12:11,837 INFO  c.h.d.p.PricingToolProcessor - Statistics: Tota
l requests: 1, Total time [s] 0, Request per second: 16, Min/Max/Avg response
Time [ms]: 59/59/59, OK/FAIL/ERROR count: 1/0/0, Exceptions caught count: 0
2016-03-01 21:12:11,837 INFO  c.h.d.p.PricingToolProcessor - Statistics: LEGE
ND|InputFile;TotalRequest;TotalTime;RequestRate;ResponseTimeMin;ResponseTimeM
ax;ResponseTimeAvg;ResulCodeOK;ResulCodeFAIL;ResulCodeERROR;
2016-03-01 21:12:11,838 INFO  c.h.d.p.PricingToolProcessor - Statistics: DATA
|/opt/seagull/diameter/dare3-pricing-tool-test/provisioning/data/prov_SA_LH2.
csv;1;0;16;59;59;59;1;0;0;
2016-03-01 21:12:11,838 INFO  c.h.d.p.PricingToolApplication - Done
```

Obrázek 32 Log úspěšné aktivace tarifu Cestovatel 3, Zdroj: Vlastní zpracování

Cílem druhého kroku tohoto testovacího případu je provést aktivaci roamingového balíčku (ID 200275184889) příslušnému zákazníkovi. Nejprve je nutné připravit samotný soubor pro aktivaci balíčku (ID 200275184889) danému zákazníkovi a po té bude soubor opět zpracován aktivačním skriptem. Soubor pro aktivaci roamingového balíčku musí mít následující strukturu: SATAO; extIdSu;tariffAddOn;;. Význam jednotlivých parametrů souboru SATAO (Subscriber Activate Tariff Add-on) byl již v této kapitole popsán, při exekuci testovacího případu č. 1. Obrázek č. 33 znázorňuje vstupní soubor, který bude využit pro realizaci druhého kroku tohoto testovacího případu.



Obrázek 33 Soubor pro aktivaci roamingového balíčku Cestovatel 3 addon, Zdroj: Vlastní zpracování

Aktivační skript opět provede zpracování výše uvedeného souboru (SATAO) a výsledkem tohoto procesu bude aktivaci balíčku s ID 200275184889 zákazníkovi s tarifem ID 10000092. Na obrázku č. 34 jsou opět označeny důležité části logu. Obsahem první označené části je request s parametry, které byly uvedeny do vstupního souboru. Další označená část logu obsahuje response, ve které se nachází velmi důležité pole status = OK, které je na obrázku zvýrazněno zeleným rámečkem. Podle hodnoty OK parametru status je možné identifikovat, že proces aktivace proběhl správně. Za polem status následují opět parametry, které obsahuje response a které odpovídají parametrům uvedeným ve vstupním souboru

```
2016-03-01 21:12:11,772 INFO  c.h.d.p.PricingToolProcessor - Invoking 4 sende
r(s).
2016-03-01 21:12:11,774 DEBUG com.hp.dare.pricingtool.Sender - processing: cs
v='SA;9874146;733125376;52;DTTP0001;'
2016-03-01 21:12:11,774 DEBUG com.hp.dare.pricingtool.Sender - subscriberActi
vate request: extIdSu=9874146, msisdn=733125376, billCycle=52, tariffCode=DTT
P0001
2016-03-01 21:12:11,834 DEBUG com.hp.dare.pricingtool.Sender - subscriberActi
vate response: status=OK, code=null, desc=null, extIdSu=9874146, responseTime
=59
2016-03-01 21:12:11,834 INFO  c.h.d.p.PricingToolProcessor - Collecting resul
ts.
2016-03-01 21:12:11,837 INFO  c.h.d.p.PricingToolProcessor - Statistics: Tota
l requests: 1, Total time [s] 0, Request per second: 16, Min/Max/Avg response
Time [ms]: 59/59/59, OK/FAIL/ERROR count: 1/0/0, Exceptions caught count: 0
2016-03-01 21:12:11,837 INFO  c.h.d.p.PricingToolProcessor - Statistics: LEGE
ND|InputFile;TotalRequest;TotalTime;RequestRate;ResponseTimeMin;ResponseTimeM
ax;ResponseTimeAvg;ResulCodeOK;ResulCodeFAIL;ResulCodeERROR;
2016-03-01 21:12:11,838 INFO  c.h.d.p.PricingToolProcessor - Statistics: DATA
|/opt/seagull/diameter/dare3-pricing-tool-test/provisioning/data/prov_SA_LH2.
csv;1;0;16;59;59;59;1;0;0;
2016-03-01 21:12:11,838 INFO  c.h.d.p.PricingToolApplication - Done
```

Obrázek 34 Log úspěšné aktivace roamingového balíčku Cestovatel 3 addon, Zdroj: Vlastní zpracování

Na obrázku č. 35 je znázorněn výpis zákazníka s MSISDN 733125376, jemuž byl v přechozích krocích aktivován tarif ID 10000092 a následně roamingový balíček ID 200275184889.

