Testování aplikací pro mobilní telefony

Testing of mobile applications

Bc. Martin Malaník

Diplomová práce
2011

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin MALANÍK**
Osobní číslo:	**A09483**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Testování aplikací pro mobilní telefony**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na dané téma. Zaměřte se na principy a metodiku, uveďte rozdíl proti testování aplikací pro platformu PC. Soustřeďte se zejména na operační systémy Android a iPhone OS.**
2. **Proveďte analýzu nejčastějších problémů při testování mobilních aplikací včetně hodnocení rizik a možných důsledků.**
3. **Navrhněte optimální strategii a postup (optimálně ve formě vývojového diagramu, tzv. testing skeleton) pro black-box testování aplikací pro mobilní telefony. Zohledněte analýzu rizik při definici prioritizace. Odhadněte náročnost každého kroku (např. v procentech celkového času), nejlépe ve formě srovnávací tabulky.**
4. **Navržený postup aplikujte na vybranou aplikaci a proveďte její black-box testování z hlediska trasovatelnosti zákaznických požadavků. Vypracujte design testů a testovací závěrečnou zprávu.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **PATTON, Ron. Testování softwaru. 1. vyd. Praha: Computer Press, 2002**
2. **GLENFORD J. Myers. The Art of Software Testing. John Wiley and Sons, Inc., 2th edition, 2004**
3. **Testing Mobile Applications is Different from Testing Traditional Application. A white paper by L1onbridge, USA, 2006**
4. **Google : Android [online]. 2010. Dostupné z WWW: http://www.android.com**
5. **Apple : iOS [online]. 2010. Dostupné z WWW: http://www.apple.com/ios**
6. **ŠTRBÁK, Martin; VALIENTOVÁ, Silvia; GRÖSSL, Zdeněk. Blog o testování [online]. 2010. Dostupné z WWW: http://cz-testing.blogspot.com**

Vedoucí diplomové práce: **Ing. František Gazdoš, Ph.D.**
Ústav řízení procesů

Datum zadání diplomové práce: **24. února 2011**

Termín odevzdání diplomové práce: **18. května 2011**

Ve Zlíně dne 24. února 2011

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce je analyzovat a navrhnout optimální postup pro testování aplikací pro mobilní telefony (tzv. mobilních aplikací). Teoretická část se skládá ze shrnutí dosavadních postupů a znalostí z oboru testování, identifikace rozdílů mezi testováním počítačového softwaru a aplikací pro mobilní zařízení a analýzou podpory testování v operačních systémech. Praktická část se pak zabývá vytvořením jednotného testing skeletonu, který popisuje jednotlivé činnosti testera během projektu. Použitelnost tohoto testing skeletonu je ověřena praktickým otestováním aplikace pro mobilní operační systém Android.

Klíčová slova: Testování, Testování aplikací pro mobilní zařízení, Android, iOS, Testování černé skříňky, Testing skeleton

ABSTRACT

The aim of this thesis is to analyze and suggest optimal procedure for testing of applications for mobile phones (i.e. mobile applications). The theretical part consists of existing procedures and knowledge summary in testing domain, identification of differences between PC software testing and mobile applications testing and analysis of testing support in operating systems. The practical part is concerned with creating of unified testing skeleton which describes the various tester's activities during the project. Usability of this testing skeleton is verified by practical testing of application for Android mobile operating system.

Keywords: Testing, Testing of mobile applications, Android, iOS, Black-box testing, Testing Skeleton

Na tomto místě bych rád poděkoval konzultantovi své diplomové práce Ing. Davidu Janotovi, PhD. za pomoc a veškeré náměty. Dále pak svému velmi dobrému kamarádovi a spolužákovi Ing. Petru Janků za vzájemnou pomoc při studiu. A v neposlední řadě také své rodině a přítelkyni Lucii za jejich obrovskou podporu.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně ……

podpis diplomanta

OBSAH

ÚVOD

Současný trh je přesycen různými vývojářskými týmy či outsourcingovými společnostmi, které se doslova předhánějí v nabídkách pro potenciální zákazníky. Pro zadavatele projektu se tak sice nabízí možnost širokého výběru z rozmanitého spektra uchazečů, zároveň díky tomu však roste i nebezpečí špatné volby a s ní spojené zbytečné vynaložení financí a úsilí, které je ve výsledku odměněno nekvalitním produktem.

Stále více společností tak přichází na to, že je to právě kvalita, která jim může pomoci zákazníka nejen oslovit, ale i udržet. Z tohoto důvodu jsou zaváděny tzv. quality assurance oddělení (zkráceně QA), která mají za úkol dohlížet na kvalitu jak finálních produktů, tak vnitřních procesů firmy a s tím spojené i efektivnější využívání zdrojů.

Na poli vývoje softwaru se pak procesy zajištění kvality aplikují především užitím testování vyvíjeného produktu. Ačkoli se tato činnost může zdát poněkud jednoduchá, opak je pravdou. Pokud má být provedeno správně, jde o sofistikovaný proces se specifickými pravidly, postupy a metodikami, při kterém je kladen velký důraz jak na správné výsledky, tak na erudovanost testera.

V posledních letech se navíc stále víc a víc projevuje fenomén mobilních zařízení a to jak na poli hardware (v podobě nových a rychlejších zařízení), tak v oblasti software vývojem nových a nových aplikací a operačních systémů. Na toto samozřejmě reagují i společnosti zabývající se vývojem softwaru - přetváří své aplikace do úspornějších mobilních podob a vyvíjejí nové, určené speciálně pro využití v mobilních zařízeních.

S touto změnou vzniká otázka, zda je nutné novému směru přizpůsobit i přístup k testování. Ačkoliv se mnozí domnívají, že téměř žádné odlišnosti neexistují, tak opak může být pravdou - testování mobilních aplikací se skutečně v některých pohledech liší od testování klasického počítačového software. Cílem této práce je tyto odlišnosti najít a identifikovat, určit jejich optimální řešení a vytvořit tak nový, komplexní přístup k testování aplikací pro mobilní zařízení pomocí tzv. testing skeletonu.

I. TEORETICKÁ ČÁST

1 TESTOVÁNÍ SOFTWARU A ZAJIŠTĚNÍ KVALITY

Softwarové systémy jsou rozšiřující se součástí života, od obchodních aplikací (např. bankovnictví) až po spotřebitelské produkty (např. automobily). Bohužel, ne vždy tyto systémy pracují správně a většina lidí má tak zkušenosti se softwarem, který nepracoval,jak se očekávalo. Software, který nepracuje správně, může způsobit mnoho problémů, včetně ztráty peněz, času nebo obchodní reputace, může dokonce způsobit zranění nebo smrt. [16]

Důsledné testování softwaru a dokumentace mohou pomoci snížit riziko těchto problémů. Tím přispívají ke kvalitě softwarového systému (jestliže jsou zjištěné defekty opraveny dřív, než je systém uvolněn do provozu). [16]

Pomocí testování je možné měřit kvalitu softwaru ve smyslu zjištěných defektů, pro funkcionální a také pro nefunkcionální softwarové požadavky a charakteristiky (např. spolehlivost, použitelnost, účinnost, udržovatelnost a přenositelnost). Samozřejmě, že testování SW má i svá pravidla a ověřené postupy. Pro pochopení celé problematiky jsou v dalších kapitolách definovány základní pojmy [16].

1.1 Definice chyby

1.1.1 Případová studie – Problém roku 2000 (Y2K)

V sedmdesátých letech pracoval pro svoji firmu jistý počítačový programátor. Jeho úkolem bylo vyvinout mzdový systém, který by usnadnil práci administrativnímu oddělení. V této době však počítače disponovaly relativně malou operační pamětí, a proto bylo nutné psát co nejúspornější programy. Zmíněný programátor tento problém vyřešil tak, že údaje o datech zkrátil ze čtyř cifer (např. „1973“) na pouhé cifry dvě (tedy „73“). Díky této úpravě bylo ušetřeno velké množství místa v paměti. To vše v domnění, že za 25let bude program jistě nahrazen novým, rychlejším. V roce 1995 byl však program i nadále používán. Jeho autor mezitím odešel do důchodu a nikdo nevěděl, zda program zvládne přechod na rok 2000. [1]

Odhaduje se, že v rámci náhrady a aktualizace počítačových programů jako byl tento a v rámci potenciálního selhání z důvodu problému roku 2000 (Y2K) bylo nutné investovat několik stovek miliard dolarů po celém světě. [1]

1.1.2 Případová studie – Webové stránky Ministerstva průmyslu a obchodu ČR

Jedna nejmenovaná společnost, která se zabývá mj. i zajišťováním kvality, realizovala bezpečnostní testování modernizovaných webových stránek Ministerstva průmyslu a obchodu ČR. Díky svým zkušenostem a vhodně zvoleným postupům byla schopna provést svou práci během několika dní. Výstupem tohoto testování bylo odhalení několika chyb, které tak mohly být odstraněny ještě před nasazením stránek do ostrého provozu. Odhaduje se, že při neopravení nalezených chyb by vzniklé škody mohly dosáhnout až několika milionové částky. [17]

Tento případ naopak dokazuje, že kvalitní testování pomáhá předejít případným budoucím problémům (např. Útok hackera) a s tím spojené např. nemalé finanční náklady.

1.1.3 Co je to chyba?

Z předchozích kapitol je jasné, co všechno se může stát při selhání softwaru a jak může kvalitní testování tomuto selhání předcházet. Ve všech těchto případech je totiž zřejmé, že software nepracoval v souladu s původním záměrem. Většina selhání je docela malých a některé z nich dokonce tak bezvýznamné, že není úplně vždy jasné, jestli se jedná o skutečné selhání – chybu. [1]

Důsledkem selhání softwaru může být tedy určité nepohodlí = chyba. Tímto nepohodlím může být cokoli od znechucení uživatele až po milionové ztráty či dokonce katastrofu, která může zapříčinit ztrátu na životech. [1]

1.2 Co je cílem testera?

Cíl (nejen) softwarového testera je definován poměrně jednoduše: *„vyhledávat chyby, a to vyhledávat je co nejdříve a zajistit jejich nápravu“*. [1] Jednotlivé části této definice budou rozebrány v následujících kapitolách.

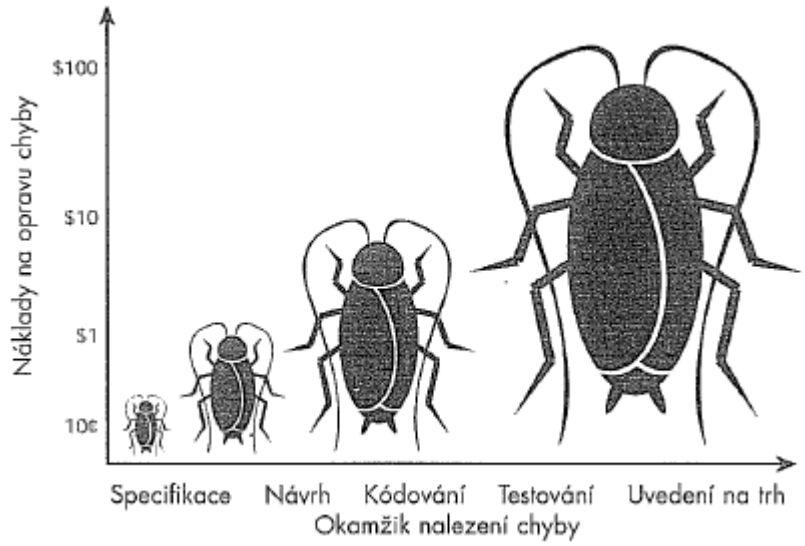
1.2.1 Cíl softwarového testera: „…vyhledávat chyby…“

Pro softwarového testera je prioritou vyhledávat chyby. Pokud chybu nenajde, pak existují jen dva důvody. Buď je systém bezchybný (v praxi méně obvyklé až nemožné) nebo tester neprovedl svou práci dobře (bohužel nejčastější). Je proto nutné vytvářet testy, které se

snaží chybu skutečně odhalit a tím snížit možné budoucí prodražení celého projektu a s tím spojené zvýšení nákladů na odstranění chyby. [1]

1.2.2 Cíl softwarového testera: „…vyhledávat chyby co nejdříve…“

Kvalitní testování je spojeno nejen se schopností vyhledávat chyby, ale i s rychlostí jejich vyhledání. S rostoucím časem totiž rostou i náklady na opravu chyby (viz Obrázek 1). Pokud například chybu objevíme již během vývoje, může oprava zabrat vývojáři 30 minut jeho času. Pokud však na chybu narazí až zákazník, budou náklady patrně vyšší, a to včetně možné ztráty zákazníkovy důvěry. [1]



Obrázek 1 – Graf závislosti nákladů na čase objevení chyby. [1]

1.2.3 Cíl softwarového testera: „…zajistit opravu chyb…“

Nalezením chyby testerova práce nekončí. Chybu je totiž nutné oznámit např. vývojáři (ve formě zdokumentování chyby do reportovacího systému týmu) a jakmile tento vývojář chybu opraví, je nutné opět zkontrolovat, že vše bylo opraveno správně a oprava nezasáhla ostatní části systému. [1]

1.3 Zajišťování kvality (QA)

Následující text se zabývá vysvětlením pojmu „QA“ a popisem některých systému řízení jakosti.