V první označené oblasti je uvedeno telefonní číslo zákazníka (MSISDN), jednoznačný interní identifikátor zákazníka (extIdSu) a parametr status, který nabývá hodnoty active, z čehož vyplývá, že zákazník má aktivní stav. Hodnoty všech tří zmiňovaných parametrů lze označit za správné, protože hodnoty prvních dvou parametrů se shodují s hodnotami z přechozích kroků tohoto testovacího případu. Hodnotu parametru

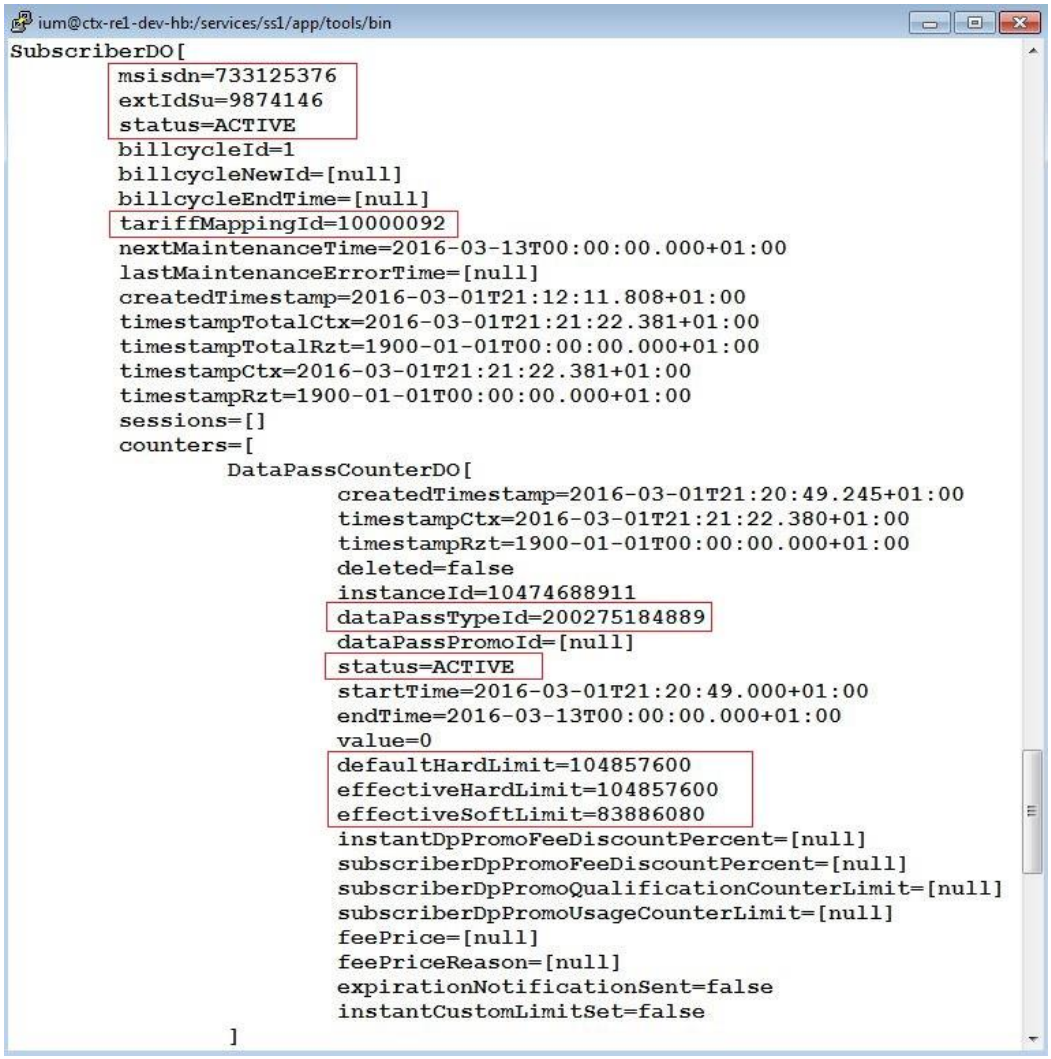
status lze také označit za správnou, protože zákazník byl v prvním kroku založen s požadovaným tarifem, a proto musí mít status aktivní.

Druhá označená oblast obsahuje parametr tariffMappingId, jehož hodnota je ID tarifu zákazníka (v tomto případě se jedná o ID 200275189299). Z obrázku je zřejmé, že je ID tarifu správné, protože tento tarif byl danému zákazníkovi aktivován v prvním kroku tohoto testovacího případu.

Sekce DataPassCounterDo obsahuje DataPassTypeId (ID datového balíčku je 200275184889), které souhlasí s aktivovaným balíčkem, dále obsahuje parametr status, jež nabývá správně hodnoty aktivní.

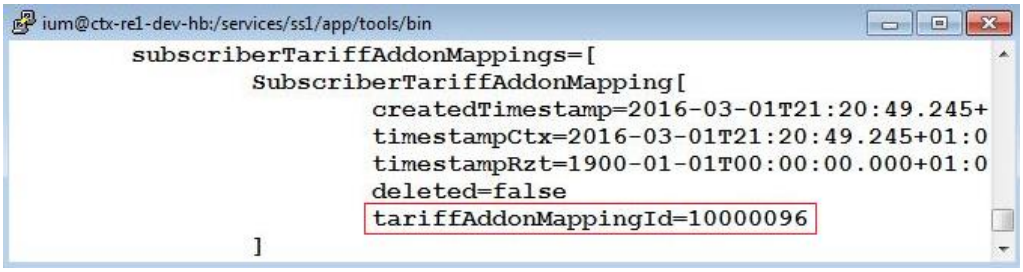
V další zvýrazněné oblasti se nachází parametr defaultHardLimit, který označuje datový limit příslušného balíčku v bajtech (jedná se o 100 MB datový balíček). Dalším důležitým parametrem na níže uvedeném obrázku je parametr efectiveSoftLimit, který označuje 80% vyčerpaného limitu (v tomto případě 80 MB).

Na základě údajů z kapitoly 4.2 a výsledků testovacích případů typu ECH, je možné označit všechny hodnoty parametrů za správné.



Obrázek 35 Výpis údajů k zákazníkovi s MSISDN 733125376 před datovým provozem, Zdroj: Vlastní zpracování

V závěru exekuce tohoto testovacího případu je nutné provést kontrolu správné hodnoty parametru tarifAddonMapingId. Jak je patrné z obrázku č. 36, hodnota parametru tarifAddonMapingId odpovídá příslušné hodnotě ID uvedené v testovacím případu č. 5 v kapitole 4.4.2.



Obrázek 36 Výpis mapovací položky ID 10000096, Zdroj: Vlastní zpracování

Na základě výše popsaných kroků, které byly provedeny za účelem exekuce pátého systémového testovacího případu, je množné testovacímu případu přiřadit stav Passed.

Exekuce systémového testovacího případu č. 6

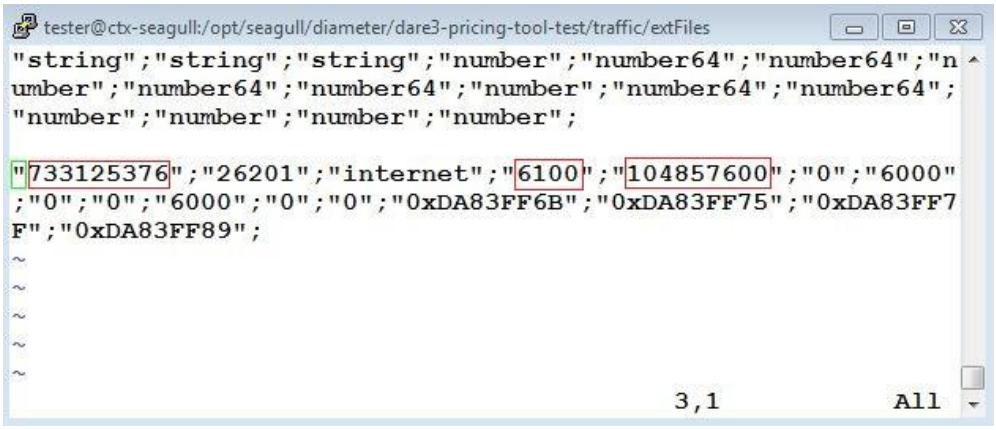
Cílem testovacího případu je otestovat vyčerpání datového limitu. Datový provoz bude generován na zákazníkovi s MSIDDN 733125376, který má tarif cestovatel 2 (ID 10000092) a datový balíček Cestovatel 3 addon (ID 200275184889).

První krok tohoto testovacího scénáře se skládá z několika dílčích kroků. Obrázek č. 37 zobrazuje soubor, s jehož pomocí je vytvořena tzv. datovací věta. Na obrázku jsou zvýrazněny parametry, které jsou důležité pro čerpání datového limitu. Prvním parametrem je MSISDN (telefonní číslo) s hodnotou 733125372. Dalšími podstatnými parametry jsou network ID usage type, usage in MB a output. Význam a hodnoty jednotlivých parametrů, stejně jako v případě testovacího scénáře č. 4, jsou shodné jako v případě exekuce testovacího případu č. 2.

MSISDN	733125376				
IS TRAFFIC FREE ?	N	Y/ N			
NETWORK ID	26201	23001	23106	Zona 1 - 26201 - Německo	Zona 2 - 22801 - Švýcarsko
USAGE TYPE	6100	1100 local charged	6100 roaming charged	4100 - Mobil CZ free	
USAGE IN MB	100,00	104857600	bytes		
TIME	4.3.2016 14:33:01	4.3.2016 14:33:11	4.3.2016 14:33:21	4.3.2016 14:33:31	
čas num (dny)	42433,606258217600000	42433,606373958300000	42433,606489699100000	42433,606605439800000	
čas num (s) opravený	3666087181	3666087191	3666087201	3666087211	
čas hex	DA840D0D	DA840D17	DA840D21	DA840D2B	
čas hex	D7B80948	D7B80952	D7B8095C	D7B80966	
čas num (s) opravený	3619338840	3619338850	3619338860	3619338870	