1.3.1 Co je QA?

QA neboli „Quality Assurance“ se dá do českého jazyka volně přeložit jako zajišťování kvality. Hlavním úkolem osoby odpovědné za zajišťování kvality softwaru je zkoumání a měření současného procesu vývoje softwaru a nalezení způsobů jeho zdokonalení, jejichž cílem je zabránit vzniku chyb. [1]

Skupina zajišťování kvality softwaru má mnohem širší záběr a odpovědnost než skupiny určené pro testování softwaru – anebo by podle své pracovní náplně aspoň měla mít. Kromě toho, že provádí testování softwaru nebo jeho část, má také za úkol předcházení vzniku chyb a zajišťovat určitou, samozřejmě vysokou, kvalitu a spolehlivost softwaru. To znamená, že neprovádí jen testování a oznamování chyb – její povinnosti jdou mnohem dále – například zavádění a udržování různých systémů řízení kvality. [1]

1.3.2 Příklady systémů řízení kvality (QMS) v softwarových společnostech

S častějším využíváním testování a sledování kvality jako součásti vývoje, vznikla potřeba ucelených pravidel pro tuto činnost. Proto vznikly různé typy systémů řízení kvality (Quality Management Systems, QMS), které umožňují v organizaci rozpoznávání, měření a zlepšování různorodých procesů tak, že vedou ke zlepšení výkonu společnosti. V následujících kapitolách budou uvedeny některé z takových systémů. [26]

EN ISO 9000

ISO 9000 popisuje základní principy systémů managementu kvality a specifikuje terminologii systémů managementu kvality. [18]

EN ISO 9001

ISO 9001 specifikuje požadavky na systém managementu kvality pro případ, že organizace musí prokázat svoji schopnost poskytovat produkty, které splňují požadavky zákazníka a aplikovatelné požadavky předpisů a že má v úmyslu zvýšit spokojenost zákazníků. [18]

EN ISO 9003

Tato mezinárodní norma specifikuje, jaké požadavky na systém jakosti mají být použity v případech, kdy je třeba prokázat způsobilost dodavatele zjišťovat a řídit vypořádání každé

neshody výrobku v průběhu výstupní kontroly a zkoušení. Tato norma je v současné době neplatná. [19]

EN ISO 9004

ISO 9004 poskytuje směrnice, které berou v úvahu jak efektivnost, tak účinnost systémů managementu kvality. Cílem této normy je zlepšování výkonnosti organizace, spokojenosti zákazníků a jiných zainteresovaných stran. [18]

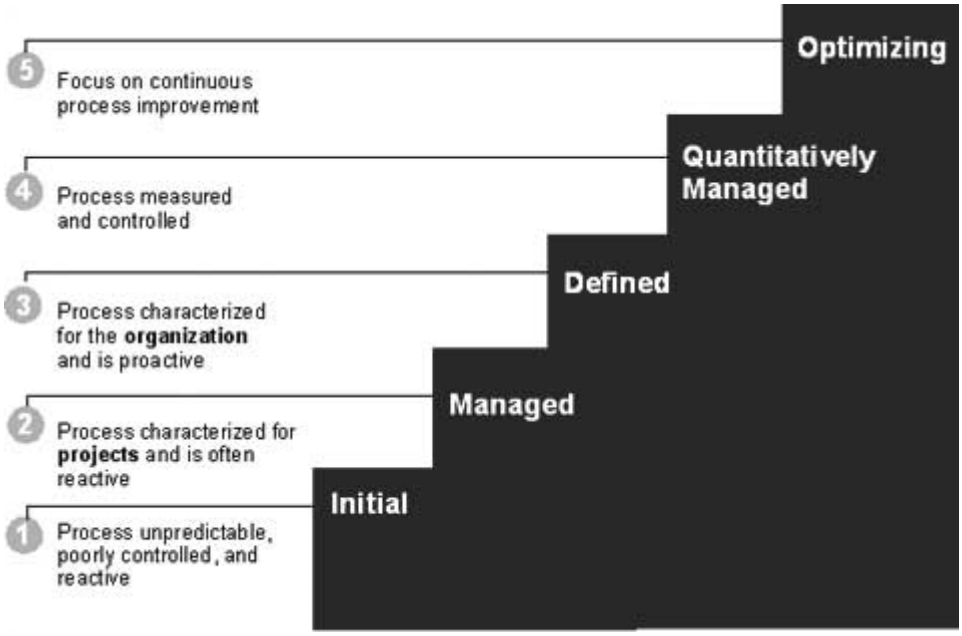
EN ISO 20000

Norma ISO/IEC 20000:2005 je první celosvětový standard, který se speciálně vztahuje k managementu služeb IT a zaměřuje se na zlepšování kvality, zvyšování efektivity a snížení nákladů u IT procesů. ISO 20000, které vzešlo ze standardu BS 15000, popisuje integrovanou sadu procesů řízení pro poskytování služeb IT [20]. Jedná se také o normu, která reaguje na potřebu zavedení českého ekvivalentu normy ITIL.

CMMI

Model zralosti a schopnosti (Capability Maturity Model Integration) se zabývá definicí, standardizací a postupným zlepšováním organizace, která se zabývá obecně vývojem, nejčastěji vývojem softwaru. [1]

Tento model je obecný a lze jej uplatnit v softwarové společnosti jakékoliv velikosti. Jeho pět úrovní (viz. Obrázek 2) představuje jednoduchý prostředek ohodnocení zralosti procesu vývoje softwaru v dané společnosti a určuje klíčové postupy, které je třeba přijmout pro přechod na další úroveň zralosti. [1]



Obrázek 2 – Popisy jednotlivých levelů modelu CMMI. [27]

1.4 Testovací dokumentace

Existují dva hlavní dokumenty, které spravuje test analytik popřípadě tester:

- *System Test Specification (STS)* – obsahuje kompletní dokumentaci k testování (jednotlivá pravidla, definice, na čem se má testovat, jednotlivé testovací scénáře apod.). Zjednodušeně říká „co a jak testovat“.
- *System Test Report (STR)* – slouží jako závěrečná zpráva po testování – (obsahuje tedy informace, co a jak bylo otestováno a především s jakými výsledky)

Navíc existuje ještě jeden dokument, ke kterému by měl mít test analytik i tester neustálý přístup a na jehož tvorbě (testování) by se měl zčásti i podílet:

- *System Requirements Specification (SRS)* – skládá se ze specifikace požadavků na vyvíjený systém (tedy jak má konečná aplikace vypadat)

2 ZPŮSOBY TESTOVÁNÍ

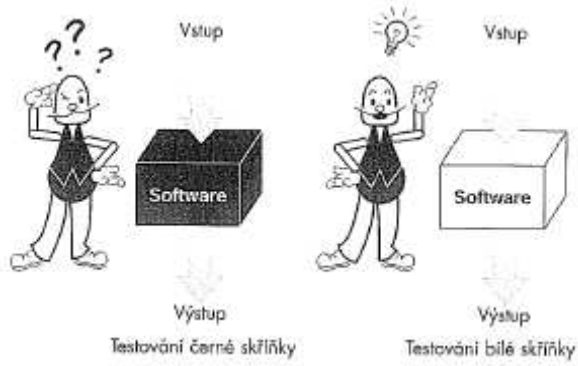
Aby mohl tester softwaru správně provádět svou práci, musí znát určitá pravidla. V následujících kapitolách jsou uvedena základní rozdělení přístupů k testování a s nimi spojené aktivity a nástroje.

2.1 Rozdělení přístupů k testování

Testy můžeme rozdělit do několika kategorií. Dále jsou rozebrány ty nejpoužívanější.

2.1.1 White box a Black box testování

Celkový postup při testování softwaru se charakterizuje jedním z následujících dvou pojmů: *testování černé skříňky* a *testování bílé skříňky*. Rozdíl mezi oběma pojetími ukazuje Obrázek 3. Při testování černé skříňky ví tester jen to, co má předložený software dělat – dovnitř skříňky se podívat nemůže a neví tedy, jak pracuje uvnitř. Jestliže napíše nějaký údaj na vstupu, dostane určitý odpovídající výsledek na výstupu. Neví, proč a jak se zrovna tento výsledek objevil, pouze jej pozoruje. [1]



Obrázek 3 – Testování černé a bílé skříňky [1]

U testování bílé skříňky má oproti tomu softwarový tester přístup ke zdrojovému kódu programu a jeho zkoumání mu může pomoci při testování – vidí tedy jakoby „dovnitř“ skříňky. Z tohoto pohledu pak tester může odhadnout, jestli určitá kombinace vstupů způsobí chybu a podle těchto informací může lépe přizpůsobit další testování.

2.1.2 Statické a dynamické testování

Při statickém testování se testuje něco, co neběží – to znamená, že zkoumaný objekt pouze prohlížíme a kontrolujeme (revidujeme). Statické testy (revize) jsou způsobem testování

softwarových pracovních produktů (včetně kódu) a mohou se vykonat mnohem dříve než dynamické testy. Defekty nalezené během revizí v časných fázích životního cyklu jsou často odstranitelné mnohem levněji než ty, které jsou nalezeny až během vykonání testů (např. defekty nalezené v požadavcích). [16]

Dynamické testování je nejčastějším typem testování – software je spuštěn a tester s ním tedy pracuje v dynamickém stavu. [1]

2.1.3 Funkční a nefunkční testování

Funkce, které systém, subsystém nebo komponenta má vykonávat, mohou být popsány v pracovních produktech, jako jsou specifikace požadavků, případy užití nebo funkční specifikace nebo mohou být nedokumentované. Funkce představují to, *co* systém dělá. [16]

Funkční testy jsou založeny na funkcích a vlastnostech (jak jsou popsány v dokumentech nebo chápány testery) a jejich spolupůsobení se specifickými systémy a mohou být vykonávány na všech úrovních testování (např. testy komponent mohou být založeny na specifikaci komponent). [16]

Nefunkční testování zahrnuje testování výkonu, zátěžové testování, stres testování, testování použitelnosti, testování udržovatelnosti, testování spolehlivosti a testování přenositelnosti, neomezuje se ale jen na ně. Je testováním toho, *jak* systém pracuje. Nefunkční testování může být vykonáváno na všech úrovních testování. Termín nefunkční testování popisuje testy vyžadované pro měření charakteristik systému a softwaru, které mohou být kvantifikovány vůči různým stupnicím měření, jako například doby odezvy pro testování výkonu. [16]

2.1.4 Regresní a konfirmační testování

Poté, co je defekt nalezen a opraven, by měl být software retestován za účelem potvrzení, že původní defekt byl úspěšně odstraněn. Takovéto testování se nazývá konfirmační testování. [16]

Regresní testování je opakované testování již testovaného programu po modifikaci s cílem najít všechny defekty, které mohly být zaneseny nebo objeveny jako důsledek jiné změny (změn). Tyto defekty se mohou nacházet v testovaném softwaru nebo v jiné související nebo nesouvisející softwarové komponentě. Regresní testování se provádí, když je změněn

software nebo jeho prostředí. Rozsah regresního testování je odvozen od rizika nenalezení defektů v softwaru, který předtím fungoval. [16]

Testy by měly být opakovatelné, pokud se mají používat v konfirmačním testovaní a pomáhat regresnímu testování. Regresní testování může být vykonáváno na všech úrovních testování a využívá se na funkcionální a nefunkcionální testy produktu. Sady regresních testů jsou spuštěny mnohokrát a všeobecně se vyvíjejí pomalu, proto je regresní testování silným kandidátem na automatizaci. [16]

2.2 Automatické testy a testovací nástroje

S postupem času byly a jsou vyvíjeny stále rozsáhlejší systémy a to má samozřejmě za následek i růst objemu testů. Při testování takových rozsáhlých systémů již není možno vystačit s obyčejným poznámkovým blokem nebo jednoduchou tabulkou - vznikají tak SW nástroje (např. Microsoft Test Manager či Testlink) a technologie (Selenium), které se testerovi snaží jeho práci jak ulehčit, tak zvýšit její efektivitu. Pro účely testování mobilních aplikací pro Android je možné využít např. Robotium (http://code.google.com/p/robotium), jenž se velmi podobá technologii Selenium, ale je určeno pro Android.

2.2.1 Vlastnosti automatických testovacích nástrojů

Nejdůležitější vlastnosti a výhody testovacích nástrojů a automatizace jsou:

- **Rychlost**
    - Zřejmě nejdůležitější vlastnost automatických nástrojů. Všechny úkoly jsou zpracovány strojově a bez prodlev. Je prokázáno, že při správně napsaných testech dokáže automatický nástroj urychlit testování až 15x. [1]
- **Efektivita**
    - Tato výhoda spočívá v tom, že testování přebere jakoby další tester (stroj). Fyzický tester se tak může věnovat úpravě stávajících testů či testováním testovacích případů, které nemohly být zautomatizovány. [1]

- **Správnost a přesnost**
    - Člověk je mnohdy ovlivněn stresem či únavou, které se mohou negativně projevit na jeho pozornosti při testování. Výhodou stroje oproti člověku je jeho neúnavnost. Všechny úkoly vykonává tak, jak jsou nadefinovány a ať jsou spuštěny kdykoliv, vždy je výsledek zpracován naprosto stejně a správně. [1]

2.2.2 Výhody a nevýhody automatických testů

Výhody automatizovaného testování jsou zejména tyto [21]:

+ Eliminuje vyšší náklady spojené s dodatečnou údržbou testů
+ Zvýšení produktivity (až 15x rychleší než manuální testování)
+ Zvýšení kvality (větší pokrytí, vyloučení chyb lidského faktoru)

Naopak nevýhody lze definovat následovně:

– Komplikace při rozpoznání objektů (stroj není schopen „vidět“ to, co člověk např. díky nemožnosti algoritmizace)
– Komplikace při synchronizaci (je stroj schopný vyčkat do vykreslení obrazovky?)
– Vhodné pouze pro regresní testování
– Vyšší počáteční časové náklady pro psaní automatických skriptů

2.3 Srovnání softwarového testování a testování mobilních aplikací

Ačkoli je testování aplikací pro mobilní zařízení velmi podobné klasickému testování počítačového software, tak v žádném případě není totožné. Existuje tedy několik, občas dokonce zásadních, rozdílů. První z následujících tabulek uvádí rozdíly mezi počítačovým software a aplikacemi pro mobilní zařízení. Druhá tabulka pak uvádí rozdíly v samotných testech.

Tabulka 1 – Srovnání SW testování a testování aplikací pro mobilní zařízení - činnosti

Testy	**Software pro PC**	**Mobilní aplikace reálné zařízení**	**Mobilní aplikace virtuální emulátor**
Ovládací prvky	**Klávesnice + Myš**	**Dotykový displej**	**Myš**
Tvorba screenshots	**Ano**	**Ne**	**Ano**
Podporované multimediální formáty	**Závislost na SW**	**Závislost na SW i OS**	
Rychlost připojení k síti	**Jen ve specifických případech, např. v kritických provozech**	**Proměnlivá dle síly signálu**	**Stabilní**
Spotřeba při spuštění aplikace	**Nesleduje se**	**Nutné sledovat**	**Nesleduje se**
Ovládací gesta	**Nepoužívají se**	**Možné testovat**	
Místo instalace	**Většinou pevný disk**	**Vnitřní paměť či paměťová karta**	
Nároky na výkon HW	**Výkon většinou přesahuje nároky aplikace**	**Nutné brát ohled na pomalejší procesory**	**Výkon většinou přesahuje nároky aplikace**
Rozdíly v OS	**OS dle verze**	**OS dle verze a výrobce zařízení**	
Externí zařízení	**Připojené k počítači**	**Součástí přístroje**	

Tabulka 2 - Srovnání SW testování a testování aplikací pro mobilní zařízení – typy testů

Testy	**Software pro PC**	**Mobilní aplikace reálné zařízení**	**Mobilní aplikace virtuální emulátor**
Funkční testy	**Ano**	**Ano**	
Nefunkční testy	**Ano**	**Ano**	
„White-box“ testy	**Ano**	**Ne**	
„Black-box“ testy	**Ano**	**Ano**	
Statické testy	**Ano**	**Ne**	
Dynamické testy	**Ano**	**Ano**	
Regresní testy	**Ano**	**Ano**	
Performance testy	**Ano**	**Ne**	
„Monkey“ testy	**Ne**	**Ne**	**Ano**
Manuální testy	**Ano**	**Ano**	
Automatické testy	**Ano**	**Ne**	

3 PROBLÉMY PŘI TESTOVÁNÍ

Následující text definuje a analyzuje nejčastější problémy a překážky, se kterými se tester může v praxi setkat.

3.1 Testovací data

Testovací data, převážně jejich vhodný výběr a kvalitní příprava, jsou jednou z nejdůležitějších částí tvorby testů. Vhodný výběr může přispět nejen ke zrychlení práce s testy (využití stejných dat pro kontrolu více funkcionalit testovaného programu), ale také zkvalitnit finální výsledky testů. Naopak jejich špatný výběr může způsobit řadu problémů, od neshody s produkčními daty až po narůstání objemu testů a tudíž i jejich následnou problematickou údržbu.

Testovací data samotná vyjadřují data, se kterými vyvíjená aplikace pracuje. Tato data se mohou vyskytovat na vstupu (nejčastější) nebo na výstupu.

3.1.1 Způsob získávání testovacích dat

V praxi se využívají převážně tři způsoby získávání dat.

Data od zákazníka

Zákazník nejlépe ví, kam přesně bude vyvíjený systém umístěn a jaká data bude zpracovávat – díky tomu má jasný přehled o tom, jaká data budou do systému vstupovat a jaké výsledky (výstupní data) budou ze systému vystupovat. Pokud je to tedy možné, je nejsnazší zákazníka požádat o reálný vzorek těchto dat a ty si případně dál upravit dle vytvořených testovacích případů (TCs - test cases). Takový přístup ušetří spoustu času, který bylo nutné vynaložit na manuální tvorbu dat a zároveň zajistí, že data se velmi podobají reálným datům, se kterými bude systém pracovat po jeho uvedení do provozu. Je tedy z uvedených možností nejvhodnější.

Generátor testovacích dat

Pokud není možnost získat data od zákazníka, je možné si vytvořit generátor testovacích dat, který vytváří umělá testovací data např. na základě specifikace. Data je nutné generovat tak, aby kvalitativně pokryla celé spektrum dat reálných (např. pomocí využití vzorkování, viz [28]). Tento způsob se může jevit pro testera

jako ideální, není však úplně ideální pro projekt - na tvorbu generátoru je totiž nutné alokovat nějakou pracovní sílu a spotřebovaný čas pak může chybět jinde, hrozí pak nedodržení termínů, vyšší náklady apod. Tento způsob je tedy vhodný pro dlouhodobější projekty či pro větší balíky testovacích dat.

Vlastní tvorba

Časově velmi náročný a chybově velmi náchylný způsob tvorby testovacích dat. Výhodu má snad jen v tom, že tester se při jejich tvorbě dokonalé seznámí s typem (formátem) těchto dat a tudíž s nimi dokáže v budoucnu rychleji a lépe pracovat.

3.1.2 Verifikace a validace testovacích dat

Při verifikaci se provádí kontrola, že postupujeme správně, tzn. zda používáme pro testování správná data. Toto je nutné provést ideálně ještě před započetím tvorby dalších testů či testováním samotným. Někdy bývá pojem verifikace nahrazen pojmem „kontrola správnosti“.

U validace se naopak snažíme potvrdit, že výstup testovacích dat odpovídá požadavkům, případně se jim maximálně přibližují a to v případě, že je tester testovací data nucen vytvářet uměle. Validaci lze provést matematickými postupy, typicky se také provádí konzultace se zákazníkem. Ideálním případem je tedy při zasílání náhledu testovacích scénářů poslat zákazníkovi i vzorek testovacích dat pro jejich validaci.

3.1.3 Typy a formáty

Testovací data můžou být nejrůznějších typů a formátů. Můžou existovat v některém ze známých formátu jako BMP či JPG (například pro obrázkové editory), TXT (textové prohlížeče), AVI či MP3 (audiovizuální nástroje) či XML (univerzální použití). Případně může být využit formát přesně na míru dané aplikace s kódováním například do hexadecimálního formátu.

1. **Statická data**

   Tato data se nacházejí ve statické (časově neměnné) formě a jsou fyzicky přítomna v testovaném zařízení (na pevném disku počítače či paměťové kartě mobilního telefonu) nebo můžou být umístěny na vzdáleném serveru, odkud jsou čteny

pomocí připojené sítě. Výhoda těchto dat je, že je lze připravit dopředu a využívat ve zvolený okamžik. Další výhodou je, že je lze dle libosti upravovat.

2. **Dynamická data**

   Dynamická data jsou měněna dynamicky v čase (většinou produkována systémy třetích stran) a jsou tedy velmi těžko predikovatelná. To velmi ztěžuje práci testera, protože ten pak není schopen přesně určit předpokládaný výstup programu ve zvolený okamžik a tím není také někdy schopen chybu zreprodukovat. Tomuto se dá předcházet tvorbou tzv. klonů (generátorů dat). Tyto klony pak produkují totožná data jako systémy třetích stran, ale jsou plně konfigurovatelné a tester tak může ovlivňovat jejich výstupy. Takový způsob ale stejně jako při ruční tvorbě testovacích dat zabere určitý čas, a proto je vhodný pouze při déletrvajících projektech, kde je taková snaha odůvodněná návratem takové investice (ROI – return of investments).

3.1.4 Nemožnost pokrytí veškerých dat

Jeden z velkých problémů u tzv. mass systémů je problém nemožnosti pokrytí kompletních dat. Jedná se o systémy, které zpracovávají obrovské množství nejrůznějších dat (např. burzovní nebo telekomunikační systémy). Zde je nutné volit optimální vzorek dat tak, abychom například v již zmíněném burzovním systému „zasáhli“ všechny obchodní skupiny, v případě jejich vysokého počtu pak ty nejpoužívanější. Je tedy důležité si zachovat jistou úroveň abstrakce, tedy udržet velikost testovacích dat na minimální možné úrovni při získání maximální efektivity. Proto je pozice test analytiků a lidí, kteří test data a testy samotné připravují, možná nejnáročnější vůbec.

3.2 Testovací prostředí

Stejně jako je důležité vhodně volit testovací data, je stejně důležité volit vhodné testovací prostředí a to v nejvyšší možné míře shodné s prostředím produkčním. S tímto je nutné počítat i v rámci finančních nákladů v rámci projektů (nákup dalšího HW či modelů mobilních zařízení, licence apod.). Důraz je nutné klást jak na kvalitu HW, tak i na jeho firmware (malá změna ve verzi může způsobit vyřazení instrukce z instrukčního setu a tím vyřadit z provozu celý systém). Na přesnou specifikaci cílového prostředí je nutno se zaměřit již v procesu definice požadavků anebo dokonce již při podepisování kontraktu.

Tester by měl mít pro svou práci opět k dispozici konfigurovatelné testovací prostředí. Díky konfigurovatelnosti je pak schopen simulovat například selhání či odpojení některého HW prvku či např. změnu firmware klíčového zařízení.

3.2.1 Způsob definování testovacího prostředí

Opět existuje několik způsobů, jak získat či vytvořit testovací prostředí.

1. **Zařízení zapůjčené zákazníkem**

   Pokud zákazník dodá/zapůjčí vlastní zařízení, či k nim povolí přístup, pak má tester jistotu reálných zařízení s předem definovaným firmware a konfigurací. Určitá nevýhoda může být ta, že například oproti virtualizaci se hůře simulují defekty a také se zařízení obvykle pomaleji dostává do původního nastavení/stavu. Jedná se však o nejvhodnější řešení.

2. **Zakoupení vlastních zařízení**

   Obvykle nejnákladnější možnost. Jednoduše spočívá v tom, že se pro projekt nakoupí zákazníkem předem definovaný hardware a software (včetně licencí) a ten se pak po celou dobu života projektu využívá dle potřeb týmu. Jedinou výhodou je to, že po skončení kontraktu se všechna tato zařízení dají využít v projektech dalších. Je zřejmé, že tento způsob je tedy vhodný pro projektové týmy, jež se specializují na určitý obor (například vývoj aplikací pro mobilní telefony běžící na OS Android).

3. **Virtualizace**

   V rámci úspor je pak možné vytvořit či zakoupit virtuální přístroje, které umožňují plnou konfiguraci prostředí na programové úrovni a jsou tak schopny simulovat různé podmínky a chyby v prostředí. Jejich velkou výhodou je také to, že se dají velmi rychle uvést do původního, tzv. čistého stavu a tím tak urychlit přechod na další iteraci testů.

3.2.2 Nemožnost pokrytí veškerého HW

Tester ani nikdo jiný nikdy nebude schopen zákazníkovi zaručit, že vytvořený software poběží na jakémkoli stroji za jakýchkoli podmínek. Kdykoliv se totiž v budoucnu může stát, že nastane porucha starší komponenty, která už se nebude vyrábět, a proto bude nutné

systém migrovat na úplně jiný HW, který se sice podobá původnímu, ale díky odlišnému výrobci může postrádat možnost některého nastavení. V případě volby testovacího prostředí je tedy nutné (stejně jako u testovacích dat), aby zvolené zařízení nijak výrazně nepřesahovalo náklady projektu a zároveň splňovalo všechny potřebné požadavky pro kvalitní otestování finálního produktu.

3.3 Požadavky od zákazníka

Požadavky od zákazníka (tzv. URD – user requirements definition) musí být zejména při vývoji na zakázku základní kámen celého projektu. Je velmi důležité v něm definovat tyto informace:

- **Přesné verze podporovaných prostředí (OS)** – v případě mobilních zařízení tedy například Android 2.3 či iOS 4.2.1
- **Typ použitých zařízení** – zde se jedná především o výrobce, model, verze firmware (jednotlivé modely se totiž mohou lišit jak v klávesách tak třeba v rozlišení a velikosti displeje a dalším HW)
- **Jaké externí zdroje budou využity** – Internet, GPS, senzor gyroskopu a další připojitelné zařízení
- **Specifikace objemu přenášených dat** – umožní optimalizaci aplikace pro vysokorychlostní sítě WiFi (jaké pásma), či pomalejší 3G nebo GPRS
- **Cílová země a jazyk** – týká se dvou problémů: jedním jsou rozdíly v HW a SW zařízení pro různé trhy a druhým jsou podporované jazykové balíčky a tím rozdílné uživatelské prostředí (UI – user interface) vyvíjené aplikace
- **Jakými způsoby bude aplikace distribuována** – FTP, web, USB, Bluetooth či jiné
- **Přesná specifikace požadavků** – nejlépe podložená grafickými návrhy a v jazyce, kterému je schopen porozumět každý člen týmu

3.4 Na co si zákazník často stěžuje

Následující text je zaměřen zejména na fázi získávání prvních požadavků od zákazníka a předprojektového plánování. Upozorňuje především na časté problémy, ke kterým je v této fázi potřeba velmi pozorně přihlížet.