čas num (dny)	41890,495833333300000	41890,495949074100000	41890,496064814800000	41890,496180555600000	
čas	08.09.2014 11:54:00	08.09.2014 11:54:10	08.09.2014 11:54:20	08.09.2014 11:54:30	
OUTPUT	"733125376";"26201";"internet";"6100";"104857600";"0";"6000";"0";"0";"6000";"0";"0";"0xDA840D0D";"0xDA840D17";"0x				

Obrázek 37 Soubor pro tvorbu datovací věty - zákazník s MSISDN 733125376, Zdroj: Vlastní zpracování

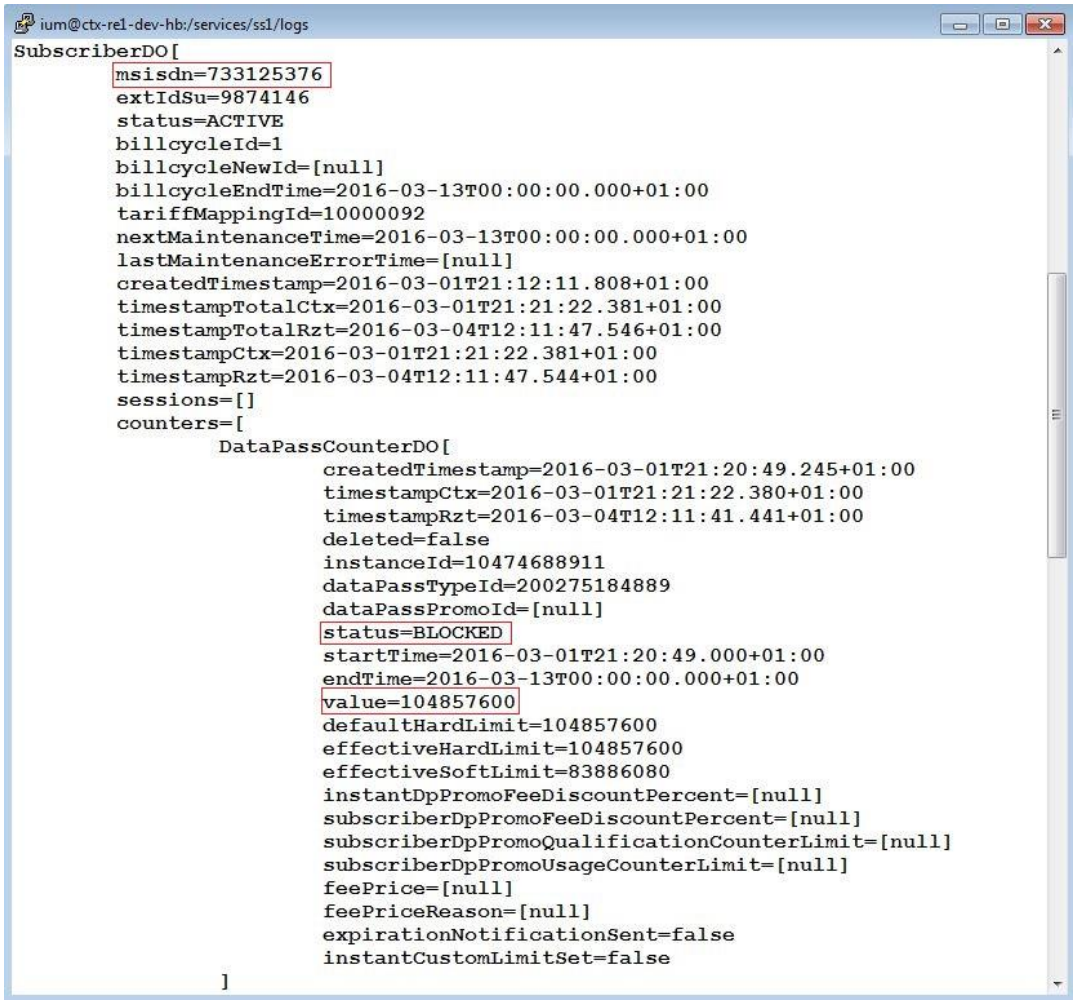
Sestavená datovací věta je následně vložena do souboru s připravenou hlavičkou. Výsledkem tohoto dílčího kroku je vytvoření datovacího souboru, jehož strukturu znázorňuje obrázek č. 38. Vytvořený soubor je nutné zkontrolovat. Označené oblasti obsahují telefonní číslo (MSISDN), lokalitu, v níž dochází ke generování datového provozu, typ generování datového provozu a poslední zvýrazněná oblast obsahuje hodnotu datového provozu v bajtech. Všechny hodnoty datovací věty v souboru odpovídají hodnotám, které byly zadány při tvorbě datovací věty.



Obrázek 38 Datovací soubor – zákazník s MSISDN 733125376, Zdroj: Vlastní zpracování

Po vytvoření datovacího souboru je nutné spustit tzv. datovací skript, který tento soubor zpracuje a tím bude provedena simulace datového provozu.

Na závěr je nutné provést kontrolu celého procesu v databázi. Obrázek č. 39 znázorňuje údaje k zákazníkovi s MSISDN 733125376. V sekci DataPassCounterDo se nachází parametr status. Při aktivaci, jejíž realizaci se věnoval předcházející testovací případ, nabýval parametr status hodnoty active. Z níže uvedeného obrázku je patrné, že se hodnota parametru status změnila z hodnoty active na hodnotu blocked. Cílem tohoto testovacího scénáře bylo provést vyčerpání datového limitu balíčku, a proto musí dojít i ke změně hodnoty. Parametr status nabývá hodnoty blocked právě tehdy, když dojde k jeho vyčerpání. Této skutečnosti odpovídá také hodnota 104857600 (100 MB) parametru value. Zmiňovaná hodnota je limitem, při němž má dojít k vyčerpání datového balíčku a tedy i ke změně stavu parametru status.



Obrázek 39 Výpis údajů k zákazníkovi s MSISDN 733125376 po datovém provozu, Zdroj: Vlastní zpracování

Na základě všech výše uvedených kroků, které byly provedeny za účelem exekuce šestého systémového testovacího případu, je množné testovacímu případu přiřadit stav Passed.

5 Zhodnocení výsledků a diskuze

V úvodu praktické části diplomové práce bylo specifikováno zadání formou dokumentace ke změnám testovaného systému. Ze zadání vyplynulo, že je nutné provést otestování třech nových tarifů (Cestovatel 1, Cestovatel 2 a Cestovatel 3), dvou nových roamingových balíčků (Cestovatel 1, 2 addon a Cestovatel 3 addon) a třech tzv. mapovacích položek tzv. Tariff Add-ons Mapping Tariffs (Cestovatel 1 addon, Cestovatel 2 addon a Cestovatel 3 addon), které zajišťují správné přiřazení balíčku zákazníkům s příslušnými tarify. Je tedy zajištěno, že si zákazník s určitým tarifem může aktivovat takový roamingový balíček, který je pro jeho tarif určen. V případě testovaného systému to znamená, že k tarifu Cestovatel 1 náleží roamingový balíček Cestovatel 1, 2 addon a mapovací položka Cestovatel 1 addon. Pro tarif Cestovatel 2 je určen roamingový balíček, stejně jako pro tarif Cestovatel 1, Cestovatel 1, 2 addon a mapovací položka Cestovatel 2 addon, což znamená, že mapovací položky Cestovatel 1 addon a Cestovatel 2 addon mapují tarify Cestovatel 1 a Cestovatel 2 na stejný roamingový balíček. Tohoto způsobu implementace může být využito zejména proto, že se liší konfigurace obou uvedených tarifů. Avšak zákazníci, kteří mají tyto tarify, si mohou aktivovat roamingový balíček Cestovatel 1, 2 addon.

Dále byl jako součást praktické části diplomové práce sestaven test plán, který definuje činnosti, zdroje, dokumenty a další potřebné nástroje pro úspěšnou realizaci projektu. V úvodu test plánu bylo jako cíl testování definováno ověření všech aktivačních procesů v rámci nově implementovaných položek. Bylo by možné realizovat další testy pro procesy deaktivace a změn tzv. upgrade, nicméně pro testování v rámci této diplomové práce byl vybrán aktivační proces, který má z obchodního hlediska pro společnost nejvyšší prioritu. Navíc zbylé dva procesy, tedy deaktivace a upgrade, probíhají identicky s tím rozdílem, že jsou za pomoci skriptů zpracovávány soubory stejné struktury pouze s jinými názvy.

Dále byla vytvořena alokační tabulka pro jednotlivé členy projektového týmu. Ten je složen z jednoho projektového manažera (alokace 0,25), dvou členů analytického týmu (alokace 0,2), dvou členů vývojového týmu (alokace 0,2) a jednoho člena testovacího týmu (plná alokace pouze na tento projekt v daném období). Pro zajištění vhodné komunikace bylo definováno názvosloví vybraných pojmů tak, aby nedocházelo

k problémům v komunikaci. Součástí test plánu je také rozdělení odpovědností, kdy je projektový manažer zodpovědný za řízení projektu a zajišťování veškerých zdrojů a prostředků pro jednotlivé členy týmu. Analytický tým zodpovídá za návrh řešení a tvorbu dokumentace, vývojový tým zodpovídá za implementaci a opravu chyb, a testovací tým zodpovídá za vytvoření test plánu, návrh testovacích případů a provedení příslušných testů. V rámci projektu by bylo možné přidat některé další role (např. test manažer), či oddělit roli testera a test analytika, nicméně pro potřeby tohoto projektu a s ohledem na jeho rozpočet byly jednotlivé role zvoleny výše uvedeným způsobem.

Za rozdělením odpovědností následuje v test plánu volba fáze testování. Pro daný projekt bylo zvoleno systémové testování, protože dochází pouze ke změnám v rámci testovaného systému, a proto není nutné věnovat se integračnímu testování, které by zahrnovalo i ostatní systémy. Pokud by bylo zvoleno systémové integrační testování, jednalo by se v rámci okolních systémů o regresní integrační testování, které bylo v rámci tohoto projektu s ohledem na čas a finanční zdroje vyhodnoceno jako nepotřebné.

V rámci strategie testování bylo rozhodnuto, že na testovaný systém bude aplikováno tzv. black-box testování, protože tento způsob testování, kromě funkcionálních chyb, odhalí i případné chyby v kódu programu, které budou následně odstraněny vývojářem.

Projekt byl zahájen 2. 2. 2016 a předpokládaný konec byl naplánován na 29. 2. 2016. Dle plánu byla technická dokumentace dokončena v řádném termínu, tj. do 5. 2. 2016. Implementace byla dokončena do 14. 2, test analýza probíhala od 8. 2. do 14. 2. a v termínu od 15. 2. do 16. 2. probíhala instalace na testovací prostředí. Samotné testování probíhalo od 17. 2. do 29. 2. Nasazení na produkční prostředí bylo provedeno 10. 3, přičemž projekt byl nasazen v rámci releasu s ostatními projekty, které byly do příslušného releasu zařazeny. Období od 1. 3. do 9. 3. bylo pro projekt vyčleněno jako časová rezerva pro případ, že by vznikly problémy, které by znamenaly zpoždění. Období časové rezervy bylo zvoleno po dohodě projektového manažera s ostatními členy projektového týmu tak, aby bylo minimalizováno riziko, že projekt nebude moci být nasazen 10. 3. 2016 na produkční prostředí. Tuto rezervu nebylo nutné využít a všechny termíny v rámci harmonogramu projektu byly splněny.