- Mobilní telefony jsou silně závislé na signálu, a proto se internetové aplikace mohou často potýkat se **zamrzáváním či přerušením stahování**
- Problémy s **časovými limity u ověřování kreditních karet** se vyskytují u bankovních mobilních aplikací
- **Rychlost aplikace během slabého signálu** připojení k internetu či mobilní síti
- **Nekonzistentní tlačítka, fonty, nadměrná „barevnost“**
- **Chybějící nebo poškozené linky** – mobilní provedení funkce copy+paste je na rozdíl od PC obtížněji proveditelná
- **Aplikace není podporována zvoleným zařízením**
- **Zastaralé verze** – a tudíž staré chyby v aplikaci
- **Nemožnost blíže upřesnit nalezené problémy** – zákazník obvykle nemá možnost zasílat jakékoli screenshots či logy nalezených chyb v mobilním zařízení

3.5 Nevyslovené požadavky

Největším problémem při definici požadavků na produkt jsou tzv. zažitá pravidla, tedy něco, o čem se předpokládá, že to každý zná. Příkladem mohou být potvrzovací dialogová okna v MS Windows - v drtivé většině dokumentací projektů se dočtete, že po vykonání určité operace se má zobrazit potvrzovací okno s možnostmi „OK“ a „Cancel“, případně dokumentace obsahuje i funkcionalitu jednotlivých tlačítek. Absolutní většina uživatelů je zvyklá na to, že tlačítko „OK“ je vždy vlevo, zatímco tlačítko „Cancel“ vždy vpravo. Je to standard, který ale není nikde definován, tedy nevyslovený požadavek. Je nezbytně nutné, aby tyto standardy byly známé také vývojovému týmu. Proto je doporučeno dokumentaci doplňovat ideálně o grafické náhledy všech dostupných oken, případně uvádět i jejich rozměry či barevný odstín. Zároveň je ideální uvádět průměrnou požadovanou dobu reakce

aplikace při specifických úkonech. Vývojový tým se díky tomu pak vyvaruje budoucí dodatečné úpravě kódu a dokumentace.

Nevyslovené požadavky bývají velkým problémem převážně u společností/zákazníků, kteří s vývojem nemají příliš zkušeností. Takové společnosti se typicky vyznačují tím, že jsou chaotické, nemají žádný systém řízení kvality, jsou příliš malé, špatně organizované či mají nejasné kompetence jednotlivých zaměstanců. Jejich požadavky pak mají tendenci se často měnit a mají nepřesnou formu.

3.6 Uchování testů

Všechny pracně vytvořené testy je samozřejmě nutné někde uchovávat. V dnešní době, kdy se vyvíjejí rozsáhlé informační systémy, již není možné uchovávat testy v papírové formě. Ke slovu pak přicházejí různé programy pro podporu ukládání testovací dokumentace. Tyto nástroje jsou většinou připojeny k databázi testů a umožňují tak přes uživatelské rozhraní jejich správu. Ty nejzákladnější nástroje (většinou také jako freeware) umožňují právě takové uchování a editaci. Ty robustnější umožní testy nejen spravovat, ale také spouštět, reportovat chyby či dokonce tvořit plně automatizované testy.

Takových nástrojů je velké množství, v následujících kapitolách se proto zmíním o těch nejfrekventovanějších

3.6.1 TestLink

TestLink je jeden z webových nástrojů pro uchování, správu a spouštění testů. Jako jeden z mála je bezplatně dostupný a jeho vývoj je financován výhradně z dobrovolných příspěvků uživatelů. Existuje ve formě webové aplikace, a tudíž nabízí i možnost tvorby a správy přístupových účtu jednotlivých testerů, jejich alokace na projektu a správu projektů samotných. Každý uživatel pak může dle svých práv v rámci přiřazeného projektu spravovat požadavky od zákazníka, vytvářet test cases a test suites a z nich následně tvořit kompletní test plány – samozřejmě včetně verzování. Pro připravené testy pak TestLink umožňuje vyplnění výsledků spuštěných testů, které ukládá a následně připraví nejrůznější metriky a test reporty. [22]

3.6.2 Microsoft Test Manager

Test Manager (někdy též Test Professional) je distribuován jako doplněk k Microsoft Visual Studio. Nástroj spolupracuje jak s balíkem TFS (Team Foundation Server), tak s Visual Studiem. Kromě klasického managementu testů umí Test Manager i testy spouštět a to jak manuálně, tak dokonce i automatizovaně. Automatizace spočívá v tom, že při prvním spuštění manuálních testů nástroj nabídne možnost nahrání testerovy práce. Při každém dalším spuštění pak Test Manager „nakliká“ většinu test kroků sám. [23]

Jednou z nevýhod tohoto nástroje (například oproti TestLinku) je jeho vysoká HW náročnost, a proto se nehodí pro firmy se starším HW. Další nevýhodou je velmi špatná dostupnost informací a nepřehledná nápověda. Uplatnění však nalezne ve větších společnostech, kde projektové týmy využívají různých SW balíků firmy Microsoft. [23]

3.6.3 HP QuickTest Professional (QTP)

Tento nástroj poskytuje možnost automatizace funkčních a regresních testů pro nejrůznější aplikace a prostředí. Je součástí balíku HP Quality Center, který je určen pro zajišťování kvality ve společnostech. [29]

QTP podporuje klíčová slova a tvorbu skriptů pro testování SW na uživatelské i programové úrovni. Ke specifikaci testovacího procesu a k manipulaci s jednotlivými objekty a kontrolními prvky v aplikaci využívá skriptovací edici Visual Basic (VBScript). [29]

3.6.4 Borland SilkTest

SilkTest je dalším z nástrojů pro automatizaci funkčního a regresního testování. Je určen také pro business analytiky, test analytiky i vývojáře. [24]

Kromě testování klasického uživatelského rozhraní jeho prostředí podporuje tvorbu skriptů, které mohou být využity pro opakované testování aplikací ve většině z dostupných webových prohlížečů. Umožňuje také rychlé spouštění testů. [24]

3.6.5 IBM Rational Robot

Rational Robot je automatizační tool pro funkční testování klient/server aplikací. Je určen kompletním QA týmům, kterým umožňuje detekování chyb, správu testovacích případů i

celého testovacího procesu. Jeho výhodou je také podpora více UI (user interface) technologií. [25]

3.6.6 Ostatní nástroje

Z dalších nástrojů pak například: [30]

- TestComplete (SmartBear Software)
- Parasoft SOAtest (Parasoft)
- Ranorex (Ranorex GmbH)
- Selenium (Open source)
- WATIR (Open source)

3.7 Problémy při testování mobilních aplikací

V následujících několika kapitolách budou uvedeny kritické body a procesy, kterých je nutné si všímat při navrhování, vývoji a testování mobilních aplikací. Tyto údaje jsou získány převážně z praktických zkušeností.

3.7.1 Problémy při testování funkčních požadavků

Zde je uvedeno několik bodů, ke kterým je třeba přihlížet obzvláště během testování aplikací pro mobilní zařízení.

- **Ovládání klávesnicí, prstem či pointerem** – nutno vyzkoušet všechny možnosti
- **Gesta**[1] – především u aplikací pro iOS a Android
- **Využití simulátorů** – simulátory jsou vhodné pro rané testování či při nedostupnosti fyzického zařízení. Zároveň jsou schopny provádět stress testy či monkey testy (viz. níže)

[1] Gestem se zde rozumí specifický pohyb prstu po dotykovém displeji mobilního zařízení. Tato gesta mají často i svá označení (Slide, Tap, apod.)

  - **Stress a „monkey“ testy** – monkey test spočívá v chaotickém „klikání“ do různých oblastí obrazovky – může odhalit skryté slabiny UI
  - **Benchmarking** – aneb testování výkonu a měření množství paměti využívané aplikací (včetně doby načítání jednotlivých obrazovek)

- **Propojení mezi aplikacemi** – test komunikace a předávání dat mezi aplikacemi (nutno otestovat, co se stane, pokud druhá aplikace chybí, je poškozena či existuje v jiné verzi – např. rozmanitost multimediálních přehrávačů v mobilních zařízeních)
- **Změna rozlišení a orientace** – mnohá zařízení dnes během změny jejich polohy podporují dynamické otáčení displeje – je nutné zjistit, zda je obrazovka překreslena správně
- **Dynamické změny konfigurace** – změny v dostupnosti klávesnice, změna času či jazyka
- **Závislost na externích zdrojích** – pokud je aplikace závislá na přístupu do sítě, SMS, gyroskopu či datech z GPS, je nutné otestovat, zda se dokáže vypořádat s nedostupností těchto dat (tedy například pokud je jejich externí zdroj vypnut)
- **Závislost na internetovém připojení a jeho rychlosti** – tedy například změna připojení z rychlého na pomalé (WiFi na GPRS)
- **Podporované formáty** – pokud se aplikace zabývá např. multimediálními formáty, je třeba ověřit dostupné formáty v rámci platformy/OS
- **Crowdsourced aneb davové testování** – jestliže je aplikace určena pro široké masy, je vhodné zajistit patřičný vzorek testerů pro daný projekt (přínosné především pro testování UI)
- **Zjistit možnosti pro reportování chyb** – jakým způsobem budou tvořeny logy či screenshots z aplikace
- **Možnosti instalace/odinstalace aplikace a její ukončování/spouštění** – test, zda je aplikace vždy správně ukončena či smazána
- **Verze firmware** – testy pro všechny požadovaná zařízení a jejich konfigurace

- **Přerušení běhu aplikace** – například přesunutím na pozadí pomocí uživatelovy aktivity či během příchozího hovoru, SMS, slabá baterie, připojení/vypojení baterie
- **Využití skrytých údajů o telefonu** – může se hodit např. při performance testování (u OS Android je takové menu dostupné pomocí vytočení „*#*#4636#*#*“)
- **Look and Feel** – závisí především na zkušenostech testera a jeho znalosti dané platformy – je nutné zajistit, aby aplikace fungovala (a dala se ovládat) pomocí uživatelsky zažitých tradic (tedy např. gestikulace prstů, velikost klikatelných tlačítek, čitelnost údajů, způsoby prohlížení galerií či vzhled menu)

3.7.2 Problémy při testování nefunkčních požadavků

V této kapitole je uvedeno několik praktických rad určených pro vývojáře (nejen však pro ně), které jim můžou být užitečné a především jim mohou pomoci předejít negativní kritice jejich práce.

- **Operační a úložná paměť** – mobilní zařízení mají limitovanou velikost operační a úložné paměti, při testování aplikace je tedy nutné sledovat i tuto veličinu
- **Výkon** – starší telefony mají pomalejší procesory, pokud se vyvíjí aplikace i pro ně, mělo by s tím být počítáno při optimalizaci (např. Android 3.0 podporuje více procesorů, starší verze nikoli), tester tedy musí sledovat, jak moc se aplikace promítá do aktuálního výkonu daného zařízení
- **Kompatibilita** – je nutné zjistit informace o všech zařízeních, pro které je aplikace určena (pro OS Android jsou tyto informace specifikovány v manifest-file aplikace)
- **Spotřeba** – potřeba testů pro měření spotřeby baterie
- **Podpora zařízení s rozdílným rozlišením či více displeji** – jedná se o nový trend na poli mobilních zařízení, se kterým je nezbytné počítat
- **Neinstalovat aplikaci na externí úložiště** – taková instalace může způsobit problémy při přepnutí do režimu „USB mass storage“ během běžící aplikace

- **UnitTesty**[2] – specializace pro každý OS (např. JUnit3 pro Android)

3.7.3 Srovnávací tabulka výskytů a závažností

Následující tabulka uvádí pro jednotlivé body z předcházejících kapitol jejich závažnost (tj. jak moc mohou ovlivnit úspěch projektu) a výskyt (tj. jak často se problém vyskytuje). Na konci kapitoly je pak uvedena tabulka s vysvětlením hodnot pro tyto pojmy. Poslední sloupec tabulky pak uvádí součet hodnot přecházejících obou hodnot (čím vyšší je hodnota, tím větší riziko daný problém představuje).

Tabulka 3 – Závažnost a výskyt problémů při testování funkčních požadavků

Testování funkčních požadavků			
Problém	**Závažnost**	**Výskyt**	**SOUČIN**
„Look and Feel“ problémy	3	3	**9**
Aplikace je pomalá	2	3	**6**
Pomalá komunikace po síti	2	3	**6**
Chyby v komunikaci s HW zařízení (např. GPS)	3	2	**6**
Chyby při instalaci/odinstalaci	3	2	**6**
Problémy s ovládáním (prsty, klávesnice, pointer)	3	1	**3**
Problémy při přerušení běhu aplikace	2	3	**5**
Špatné zobrazování na displeji s odlišnými parametry, špatné překreslení při otočení zařízení	2	2	**4**
Chyby v komunikaci mezi aplikacemi	2	2	**4**
Chyby při změně konfigurace zařízení	2	2	**4**
Nepodporované formáty	2	1	**2**
Špatné rozpoznání gest	1	1	**1**
Selhání v „monkey“ testu	1	1	**1**

[2] JUnit je testovací framework psaný v jazyce Java. Tento framework slouží k tvorbě jednotkových testů za účelem otestování zdrojového kódu programu.

Tabulka 4 – Závažnost a výskyt problémů při testování nefunkčních požadavků

Testování nefunkčních požadavků			
Problém	**Závažnost**	**Výskyt**	**SOUČIN**
Nedostatek volné paměti RAM	3	3	**9**
Problémy s kompatibilitou (různí výrobci zařízení)	2	3	**6**
Vysoká spotřeba při běhu aplikace	2	2	**4**
Nedostatečný výkon procesoru zařízení	2	2	**4**
Nedostatek místa ve vnitřní paměti zařízení	2	1	**2**
Problémy s vykreslování na více displejů zároveň	2	1	**2**
Špatné umístění instalace (paměťová karta)	1	1	**1**

Následující tabulka uvádí vysvětlení obou použitých sloupců.

Tabulka 5 – Vysvětlení značení

Závažnost		
Hodnota	**Definice**	**Příklad dopadu**
3	VYSOKÁ	Pád aplikace, zamrznutí telefonu, neakceptace zákazníka
2	STŘEDNÍ	Problém s funkčností, který však lze obejít alternativním postupem
1	NÍZKÁ	Drobný problém, kosmetická chyba
Výskyt		
Hodnota	**Definice**	
3	Velmi často	
2	Často	
1	Zřídka	

4 OPERAČNÍ SYSTÉMY MOBILNÍCH TELEFONŮ

4.1 Android od firmy Google

Obrázek 4 – Android [3]

Google Android je jedním z nejmladších operačních systémů pro mobilní zařízení („chytré“ telefony, PDA, navigace). I přesto je však v současné době jedním z nejpoužívanějších OS pro mobilní zařízení a dá se říct, že jediným větším konkurentem je pro něj iOS (systém běžící na mobilních zařízeních od firmy Apple).