Ze strany testovacího týmu byla identifikována rizika nedodržení jednotlivých termínů v rámci harmonogramu, chybná konfigurace testovacího prostředí a neaktuální dokumentace. Těmto aspektům byla v průběhu realizace projektu věnována zvýšená pozornost proto, aby se projektový tým těmto rizikům vyhnul, což se nakonec podařilo a žádné z identifikovaných rizik nenastalo.

Na základě zadání (dokumentace) byly navrženy testovací případy, které byly při test analýze rozděleny na dvě skupiny, tzv. entry check (ECH) a systémové testy (ST).

Pro testovací případy ECH byla stanovena priorita 1, protože jejich účelem bylo ověřit, zda jsou nově implementované položky testovaného systému „přítomny“ v produktovém katalogu a jestli mají správnou konfiguraci.

V prvním kroku prvního testovacího případu ECH byla provedena kontrola přítomnosti položky Cestovatel 1 v seznamu tarifů produktového katalogu. Tarif byl vyhledán dle ID 200275189297. Ve druhém kroku byla provedena kontrola parametrů Tarrif code 512 a Tariff group Postpaid zero rate national, kdy byla hodnota z produktového katalogu porovnána s hodnotou uvedenou v dokumentaci. Hodnoty byly shodné, a proto bylo možné přiřadit prvnímu testovacímu případu stav passed.

Obdobně byly navrženy testovací případy typu ECH pro zbylé dva tarify, dva roamingové balíčky a tři mapovací položky. Je možné se domnívat, že tato kontrola je zbytečná, nicméně úkolem testovacích scénářů typu ECH je ověřit, že při implementaci a konfiguraci nových položek nebylo nic opomenuto a že je vše připraveno pro zahájení systémových testů. Tento přístup byl zvolen z toho důvodu, že čím dříve je taková chyba nalezena, tím nižší jsou náklady na její odstranění.

Dále se může nabízet otázka, jestli by nebylo možné tuto jednoduchou kontrolu konfigurace nějakým způsobem urychlit. Jedním z možných řešení, jak by bylo možné ECH testy provést, je za použití automatizovaných testů. Je však nezbytné zvážit, jestli tím dojde k ušetření zdrojů (čas nebo finanční zdroje) či nikoliv. Automatizovat testy je efektivní pouze tehdy, pokud se jedná o velký objem testovacích případů, které jsou prováděny opakovaně bez jakékoliv změny nebo s minimálními změnami. V případě výše uvedených testů se jednalo pouze o několik položek. V testovaném systému nebyly,

a do budoucna s velkou pravděpodobností nebudou, realizovány žádné zásadní změny, které by podporovaly myšlenku implementace automatizovaných testů.

Druhou skupinou testovacích případů v rámci test analýzy byly systémové testy (ST), jejichž cílem bylo ověřit funkcionalitu jednotlivých aktivačních procesů a následný datový provoz s využitím nově implementovaných položek testovaného systému.

První systémový testovací případ ověřuje založení zákazníka s novým tarifem Cestovatel 1, který má roamingový datový balíček Cestovatel 1, 2 addon, jež je mapován na příslušný tarif pomocí mapovací položky Cestovatel 1 addon.

V prvním kroku byla provedena aktivace zákazníka pomocí souboru SA (subscriber activation), který byl připraven dle údajů z dokumentace a spuštěn pomocí tzv. aktivačního skriptu. Následně byla provedena kontrola odeslané response a hodnoty parametru status, který obsahoval hodnotu ok. Tímto krokem byl založen zákazník s telefonním číslem 733125374 a tarifem Cestovatel 1.

Ve druhém kroku byla provedena aktivace roamingového balíčku Cestovatel 1, 2 addon zákazníkovi s telefonním číslem 733125374 za pomocí souboru SATAO (subcriber activation tariff addon), který byl opět vytvořen na základě údajů z dokumentace a zpracován tzv. aktivačním skriptem. Po proběhnutí skriptu byla opět provedena kontrola odeslané response, která obsahovala hodnoty uvedené v dokumentaci. Dále byla provedena kontrola v databázi, kde byly ověřeny hodnoty parametrů MSISDN, extIdSU, tariffMapingId, DataPassTypeId, defaultHardLimit, defaultSoftLimit, tariffAddonMapping Id a parametru status, který nabýval hodnoty ok. Bylo nezbytné, aby hodnoty jednotlivých parametrů odpovídaly hodnotám uvedeným v dokumentaci.

Může se zdát, že kontrola jednotlivých parametrů zasílaných v response je zbytečná, nicméně může nastat situace, kdy aktivační skript proběhne se statusem ok, ale dojde k chybnému přenosu, jednoho z parametrů, proto byla provedena i kontrola jednotlivých hodnot zasílaným v response.

Druhý systémový testovací případ byl věnován ověření správné funkce roamingového balíčku při generování datového provozu ze zahraničí. První krok obsahuje několik dílčích kroků. Nejprve bylo nutné sestavit tzv. datovací větu, která je sestavena pomocí souboru vytvořeného v tabulkovém editoru MS Excel. Ve zmíněném souboru bylo

nutné vyplnit telefonní číslo (MSISDN) 733125374, hodnotu parametru network ID 26201, který udává zónu, ve které bylo prováděno čerpání roamingového balíčku. Pro testování v rámci této diplomové práce se jedná o zónu 1 (země EU). Dále bylo nutné vyplnit hodnotu parametru usage type 6100, který definuje typ datového provozu. V rámci uvedeného testování se jedná o zahraniční datový provoz v zemích EU. Na závěr bylo nutné zvolit hodnotu parametru usage in MB, která definuje objem dat, jejichž čerpání bylo simulováno. Parametru byla přiřazena hodnota 100 MB, protože bylo zapotřebí vyčerpat celý roamingový balíček, jenž obsahuje 100 MB dat pro zónu 1. Následně byla v poli output sestavena ze zadaných parametrů tzv. datovací věta, která byla zkopírována do připraveného souboru pro generování datového provozu. Poté byl nad souborem spuštěn tzv. datovací skript, který zpracoval vstupní soubor s datovací větou. Výsledkem byla simulace datového provozu v zemích EU. Druhým krokem testovacího případu bylo provedení kontroly v databázi, že parametr MSISDN nabývá hodnoty 733125374, parametr status v sekci DataPassCounterDo nabývá hodnoty blocked, která označuje vyčerpání datového limitu příslušného balíčku a parametr value nabývá hodnoty 104857600 B (100MB), tj. objem dat, jejichž čerpání bylo simulováno.

V případě generování datového provozu se opět nabízí otázka zefektivnění testování. V rámci tohoto procesu je možné doporučit vytvoření automatizovaných testů, které na základě vyplnění vstupních hodnot provedou simulaci datového provozu s odpovídajícími parametry. Na rozdíl od testovacích případů typu ECH se jedná o stejný proces, který je při testování velmi často využíván, a tak se dá předpokládat, že by došlo k úspoře času i finančních zdrojů, nicméně varianta automatizace nebyla předmětem této diplomové práce.

Zbylé testovací případy tvoří stejně jako první dva výše zmíněné dva páry, přičemž třetí a pátý systémový testovací případ, se věnují aktivačnímu procesu. V rámci těchto testovacích případů bylo postupováno stejně jako v prvním systémovém testovacím případu ovšem za použití jiných parametrů. V případě třetího testovacího případu se jedná o aktivaci tarifu Cestovatel 2, datového balíčku Cestovatel 1, 2 addon pomocí mapovací položky Cestovatel 2 addon. Pátý systémový testovací případ je věnován aktivaci tarifu Cestovatel 3, datového balíčku Cestovatel 3 addon a mapovací položky se stejným názvem, jako v případě balíčku, tedy Cestovatel 3 addon. Čtvrtý a šestý systémový

testovací případ jsou zaměřeny na ověřování správného chování datových balíčků Cestovatel 1, 2 addon a Cestovatel 3 addon při vyčerpání datového limitu, stejně jako tomu bylo v případě druhého systémového testovacího případu.

Může se nabízet otázka, proč navazující testovací případy nejsou spojeny do jednoho testovacího případu. Z procesního hlediska se jedná o dva různé procesy, a proto by bylo chybou, kdyby byly testovací případy spojeny do jednoho celku, protože pokud by nastala chyba při generování datového provozu, byl by i proces aktivací označen za failed. Pokud je však proces aktivací oddělen od procesu datového provozu, má následný report, který znázorňuje progres v testování vyšší vypovídací hodnotu, než kdyby oba testovací případy byly spojeny do jednoho.

6 Závěr

Diplomová práce se zabývá návrhem strategie testování softwaru, konkrétně systému pro simulaci datového provozu.

V úvodu kapitoly „Teoretická východiska“ byly popsány jednotlivé modely životního cyklu vývoje softwaru a metodiky vývoje softwaru a testování, přičemž hlavní pozornost byla věnována významu testování a jeho pozici v rámci jednotlivých modelů a metodik. Dále byla představena vybraná témata z oblasti testování, jejichž znalost byla nezbytná pro realizaci praktické části diplomové práce. Na základě uvedených skutečností byl splněn jeden z dílčích cílů práce, jehož úkolem bylo představit vybrané metody a přístupy k testování.

Kapitola “Praktická část“ byla věnována samotné realizaci. V jejím úvodu byl popsán testovaný software. Jednalo se o systém pro simulaci datového provozu, který zajišťuje realizaci zahraničního datového provozu tzv. roaming. Dále byla uvedena dokumentace k nově implementovaným změnám systému pro simulaci datového provozu, která sloužila zároveň jako zadání pro testování. Následně byla navržena strategie testování, která byla realizována ve formě test plánu, jenž zahrnoval všechny nezbytné prostředky pro realizaci testování. V rámci strategie testování bylo zvoleno testování funkcionality systému pro simulaci datového provozu. Dále bylo zvoleno black-box testování realizované formou systémových testů. Testovací případy byly navrženy aplikací techniky testování přechodů stavů, přičemž byly vybrány konkrétní procesy aktivací a simulace datového provozu, jakožto klíčové a nejčastěji se vyskytující procesy v rámci testovaného systému. Návrhem testovací strategie ve formě test plánu došlo k naplnění hlavního cíle diplomové práce. Dále byl proveden návrh testovacích případů, které byly rozděleny do dvou skupin a vycházely z kapitoly “Dokumentace ke změnám testovaného systému“. První skupinou byly testovací případy entry check (ECH), jejichž cílem bylo ověřit správnou konfiguraci nově implementovaných položek. Druhou skupinou testovacích případů byly systémové testy (ST), jejichž cílem bylo otestovat procesy aktivací nových tarifů, datových balíčků a správnou změnu stavů nově implementovaných datových balíčků při jejich vyčerpání. Následně byly provedeny všechny navržené testovací případy, přičemž nedošlo k nalezení žádného defektu a byla tak úspěšně ověřena správná funkcionality nově implementovaných položek systému pro simulaci datového

provozu. Provedením navržených testovacích případů došlo ke splnění druhého dílčího cíle diplomové práce, jehož úkolem bylo aplikovat vybraný přístup a metodu v rámci navržené testovací strategie.