4.1.1 Historie a základní informace

Jak již jeho úplný název napovídá, jedná se o systém od firmy Google. Jde o OS na linuxovém jádře, má však kořeny sahající mimo Google – pochází totiž z dílny firmy Android Inc, kterou v roce 2005 převzal právě Google. Ten pak veškeré zdrojové kódy předal sdružení firem Open Handset Alliance (uskupení původně 34 výrobců mobilních telefonů, telekomunikačních operátorů a technologických firem, které prosazují používání otevřených standardů na poli mobilních zařízení). Za vznikem tohoto sdružení stojí též sám Google, který také financoval odměny pro autory prvních aplikací pro tento OS (např. v rámci soutěže Android Developer Challenge). [5]

Vývoj aplikací se provádí v jazyce Java a za pomoci speciálních knihoven od Google (to vše v rámci vývojářského balíku Android SDK, které je zároveň plně podporováno vývojovým prostředím Eclipse včetně Android simulátoru). Android však Javu nativně nepodporuje. [5]

Oficiální ohlášení platformy proběhlo 5. listopadu 2007 a od počátku roku 2008 byla k dispozici první veřejně dostupná verze platformy a to jak pod licencí Apache, tak pod

licencí GPLv2 (jedná se tedy o open-source software). Android 1.0 včetně vývojového prostředí pak spatřil světlo světa 28. září 2008. [5]

4.1.2 Distribuce - Android Market

Jedná se o aplikaci v každém telefonu s OS Android, která skrze internetové připojení zařízení slouží k distribuci aplikací pro tento operační systém. Uživatel si tak může skrze něj stahovat do mobilu nejrůznější aplikace, které jsou pro přehlednost rozděleny do patřičných kategorií. V současné době je na něm k dispozici přes 150 000 aplikací a to jak placených, tak bezplatných. [5]

Díky open-source licenci je však možné instalovat aplikace i pomocí zkopírování na paměťovou kartu a následné instalace pomocí některého za správců souborů telefonu. [5]

4.1.3 Upgrade a downgrade systému

Upgrade je možný, pokud je zařízení kompatibilní s novým systémem. Záleží tedy pouze na výrobci, zda a kdy nový systém pro své zařízení zpřístupní. S downgradem je to bohužel horší, protože jsou potřeba pokročilé znalosti systému a v neposlední řadě záleží i na možnostech daného mobilního zařízení.

Seznam podporovaných zařízení lze nalézt zde [6].

4.2 iOS od firmy Apple

Obrázek 5 – iOS je využíván na většině mobilních zařízení od Apple [4]

Operační systém iOS pochází od firmy Apple, která jej vyvíjí primárně pro svá mobilní zařízení typu iPhone, iPad či iPod.

4.2.1 Historie a základní informace

Historie tohoto operačního systému se začala psát 9. ledna 2007, kdy byl vydán první iPhone. Tehdy ještě známý pod názvem iPhone OS. Název iOS se začal používat až od čtvrté verze tohoto operačního systému [7]. iOS je odnoží klasického MacOS X a je tedy postaven na UNIXovém systému, ke kterému přidává multi-touch podporu (podpora ovládání více prsty).

Původně tento systém nepodporoval žádné z aplikací třetích stran, a proto vůbec první oznámení o vývoji SDK přišlo až ke konci roku 2007. Vývoj aplikací mimo Apple začal až 6. března 2008,což je datum vydání SDK (software development kit) pro iOS. Vývojáři si tak mohli vytvářet a ladit aplikace na iPhone simulátoru, který je součástí SDK. [7]

iOS však zůstal i pak poněkud uzavřeným systémem a je možné do něj instalovat jen „ověřené“ aplikace. Pro získání této „ověřovací“ licence je nutné se přidat do iPhone Developers Programu a samozřejmě zaplatit poplatek. Apple se tím údajně snaží bránit psaní neoptimalizovaných aplikací. [7]

Vývoj probíhá v jazyce Objective-C ve vývojovém prostředí Xcode od Apple. Zároveň toto IDE běží pouze na počítačích Mac s procesory Intel. [7]

Jak již bylo zmíněno, přejmenování na iOS přišlo v červnu roku 2010. Tato zkratka zároveň pobouřila společnost Cisco Systems, jenž název IOS používala u SW, který byl instalován na jejích zařízeních. [7]

4.2.2 Distribuce - App Store

Každému vývojáři z iPhone Developers Programu je umožněno zveřejnit svou aplikaci skrze App Store (podobný Android Marketu) a také si nastavit cenu, za stažení této aplikace. Z každé prodané licence pak získá 70% podíl (zbytek připadá společnosti Apple). V současné době je v App Store k dispozici něco přes 300 000 aplikací (což je zhruba dvakrát tolik než u Androidu). [7]

Na iOS nelze oficiálně instalovat jiné než ověřené aplikace. Neoficiální cesta samozřejmě existuje a to pomocí tzv. jailbreaku. Jedná se o instalaci speciálního exploitu, který daný přístroj odemkne a uživatel pak může instalovat veškeré aplikace. Tyto exploity jsou pak většinou zveřejňovány různými hackerskými skupinami většinou několik týdnů až měsíců

po vydání nové verze iOS. Apple se však brání vydáváním nových systémů a zároveň neuznáváním reklamací takto odemknutých přístrojů. [7][8]

4.2.3 Upgrade a downgrade systému

Upgrade systému se děje většinou automaticky skrze připojení k počítači s iTunes (multimediální program pro správu zařízení firmy Apple) a Internetem. Ten při zjištění nové verze iOS na ni sám upozorní, případně stáhne a nainstaluje do připojeného zařízení. [8]

Pokud jde o downgrade, tak nastává problém. Je nutné totiž mít IPSW (jedná se o jakýsi popis předchozích iOS). Problémem je, že je nutné mít tato data přesně pro konkrétní model (IPSW je tedy vázáno na sériové číslo telefonu). A i pokud tato záložní data jsou k dispozici, tak je downgrade ne zrovna snadnou záležitostí (některé společnosti pak mají pro účely vývoje a testování hned několik iPhonů s různými verzemi iOS a přísným zákazem jejich upgradu). [8]

Neoficiální seznam podporovaných zařízení (samozřejmě pouze od firmy Apple) je zveřejněn na stránkách Wikipedie [9]

4.3 Ostatní

Kromě již zmíněných nejrozšířenějších operačních systému se samozřejmě používají i další, které přímo či nepřímo podporují někteří výrobci mobilních zařízení.

4.3.1 Windows Phone (Windows Mobile)

Windows Mobile byl jeden z prvních systému právě pro „chytré telefony“, ale v současnosti (a ve verzi Windows Phone 7) však tento mobilní systém stojí spíše ve stínu ostatních. Používá se stále především v USA. nicméně slibnější budoucnost se mu nabízí ve formě smlouvy s firmou Nokia - pokud by totiž k této dohodě skutečně došlo, tak by tato finská společnost mohla tento operační systém začít prosazovat u většiny svých zařízení (což by znamenalo zřejmě také zánik Symbianu, viz dále). [10]

4.3.2 BlackBerry

O tomto systému se v České republice příliš neví, důvodem je zřejmě malé rozšíření těchto smartphonů. Doménou těchto zařízení je především severoamerický trh, kde zaujímá přední místa v popularitě spolu se zařízeními pro Windows Mobile. Autorem tohoto systému je kanadská společnost RIM (Research in Motion). Mobilní zařízení BlackBerry si vybírají především uživatelé z oblasti managementu a to z důvodu velmi dobrého zabezpečení, kterým toto zařízení disponuje. Aby byl totiž BlackBerry využit na maximum, je nutný dodatečný SW na některém ze vzdálených serverů, který umožňuje propojení jednotlivých zařízení například v rámci společnosti (funguje jako intranet). [11][12]

4.3.3 Symbian

Posledním jmenovaným OS pro mobilní zařízení je Symbian. Jeden z nejstarších a možná i nejznámějších operačních systémů, který však po obrovském boomu Google Android upadl téměř v zapomnění (ačkoli je, stejně jako Android, postaven na svobodné licenci). I nadále však probíhá jeho vývoj - v současné době je k dispozici Symbian OS 9.4 a to především na mobilních telefonech značky Nokia. Nevýhoda tohoto systému spočívá v časté nekompatibilitě se staršími verzemi nebo také v omezení převážně na procesory ARM. [13]

II. PRAKTICKÁ ČÁST

5 TESTING SKELETON

Pojem Testing skeleton by se dal do češtiny volně přeložit jako „kostra testování“. Jedná se tedy o jakýsi manuál či popis toho, jak by měl tester/test analytik postupovat při své práci (tedy zejména přípravě, spouštění a vyhodnocení testů) na projektu. Následující kapitoly by měly sloužit právě jako takový testing skeleton (v tomto případě má však svá specifika, která jsou zaměřena primárně na testování mobilních aplikací). Postupy a myšlenky z tohoto manuálu však může využít například i tester webových či desktopových aplikací.

Následující podkapitoly slouží jako jednotlivé procesy testing skeletonu. Součástí každé podkapitoly/procesu bude vysvětlení kroků, důvod jejich zavedení a rizika, která s sebou přináší při nedodržení. V každé podkapitole je zároveň uveden výstup neboli výsledek po ukončení daného procesu. Náročnost je odhadována jako procento z celkové alokace testera na daném projektu. Jednotlivé kroky procesu jsou pro lepší orientaci číslovány (sekundární kroky, tedy nepovinné, jsou na začátku označeny hvězdičkou).

5.1 Projektová příprava

Popis: Na počátku každého projektu by se měl tester seznámit s projektem. Mělo by jej zajímat pro koho projekt, použité technologie, složení týmu a také jazyk pro komunikaci s klientem.

Výstup: Tester by měl získat základní pojem o připravovaném projektu a především se rozhodnout (případně přispět k rozhodování), zda je pro něj vhodným kandidátem (dostatečné znalosti).

Náročnost: cca 1-3%

Tabulka 6 – Popis procesu Projektové přípravy

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
1.1	**Seznámit se s cílem projektu (obor, pojmy)**	**Získání povědomí o projektu**	**Možné nepochopení pojmů z projektu, špatně definované testy**
1.2	**Pochopit použité technologie**	**Nutnost alespoň pasivní znalosti použitých technologií**	**Nepochopení základních principů a možností použitých technologií**
1.3	**Zjistit jazyk pro komunikaci s klientem**	**Důležité pro správné pochopení vstupů od klienta**	**Obtížná, pomalá a nepřesná komunikace s klientem**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***1.1**	**Zjistit bližší informace o klientovi**	**Být proaktivní (snaha identifikovat klientovy požadavky, a to dříve než jsou skutečně vyřčeny)**	
***1.2**	**Zvážit crowdsourced testování**	**Vhodné pokud je aplikace určena pro široké masy (především pro testování UI)**	

5.2 Seznámení se s týmem

Popis: Cílem tohoto procesu je základní seznámení s týmem (tedy vývojáři, designery a vedoucím týmu nebo projektovým manažerem). To zahrnuje definici způsobu komunikace a specifikaci co vyžaduje tým od testera a naopak.

Výstup: V tomto případě by měl tester získat přehled o kompetencích jednotlivých členů týmu a sjednotit se s týmem jak v komunikaci, tak v nastavení jednotlivých projektových nástrojů.

Náročnost: cca 5-7%

Tabulka 7 – Popis procesu Seznámení se s týmem

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
2.1	**Definice kompetencí jednotlivých členů týmu**	**Dotazy jsou směřovány ke správné osobě**	**Zmatená komunikace napříč celým týmem – zdržení.**
2.2	**Zpřístupnění projektových adresářů**	**Přístup k projektové dokumentaci**	**Dodatečné vyřizování - zdržení**
2.3	**Začlenění do projektové e-mail skupiny (+ vytvoření pravidel pro užívání e-mailu, např. v MS Outlook)**	**Pro zrychlení a zpřehlednění komunikace v rámci projektu**	**Nekonzistentní informovanost týmu**
2.4	**Přístup do bugsystému (Mantis, Bugzilla, TFS)**	**Přístup k reportovacímu systému**	**Dodatečné vyřizování - zdržení**
2.5	**Centralizace projektových úkolů (office documents, TFS, XPlanner)**	**Pro zpřehlednění prací na projektu**	**Chaotické řízení a nepřehlednost v projektových úkolech**
2.6	**Instalace ostatních projektových prostředí**	**Přístup ke všem informacím a začlenění do projektu**	**Pozdější instalace špatných verzí, pomalá práce s neznámými nástroji**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***2.1**	**Žádost o informování o nové verzi a místu jejího uložení (pomocí email/skype)**	**Zjednodušení komunikace a urychlení práce**	
***2.2**	**Žádost o umístění verzování přímo do aplikace (název, verze, datum)**	**Zpřehlednění aktuálně nasazených verzí**	

***2.3**	**Žádost o zveřejňování úprav v každé vydané verzi (i interní)**	**Objektivnější psaní testů a testování**
2.4	**Při opravách bugů by měli vývojáři definovat, ve které verzi se oprava vyskytne**	**Retestování jen skutečně opravených bugů**
2.5	**Zavedení tvorby logu přímo v mobilním zařízení**	**Kvalitnější údaje při reportování chyby**
2.6	**Zavedení možnosti vytváření screenshots přímo v aplikaci**	**Kvalitnější údaje při reportování chyby**

5.3 Analýza požadavků a SRS

Popis: V tomto procesu má tester za úkol si nastudovat specifikaci systému (SRS), analyzovat zákazníkovy požadavky a případně specifikovat nedostatky, které musejí být v SRS doplněny či opraveny.