Na závěr bylo provedeno zhodnocení dosažených výsledků v rámci návrhu testovací strategie i samotných testů, které byly doplněny diskuzí k možnosti volby jiného způsobu či přístupu testování. Podstatou návrhu správné testovací strategie je zohlednit charakter testovaného systému, prostředí společnosti, dostupnost jednotlivých zdrojů a dosažitelnost požadovaných cílů tak, aby výsledná testovací strategie odrážela všechny tyto požadavky a jednoznačně vedla ke splnění cíle testování.

Seznam obrázků

Seznam tabulek

Seznam použitých zdrojů

ANSI/IEEE č. 829. 1998. *IEEE Standard for Software Test.* [Dokument] New York : Software Engineering Technical Committee, 1998. ISBN 0738114448.

BLACK, Rex. 2002. *Managing the Testing process: Practical Tools and techniques for Managing Hardware and Software Testing.* Toronto : Wiley Publishing, 2002. ISBN 0471223980.

BUCHALCELOVÁ, Alena a KUČERA, Jan. 2008. *Systémová integrace,.* roč. 15. Praha : Katedra informačních technologií VŠE Praha, 2008. stránky s. 42 - 54. ISSN 12109479.

BURNSTEIN, Ilene. 2003. *Practical software testing: A process- oriented approach.* New York : Springer, 2003. ISBN 0387951318.

FALLEY-VINAY, Peter. 2008. *Manage software testing.* Boca Raton : Auerbach, 2008. ISBN 9780849393839.

CHRISPIN, Lisa a GREGORY, Janet. 2009. *Agile testing: a practical guide for testers and agile teams.* New Jersey : Addison-Wesley, 2009. ISBN 9780321534460.

ISTQB. 2013. Certifikovaný tester: Učební osnovy pro základní stupeň. [Online] 2013.

JORGENSEN, Paul. 2008. *Software testing: a craftsman's approach.* Boca Raton : Auerbach, 2008. ISBN 9780849374753.

KANER, Cen, BACH , James Marcus a PETTICHORD, Bret . 2002. *Lessons Learned in Software Testing: A Context-Driven Approach.* New York : John Wiley & Sons, 2002. ISBN 9780471081128..

KOIRALA, Shivprasad a SHEIKH, Sham. 2008. *Software Testing: Interview Questins.* Hingham : Infinity science press, 2008. ISBN 9781934015247.

LEFFINGWELL, Dean. 2011. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise.* Boston : Addison-Wesley, 2011. ISBN 0321635841.

MICHÁLEK, Lukáš. 2006. *Použití metod testování softwaru v praxi.* [Bakalářská práce] Praha : VŠE Praha, 2006.

MILI, Ali a FAIROUZ, Tchier. 2015. *Software testing : concepts and operations.* Quantitative software engineering series. Hoboken : John Wiley & Sons, 2015. ISBN 9781118662878.

MYERS, Glenford , BADGETT , Tom a SANDLER, Corey. 2012. *The Art of Software Testing.* New Jersey : John Wiley & Sons, 2012. ISBN 9781118031964.

NAIK, Kshirasagar a PRIYADARSHI, Tripathy. 2008. *SOFTWARE TESTING AND QUALITY ASSURANCE: Theory and Practice.* New Jersey : JOHN WILEY & SONS, 2008. ISBN 9780471789116.

PATTON, Ron. 2002. *Testování softwaru.* Brno : Compute press, 2002. str. 313. ISBN 8072266365.

PERRY, William . 2006. *Effective Methods for Software Testing.* Indianapolis : John Wiley & Sons, 2006. ISBN 9780764598371.

SOMMERVILLE, Ian. 2013. *Softwarové inženýrství.* Brno : Computer Press, 2013. ISBN 9788025138267.

SPILLNER, Andreas. 2011. *Software Testing Evaliation Foundations: A Study Guide for the Certified Tester Exam.* Santa Barbara : Rocky Nook, 2011. ISBN 9781933952789.

STEPHENS, Matt a ROSENBERG, Doug . 2011. *Testování softwaru řízené návrhem.* Brno : Computer Press, 2011. ISBN 9788025136072.

Testování softwaru. RUP – Rational Unified Process. *http://testovanisoftwaru.cz.* [Online] [Citace: 18. Listopadu 2015.] http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/rup/.

TIAN, Jeff. 2005. *Software Quality Engineering: Testing, Quality Assurance and Quantifiable Improvment.* New Jersey : John Wiley & Sons, 2005. ISBN 0471713457.

VAN VLIET, Hans. 2008. *Software Engineering: Principles and Practice.* West Sussex : John Wiley & Sons, 2008. ISBN 9780470031469.

WATKINS, John. 2001. *Testing IT : An Off-the-Shelf Software Testing Handbook.* Cambridge : Cambridge University Press, 2001. ISBN 9780521795463.