Výstup: Tester získá kompletní znalosti o vyvíjeném systému a případně doplní či opraví patřičnou dokumentaci.

Náročnost: cca 5-7%

Tabulka 8 – Popis procesu Analýza požadavků a SRS

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
3.1	**Nastudování URD a SRS**	**Získání všech potřebných informací o vyvíjeném systému**	**Tvorba špatných testů, nepochopení některých pojmů**
3.2	**Analýza URD a SRS (viz. Příloha III)**	**Případné doplnění všech potřebných údajů do SRS**	**Vážné odlišnosti mezi produktem a zákazníkovými požadavky**
3.4	**Založení STS (tvorba šablony dokumentu)**	**Slouží k dokumentaci testerovy práce**	**Chaotická nebo nulová testerská dokumentace**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	

***3.1**	**Rezervace/zapůjčení cílových mobilních zařízení**	**Seznámení se s jejich funkcemi a prací s nimi ještě před začátkem vývoje**
***3.2**	**Brainstorming – nechybí něco?**	**Možné doplnění a následná lepší efektivita práce**

5.4 Definice testovacího prostředí

Popis: Tento proces se zabývá instalací a nastavením testovacího prostředí. Jedná se nejen o počítače, ale také o rezervaci a přípravu mobilních zařízení.

Výstup: Po tomto procesu by měl mít tester k dispozici plně funkční a nastavená testovací zařízení i počítač, což mu bude velmi užitečné jak při tvorbě a ladění testů tak při testování samotném.

Náročnost: cca 3-5%

Tabulka 9 – Popis procesu Definice testovacího prostředí

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
4.1	**Identifikace potřebného HW a SW (včetně licencí) z SRS**	**Získání seznamu potřebných prostředků**	**Nedostatek financí na dodatečné získání opomenutých prostředků**
4.2	**Zajištění těchto prostředků – Odblokované telefony, SIM karty (přesné konfigurace)**	**Tester musí mít veškeré prostředky okamžitě k dispozici**	**Při nedostatku zařízení se může práce prodlužovat**
4.3	**Obstarání uživatelských manuálů ke všem zařízením**	**Pro snadnější rešení problémů s nastavením a ovládáním**	**Dodatečné vyhledávání může zabrat čas navíc**
4.4	**Instalace a nastavení SW (virtuální simulátor)**	**Snaha o co nejvěrnější napodobení cílového prostředí**	**Nemožnost sjíždět benchmark testy**
4.5	**Nastavení HW (dle SRS)**	**Snaha o co nejvěrnější napodobení cílového prostředí**	**Nedokonalé otestování**

4.6	**Zavedení okolního prostředí**	**Některé aplikace mohou být využívany potmě, při hluku či otřesech**	**Nedokonalé otestování**
4.7	**Doplnění údajů do STS**	**Slouží k dokumentaci testerovy práce**	**Chaotická nebo nulová testerská dokumentace**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***4.1**	**Vyhodnocení nutnosti tzv. field testu**	**Testování v reálných podmínkách, které jsou laboratorně nedosažitelné**	
***4.2**	**Tvorba/doplnění manuálů**	**Mohou pomoci nově příchozím testerům**	

5.5 Definice testovacích dat

Popis: Je nutné před samotným testováním získat specifikaci testovacích dat. Tento proces se zabývá jejich specifikací, získáním (zákazník či vlastní generátor) a vybráním kvalitního vzorku pro testy.

Výstup: Tester získá výchozí vzorek testovacích dat, ze kterého pak bude vycházet při tvorbě testů i při testech samotných.

Náročnost: cca 3-5%

Tabulka 10 – Popis procesu Definice testovacích dat

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
5.1	**Určení typu dat – dynamická/statická data (SRS/zákazník)**	**V případě dynamických dat nutné konfigurovat servery a přístupy**	**Chybná simulace vstupních dat**
5.2	**Identifikace potřebného vzorku testovacích dat (SRS/zákazník)**	**Získání seznamu potřebných prostředků**	**Nedostatek financí na dodatečné získání opomenutých prostředků**
5.3	**Zajištění testovacích dat (zákazník)**	**Tester musí mít veškeré prostředky okamžitě k dispozici**	**Zdržení při zajišťování až ve fázi příprav testů**

5.4	**Doplnění údajů do STS**	**Slouží k dokumentaci testerovy práce**	**Chaotická nebo nulová testerská dokumentace**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***5.1**	**Tvorba vlastního generátoru dat**	**Urychlení přípravy dat pro jednotlivé testy**	
***5.2**	**Tvorba/doplnění manuálů**	**Mohou pomoci nově příchozím testerům**	

5.6 Specifikace testovacích činností

Popis: Cílem je zjistit, co vlastně bude náplní testerovy práce – definice typů testů na základě výstupu předchozích kroků, případně na základě další domluvy. Zahájení práce na STS.

Výstup: Výstupem tohoto procesu je základní orientace jaké typy a rozsahy testů budou použity a jaké další činnosti bude tester provádět.

Náročnost: cca 3-5%

Tabulka 11 – Popis procesu Specifikace testovacích činností

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
6.1	**Definice typů testů**	**Získání základního přehledu o náročnosti a rozsahu testování**	**Špatné odhady náročnosti testů či jejich neúplnost**
6.2	**Doplnění údajů do STS**	**Slouží k dokumentaci testerovy práce**	**Chaotická nebo nulová testerská dokumentace**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***6.1**	**Tvorba/doplnění manuálů**	**Mohou pomoci nově příchozím testerům**	

5.7 Plánování testů

Popis: Na základě předcházejících kroků je tester povinen vytvořit odhady na časovou náročnost své činnosti. Jedná se o jednu z nejtěžších činností a je třeba ji vyhradit dostatek času, aby byla provedena správně.

Výstup: Časové odhady jsou užitečné především pro vedoucí týmu, kteří mají na starost plánování a chod celého projektu.

Náročnost: cca 3-5%

Tabulka 12 – Popis procesu Plánování testů

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
7.1	**Sepsání všech plánovaných testů a činností (odhady, psaní všech typů testů, revize testů, spouštění testů)**	**Zpřesnění a zpřehlednění plánovaných činností**	**Tvorba nepřesných odhadů**
7.2	**Ohodnocení těchto činností časovou náročností**	**Zpřesnění a zpřehlednění plánovaných činností**	**Tvorba nepřesných odhadů**
7.3	**Tvorba časových odhadů**	**Pro vhodné náplánování testerovy práce**	**Špatné rozvržení práce**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***7.1**	**Pokud je to možné, je vhodné vytvářet odhady paralelně ve více lidech**	**Porovnání výstupů může zkvalitnit výsledný odhad**	
***7.2**	**Tvorba/doplnění manuálů**	**Mohou pomoci nově příchozím testerům**	

5.8 Implementace manuálních testů

Popis: Implementace manuálních testů je zřejmě nejobtížnější fází přípravy testování. Jedná se o sepsání TS a TC všech typů testů z kapitoly 5.6. Testy (především funkční a nefunkční) jsou psané dle SRS.

Výstup: Sepsané testy jsou po revizi určené k otestování kvality vyvíjeného SW.

Náročnost: cca 12-25%

Tabulka 13 – Popis procesu Implementace manuálních testů

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
8.1	**Vytváření testů (ukázka viz. Příloha IV)**	**Vyšší efektivita tvorby testů**	**Nebezpečí přehlédnutí některých chyb**
8.2	**Testy sepisovat do definovaného prostředí/nástroje**	**Přehlednost**	**Špatná dodatečná údržba**
8.3	**Mapovat UseCases či URD na jednotlivé TS/TC**	**Zajištění otestování požadavků zákazníka**	**Netestování některých požadavků zákazníka**
8.4	**Připravit testovací data pro jednotlivé testy**	**Úplnost testů**	**Zdržení při přípravě dat během testování**
8.5	**Identifikace, které testy jsou určeny pro virtuální a které pro reálné zařízení**	**Specifikace, kde spouštět jednotlivé testy**	**Nemožnost otestování některých typů testů**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***8.1**	**Psát objektivní názvy TS/TC a jejich popisy**	**Údaje, které čte zákazník**	
***8.2**	**Doplňovat prioritu jednotlivých testů**	**Usnadnění výběru TS/TC pro regresní testování**	

5.9 Implementace automatických testů

Popis: Automatické testy na poli mobilních zařízení jsou stále ve fázi vývoje a veškeré nástroje pro tento druh testování mají v držení externí firmy, které nabízejí pouze své služby testování – nikoli však nástroje.

Výstup: Pokud jsou automatické testy využity, mohou velmi usnadnit testerovu práci (hlavně v oblasti regresního testování).

Náročnost: cca 25-37%

Pozn.: *Postup v této kapitole lze aplikovat pouze ve specifických případech a její obsah je nad rámec původního zadání.*

Tabulka 14 – Popis procesu Implementace automatických testů

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
9.1	**Zjištění, zda je možné využít automatických nástrojů**	**Vyšší efektivita práce**	**Pomalejší manuální testování, případně zbytečná investice do autom. testů**
9.2	**Instalace a nastavení nástrojů pro automatické testy**	**Vyšší efektivita práce**	**Pomalejší manuální testování**
9.3	**Psaní automatických testů (dle specifikace)**	**Vyšší efektivita práce**	**Pomalejší manuální testování**
9.4	**Identifikace, které testy jsou určeny pro virtuální a které pro reálné zařízení**	**Specifikace, kde spouštět jednotlivé testy**	**Nemožnost otestování některých typů testů**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***9.1**	**-**	**-**	

5.10 Revize testů

Popis: Revize testů má za úkol zjistit, zda připravené testy pokrývají všechny funkcionality systému, které mají být implementovány v testované verzi. Tuto revizi by měla provádět jiná osoba než autor testů. Nalezené nedostatky je třeba opravit před spuštěním testů.

Výstup: Výstupem jsou revidované testy, které jsou připraveny pro kompletní otestování aktuálně vydané verze.

Náročnost: cca 7-12%

Tabulka 15 – Popis procesu Revize testů

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
10.1	**Nezávisle identifikovat všechny zákazníkovy požadavky na testovanou verzi**	**Nezávislost může pomoci odhalit přehlédnuté požadavky**	**Nedokonalá identifikace požadavků**
10.2	**Kontrola pokrytí testů na verifikaci chyb z minulých verzí**	**Nutnost verifikovat dříve nalezené chyby.**	**Možnost nepřetestování starších chyb**
10.3	**Kontrola pokrytí těchto požadavků**	**Revize testů**	**Zbytečnost revize**
10.4	**Úprava/oprava připravených testů**	**Revize testů**	**Zbytečnost revize**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***10.1**	**Kontrola a případná úprava časových odhadů na testování**	**Možnost odhalení přehlédnutých časových nákladů**	
***10.2**	**Tvorba/doplnění manuálů**	**Mohou pomoci nově příchozím testerům**	

5.11 Provedení testů – virtuální emulátor

Popis: Tento proces se zabývá samotným spouštěním testů na virtuálním zařízení. Toto testování by však mělo být pokud možno vždy doplněno i o testování na zařízení reálném.

Výstup: Výstupem jsou výsledky spuštění jednotlivých testů na reálném zařízení.

Náročnost: cca 7–12%

Tabulka 16 – Popis procesu Provedení testů – virtuální simulátor

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
11.1	**Instalace verze určené pro testování**	**Pro otestování správné verze**	**Otestování špatné či nedokončené verze**
11.2	**Příprava testovacího prostředí dle STS**	**Pro otestování dle správných vstupních požadavků**	**Nepřesné výsledky testování**
11.3	**Příprava testovacích dat dle STS**	**Pro otestování dle správných vstupních požadavků**	**Nepřesné výsledky testování**
11.4	**Spuštění a vyhodnocení jednotlivých testů**	**Samotné testování**	**-**
11.5	**Spuštění tzv. performance testů**	**Možné pouze u virtuálních zařízení**	**Nemožnost pozdějšího otestování na reálném zařízení**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***11.1**	**Zaznamenávat poznatky z testování (úpravy TS/TC, návrhy na vylepšení postupů apod.)**	**Vhodné pro údržbu testů před další iterací**	

5.12 Provedení testů – reálné zařízení

Popis: Tento proces se zabývá samotným spouštěním testů na reálném zařízení.

Výstup: Výstupem jsou výsledky spuštění jednotlivých testů na reálném zařízení.

Náročnost: cca 7–12%

Tabulka 17 – Popis procesu Provedení testů – reálné zařízení

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
12.1	**Instalace verze určené pro testování**	**Pro otestování správné verze**	**Otestování špatné či nedokončené verze**

12.2	**Příprava testovacího prostředí dle STS**	**Pro otestování dle správných vstupních požadavků**	**Nepřesné výsledky testování**
12.3	**Příprava testovacích dat dle STS**	**Pro otestování dle správných vstupních požadavků**	**Nepřesné výsledky testování**
12.4	**Spuštění a vyhodnocení jednotlivých testů**	**Samotné testování**	**-**
Sekundární cíle procesu			
Krok	**Popis činnosti**		**Důvody zavedení**
***12.1**	**Zaznamenávat poznatky z testování (úpravy TS/TC, návrhy na vylepšení postupů apod.)**		**Vhodné pro údržbu testů před další iterací**

5.13 Report a verifikace chyb

Popis: Po testování (pokud nebylo realizováno během něj) přichází na řadu reportování a verifikace chyb. Tento proces se zabývá popisem činností, na které je třeba brát ohled. Tester by měl chyby reportovat tak, aby chyba byla již z reportu jasně a přesně pochopitelná.

Výstup: Tester bude schopen reportovat chyby přesně a jasně, aby je vývojáři byli schopni identifikovat a opravit bez dodatečné komunikace s testerem.

Náročnost: cca 5–7%

Tabulka 18 – Popis procesu Report a verifikace chyb

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
13.1	**Verifikace opravených chyb (opraveno/neopraveno)**	**Nutnost zaznamenání, které chyby jsou skutečně opraveny a které nikoliv**	**Nepřehlednost v kontrolách již opravených chyb**
13.2	**Identifikovat co chyba je a co není (proti SRS)**	**Objektivní report chyb, možnost odhalení chyby v samotných testech**	**Reportování tzv. falešných chyb**

13.3	**Název chyby - uvádět místo chyby i v prefixu názvu chyby**	**Odpověď na otázku: „Kde se chyba vyskytuje?“**	**Nepřehlednost v reportovaných chybách**
13.4	**Stručný popis chyby**	**Odpověď na otázku: „Jak se to chová?“**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.5	**Očekávaný výsledek**	**Odpověď na otázku: „Jak by se to mělo chovat?“**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.6	**Frekvence výskytu, reprodukovatelnost**	**Odpověď na otázku: „Vyskytuje se chyba pravidelně či náhodně?“**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.7	**Kroky k reprodukci**	**Odpověď na otázku: „Jak chybu zopakuji?“**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.8	**Priorita**	**Odpověď na otázku: „Jak rychle je nutné chybu opravit?“**	**Špatná posloupnost oprav chyb**
13.9	**Verze testovaného produktu**	**Odpověď na otázku: „Ve které verzi se chyba vyskytuje?“**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.10	**Verze a nastavení testovacího prostředí**	**Odpověď na otázku: „Na jakém HW/SW se chyba vyskytuje“**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.11	**Použitá testovací data (např. příloha k reportu)**	**Odpověď na otázku: „Při jakých datech se chyba vyskytuje?“**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.12	**Dodatečné informace (např. přístupový účet, použitý server)**	**Vhodné pro bližší specifikaci chyby**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**
13.13	**Přílohy (screenshot, log, atd.)**	**Urychlení identifikace a opravy chyby**	**Nejasný popis chyby = dodatečná komunikace s vývojářem**

Sekundární cíle procesu		
Krok	**Popis činnosti**	**Důvody zavedení**
***13.1**	**Pokud není automaticky pak: Unikátní ID chyby**	**Lepší přehlednost v reportech chyb**
***13.2**	**Pokud není automaticky pak: Jméno testera**	**Lepší přehlednost v reportech chyb**
***13.3**	**Pokud není automaticky pak: Datum nalezení**	**Lepší přehlednost v reportech chyb**
***13.4**	**Snaha přiřazovat chyby k TS/TC, ve kterých byla nalezena**	**Usnadnění verifikace chyb**
***13.5**	**Tvorba/doplnění manuálů**	**Mohou pomoci nově příchozím testerům**

5.14 Vyhodnocení výsledků testů

Popis: Po dokončení všech testů je nutné vypracovat tzv. System Test Report (STR). Tento dokument slouží jako doklad o provedené práci jak pro zákazníka, tak pro vedoucího týmu a vývojářský tým.

Výstup: Výstupem tohoto procesu bude STR, který bude obsahovat veškeré údaje o dosavadním testování.

Náročnost: cca 3–5%

Tabulka 19 – Popis procesu Vyhodnocení výsledků testů

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
14.1	**Tvorba základní struktury STR (obvykle z STS)**	**Zachování formátu STS versus STR (a urychlení práce)**	**Nepřehlednost výsledného reportu**
14.2	**Kontrola/doplnění testované verze**	**Nutnost specifikace testované verze**	**Neúplnost výsledného reportu**
14.3	**Kontrola/doplnění použitého testovacího prostředí**	**Pro sledování objektivnosti testování**	**Neúplnost výsledného reportu**

14.4	**Kontrola/doplnění použitých testovacích dat**	**Pro sledování objektivnosti testování**	**Neúplnost výsledného reportu**
14.5	**Kontrola/doplnění plánovaných testů**	**Revize**	**Neúplnost výsledného reportu**
14.6	**Kontrola/doplnění výsledků testů (Pass/Failed)**	**Základní údaje celého STR**	**Neúplnost výsledného reportu**
14.7	**Kontrola/doplnění výsledků testů (ID + krátký popis nalezených chyb)**	**Základní údaje celého STR**	**Neúplnost výsledného reportu**
14.8	**Kontrola/doplnění jmen jednotlivých testerů**	**Přehlednost**	**Neúplnost výsledného reportu**
14.9	**Kontrola/doplnění dat spuštění testů**	**Přehlednost**	**Neúplnost výsledného reportu**

Sekundární cíle procesu

Krok	Popis činnosti	Důvody zavedení
***14.1**	**Možnost doplnění různých statistik/grafů z testování**	**Ocení především zákazník či project manager**
***14.2**	**Možnost doplnění reálné časové náročnosti jednotlivých činností**	**Užitečné pro project managera a tvorbu příštích časových odhadů**

5.15 Údržba testů

Popis: Pokud bylo testování jen pro určitý vývojový cyklus (např. tzv. sprint v metodice SCRUM), či pokud byly v průběhu testů nalezeny chyby, bude nutné testy (nebo jejich část) spustit znova. Před tímto spuštěním by však měla proběhnout určitá údržba (aktualizace/oprava) testů.

Výstup: Tento proces se zabývá identifikací kroků, které je nutné provést před další testovací iterací.

Náročnost: cca 5–7%

Tabulka 20 – Popis procesu Údržba testů

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
15.1	**Vyhodnocení a zpracování vstupů od zákazníka, tzv. customer feedback**	**Zpracování požadavků zákazníka**	**Nespokojenost zákazníka**
15.2	**Vyhodnocení a zpracování vstupů od vedoucího týmu, tzv. PM feedback**	**Zpracování požadavků zákazníka**	**Nespokojenost zákazníka**
15.3	**Případné opravy/doplnění TS/TC**	**Vyšší kvalita testů pro příští testovací iteraci**	**Nepřesné testy**
15.4	**Případné opravy/doplnění regresních testů**	**Vyšší kvalita testů pro příští testovací iteraci**	**Nepřesné testy**
15.5	**Doplnění kontroly nalezených/neopravených chyb do příští testovací iterace**	**Umožnění např. zohlednění vyšších časových nároků na spouštění testů**	**Přehlednutí kontroly opravených chyb v příští testovací iteraci**
15.6	**Vrácení zařízení (a případně i generátorů dat) do výchozího nastavení**	**Nastavení stejných podmínek pro další testování**	**Nebezpečí odlišností ve výsledcích příštích testů**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***15.1**	**Vyhodnocení možnosti použití automatizace některých procesů testování**	**Urychlení práce v příští iteraci**	

5.16 Závěr testování

Popis: Tento proces následuje po skončení testovacích prací na projektu a slouží spíš jako doporučení pro testera, kterých kroků by se měl v takovém případě držet.

Výstup: Výstupem je tzv. čisté ukončení působnosti testera na daném projektu.

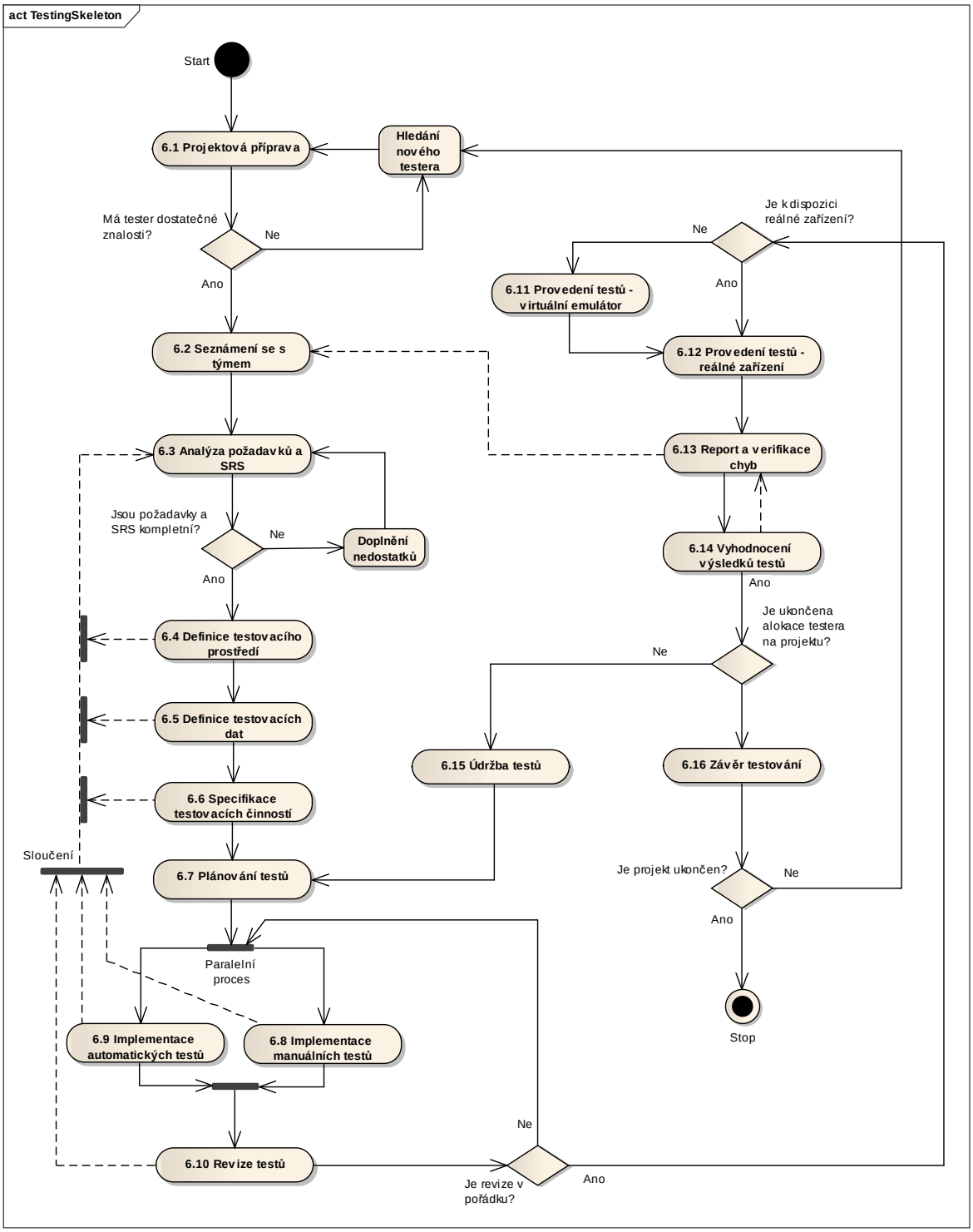
Náročnost: cca 7–12%

Tabulka 21 – Popis procesu Závěr testování

Primární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	**Rizika při nedodržení**
16.1	**Uzavření/předání všech vlastních úkolů na projektu**	**Dokončení všech alokovaných úkolů**	**Nebezpečí dodatečné uzavírání úkolů nezpůsobilými lidmi**
16.2	**Doplnění veškeré dokumentace (např. manuály)**	**Uzavření práce na dokumentech**	**Nebezpečí dodatečné dokončování dokumentů nezpůsobilými lidmi**
16.3	**Sepsání zkušeností s projektem (tzv. Lessons Learnt)**	**Může pomoci k lepším výsledkům při přístím projektu**	**Opakování stejných chyb i při přístím projektu**
16.4	**Nastavení zapůjčených zařízení do původního nastavení**	**Důležité pro postoupení zařízení na další projekty**	**Chybně nastavená zařízení, práce navíc na dalších projektech**
16.5	**Vrácení zapůjčených zařízení (včetně příslušenství)**	**Důležité pro postoupení zařízení na další projekty**	**Nedostatek zařízení pro další projekty**
16.6	**Úklid svěřených projektových složek na serverech**	**Přehlednost projektových složek**	**Nekonzistence projektových složek**
16.7	**Úklid dat na lokálním počítači**	**Uvolnění místa**	**Postupné vyčerpání kapacity pevného disku**
16.8	**Případné předání funkce a školení nového testera**	**Plynulé začlenění nového člověka do týmu**	**Zdlouhavé začlenění nového testera do týmu**
Sekundární cíle procesu			
Krok	**Popis činnosti**	**Důvody zavedení**	
***16.1**	**Tvorba/doplnění manuálů**	**Mohou pomoci nově příchozím testerům**	

5.17 Vývojový diagram

Vývojový diagram testing skeletonu skládající se z jednotlivých kroků z předchozí kapitoly a jejich posloupnosti.



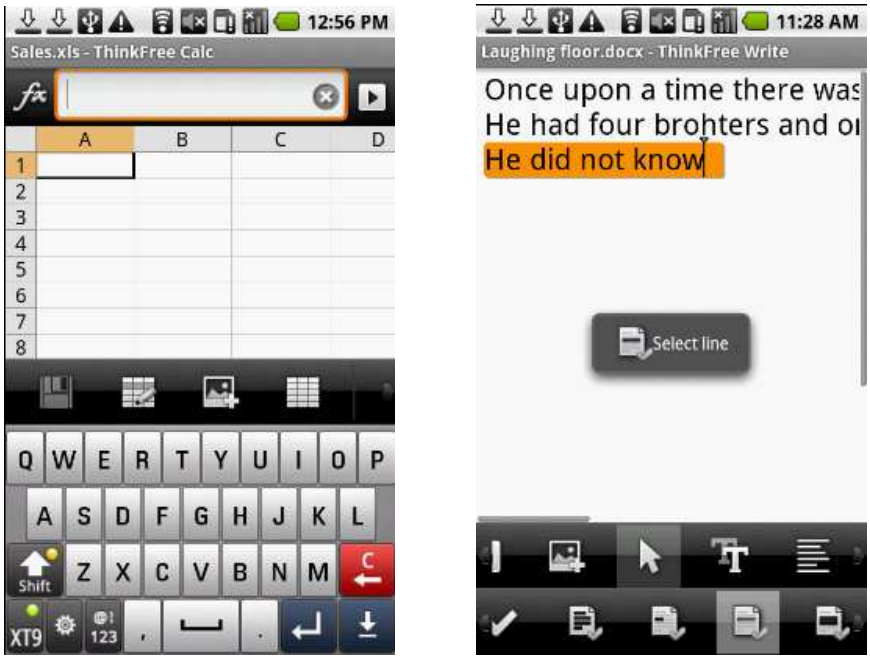
Obrázek 6 – Diagram průchodu testing skeletonem

6 TESTOVÁNÍ APLIKACE PRO ANDROID

Tato kapitola se zabývá simulací praktického nasazení předcházejícího testing skeletonu při testování aplikace pro zařízení s operačním systémem Android.

6.1 Popis aplikace

Pro účely otestování byla zvolena aplikace ThinkFree Office. Jedná se o mobilní provedení kancelářského balíku, který umožňuje otevírání, editaci a tvorbu dokumentů typu *.ppt, *.doc, *.xls.



Obrázek 7 – Náhledy aplikace ThinkFree Office pro Android [32]

6.2 Vstupy

V tomto testu bude testována jeho verze ThinkFree Mobile Viewer, která umožňuje pouze čtení dokumentů a je často k dispozici na nových mobilních zařízeních (čili bez nutnosti dodatečné instalace).

6.2.1 Vstupní požadavky

N/A – nejedná se o zákaznický projekt, v tomto případě budou tedy uvažovány pouze dva zdroje požadavků:

1. Help (pro funkční testování)

2. Obecná funkcionalita (viz kapitola Nevyslovené požadavky)

6.2.2 Testovací prostředí

Aplikace je určena pro nový mobilní telefon na trhu – LG Optimus One, na kterém bude distribuována již při koupi. Z tohoto důvodu je nutné aplikaci otestovat právě na tomto zařízení a při jeho továrním nastavení.

6.2.3 Testovací data

Testovací data nejsou dodána. Je nutné si je vytvořit manuálně. Dle zadání mají být otestovány formáty *.doc, *.docx, *.ppt, *.pptx, *.xls, *.xlsx, *.pdf.

6.3 Výstup podle testing skeletonu

Tato kapitola uvádí samotný postup činnosti testera dle navrhnutého testing skeletonu. Každá podkapitola obsahuje tabulku procesů a určitý výstup. Z hlediska objemu výstupních dat a dokumentace je výstup omezen na slovní vyjádření či odkaz na přílohu této práce.

6.3.1 Projektová příprava

Tabulka 22 – Projektová příprava - Výstup

Krok	Popis činnosti	Výstup
1.1	**Seznámit se s cílem projektu**	Otestování kancelářského balíku v jeho Lite verzi pro telefon LG Optimus One = vytvoření seznamu případných chyb
1.2	**Pochopit použité technologie**	Využito pouze OS Android a ovládacích prvků telefonu (funkční black-box testování)
1.3.	**Zjistit jazyk pro komunikaci s klientem**	Dokumentace v angličtině, STS a STR pro české zastoupení v češtině
***1.1**	**Zjistit bližší informace o klientovi**	Společnost zabývající se vývojem kancelářského SW pro Internet a mobilní zařízení. Další Informace na stránkách společnosti. (Možnost nabídnutí služeb dalšího testování)
***1.2**	**Zvážit crowdsourced testování**	Z důvodu velikosti a funkcionalitě Lite verze není nutné

6.3.2 Seznámení se s týmem

Tabulka 23 – Seznámení se s týmem - Výstup

Krok	Popis činnosti	Výstup
2.1	**Definice kompetencí jednotlivých členů týmu**	V tomto případě se jedná pouze o jednoho externího testera, který komunikuje přímo s project managerem (zástupce zadavatele)
2.2	**Zpřístupnění projektových adresářů**	Aplikace zpřístupněna na vnitřním FTP společnosti
2.3	**Začlenění do projektové e-mail skupiny (+ vytvoření pravidel v Outlooku)**	Projektová skupina neexistuje – komunikace probíhá jen mezi testerem a project managerem
2.4	**Přístup do bugsystému (Mantis, Bugzilla, TFS)**	Chyby budou značeny přímo k jednotlivým TS (viz. ukázka v Příloze V)
2.5	**Centralizace projektových úkolů (office documents, TFS, XPlanner)**	N/A
2.6	**Instalace ostatních projektových prostředí**	E-mail, FTP-klient, Skype
***2.1**	**Žádost o informování o nové verzi a místu jejího uložení (pomocí email/skype)**	Jedná se pouze o jednu iteraci testování
***2.2**	**Žádost o umístění verzování přímo do aplikace (název, verze, datum)**	Společnost má vlastní způsob verzování
***2.3**	**Žádost o zveřejňování úprav v každé vydané verzi (i interní)**	Verze bude testována jako celek dle kompletní dokumentace
2.4	**Při opravách bugů by měli vývojáři definovat, ve které verzi se oprava vyskytne**	N/A
2.5	**Zavedení tvorby logu přímo v mobilním zařízení**	N/A
2.6	**Zavedení možnosti vytváření screenshots přímo v aplikaci**	N/A

6.3.3 Analýza požadavků a SRS

Tabulka 24 – Analýza požadavků a SRS - Výstup

Krok	Popis činnosti	Výstup
3.1	**Nastudování SRS**	Nejedná se o zákaznický projekt. Jediná dokumentace je k dispozici na [32]
3.2	**Analýza URD a SRS (viz. Příloha III)**	Nejedná se o zákaznický projekt. Dodatečné informace budou vykomunikovány s odpovědným project managerem.
3.4	**Založení STS (tvorba šablony dokumentu)**	Přebrána STS šablona zadavatele
3.1	**Nastudování SRS**	Přečtena projektová dokumentace – User Guide
***3.1**	**Rezervace/zapůjčení cílových mobilních zařízení**	Vyhrazen jeden LG Optimus One pro testovací účely
***3.2**	**Brainstorming – nechybí něco?**	N/A

6.3.4 Definice testovacího prostředí

Tabulka 25 – Definice testovacího prostředí - Výstup

Krok	Popis činnosti	Výstup
4.1	**Identifikace potřebného HW a SW (včetně licencí) z SRS**	Zadání hovoří pouze o jednom zařízení – LG Optimus One (Android 2.2 Froyo), tovární nastavení, je nutné vytvořit google účet
4.2	**Zajištění těchto prostředků – Odblokované telefony, SIM karty (přesné konfigurace)**	Telefon zapůjčen včetně kompletního příslušenství (SD karta, nabíječka, USB kabel, manuál)
4.3	**Obstarání uživatelských manuálů ke všem zařízením**	Viz. předchozí bod
4.4	**Instalace a nastavení SW (virtuální simulátor)**	Využit AVD emulátor od Google. V něm vytvořeno zařízení, které je přesnou kopií LG Optimus One. Bohužel emulátor je i na 2-jádrovém procesoru velmi pomalý
4.5	**Nastavení HW (dle SRS)**	Testování bude probíhat při továrním nastavení zařízení
4.6	**Zavedení okolního prostředí**	Specifická okolní prostředí (hluk, tma, …) nejsou známy
4.7	**Doplnění údajů do STS**	Údaje o HW a SW zaneseny do připraveného STS
***4.1**	**Vyhodnocení nutnosti tzv. field testu**	Není nutno

***4.2**	**Tvorba/doplnění manuálů**	Z důvodu jednoduchosti přípravy testovacího prostředí není pro toto zaveden žádný manuál

6.3.5 Definice testovacích dat

Tabulka 26 – Definice testovacích dat - Výstup

Krok	Popis činnosti	Výstup
5.1	**Určení typu dat – dynamická/statická data (SRS/Zákazník)**	Data jsou statická – office dokumenty. Tato data jsou čtena z paměťové karty telefonu
5.2	**Identifikace potřebného vzorku testovacích dat (SRS/Zákazník)**	Vytvořeny 2 vzorky (prázdný soubor a soubor s údaji) pro každý typ kompatibilního dokumentu
5.3	**Zajištění testovacích dat (Zákazník)**	Tvorba testovacích dokumentů na PC (MS Office 2003 a 2007)
5.4	**Doplnění údajů do STS**	Údaje o použitých testovacích datech zaneseny do STS
***5.1**	**Tvorba vlastního generátoru dat**	Není potřeba
***5.2**	**Tvorba/doplnění manuálů**	Z důvodu jednoduchosti přípravy testovacího prostředí není pro toto zaveden žádný manuál

6.3.6 Specifikace testovacích činností

Tabulka 27 – Specifikace testovacích činností - Výstup

Krok	Popis činnosti	Výstup
6.1	**Definice typů testů**	Zákazník požaduje pouze funkční black-box testování a testy použitelnosti. Online sekce nebude testována.
6.2	**Doplnění údajů do STS**	Typ zvolených testů doplněn do STS
***6.1**	**Tvorba/doplnění manuálů**	N/A

6.3.7 Plánování testů

Tabulka 28 – Plánování testů - Výstup

Krok	Popis činnosti	Výstup
7.1	**Sepsání všech plánovaných testů a činností (odhady, psaní všech typů testů, revize testů, spouštění testů)**	Pouze funkční black-box a usability testy
7.2	**Ohodnocení těchto činností časovou náročností**	Shodná náročnost
7.3	**Tvorba časových odhadů**	Cca 10MD (1MD = 8h)

***7.1**	**Pokud je to možné, je vhodné vytvářet odhady paralelně ve více lidech**	N/A
***7.2**	**Tvorba/doplnění manuálů**	N/A

6.3.8 Implementace manuálních testů

Tabulka 29 – Implementace manuálních testů - Výstup

Krok	Popis činnosti	Výstup
8.1	**Vytváření testů (ukázka viz. Příloha IV)**	Ukázka použitých TS viz. Příloha V
8.2	**Testy sepisovat do definovaného prostředí/nástroje**	Testy sepisovány do šablony klienta
8.3	**Mapovat UseCases či URD na jednotlivé TS/TC**	N/A
8.4	**Připravit testovací data pro jednotlivé testy**	Data přiložena k testům
8.5	**Identifikace, které testy jsou určeny pro virtuální a které pro reálné zařízení**	Oba typy testů jsou vhodné jak pro virtuální, tak pro reálné zařízení
***8.1**	**Psát objektivní názvy TS/TC a jejich popisy**	Viz. Příloha V
***8.2**	**Doplňovat prioritu jednotlivých testů**	V tomto případě není nutné

6.3.9 Implementace automatických testů

Jelikož se jedná o jedno-iterační testování, není nutné vytvářet automatické testy.

6.3.10 Revize testů

Tabulka 30 – Revize testů - Výstup

Krok	Popis činnosti	Výstup
10.1	**Nezávisle identifikovat všechny zákazníkovy požadavky na testovanou verzi**	Z důvodu přítomnosti pouze jednoho test analytika je jím revize provedena vůči uživatelské příručce k programu a vůči doporučením v této práci
10.2	**Kontrola pokrytí testů na verifikaci chyb z minulých verzí**	N/A
10.3	**Kontrola pokrytí těchto požadavků**	Nalezeny drobné nedostatky a překlepy

10.4	**Úprava/oprava připravených testů**	Úprava aplikována
***10.1**	**Kontrola a případná úprava časových odhadů na testování**	Ň/A
***10.2**	**Tvorba/doplnění manuálů**	N/A

6.3.11 Provedení testů

Tabulka 31 – Provedení testů - Výstup

Krok	Popis činnosti	Virtuální emulátor	Reálné zařízení
11.1	**Instalace verze určené pro testování**	Tvorba virtuálního zařízení – kopie reálného	Zařízení je připraveno z výroby
11.2	**Příprava testovacího prostředí dle STS**	Spuštění virtuálního zařízení	Zařízení je připraveno z výroby
11.3	**Příprava testovacích dat dle STS**	Nakopírování dat na virtuální SD kartu skrze IDE Eclipse	Nakopírování dat na SD kartu
11.4	**Spuštění a vyhodnocení jednotlivých testů**	Viz. Příloha V	Viz. Příloha V
***11.1**	**Zaznamenávat poznatky z testování (úpravy TS/TC, návrhy na vylepšení postupů apod.)**	N/A	N/A

6.3.12 Report a verifikace chyb

Tabulka 32 – Report a verifikace chyb - Výstup

Krok	Popis činnosti	Výstup
13.1	**Verifikace opravených chyb (opraveno/neopraveno)**	N/A
13.2	**Identifikovat co chyba je a co není (proti SRS)**	Nalezené chyby verifikovány vůči uživatelské příručce
13.3	**Název chyby - uvádět místo chyby i v prefixu názvu chyby**	Doplněno do STR
13.4	**Stručný popis chyby**	Doplněno do STR
13.5	**Očekávaný výsledek**	Doplněno do STR
13.6	**Frekvence výskytu, reprodukovatelnost**	Doplněno do STR
13.7	**Kroky k reprodukci**	Doplněno do STR
13.8	**Priorita**	Doplněno do STR

13.9	**Verze testovaného produktu**	Doplněno do STR
13.10	**Verze a nastavení testovacího prostředí**	Doplněno do STR
13.11	**Použitá testovací data (např. příloha k reportu)**	Doplněno do STR
13.12	**Dodatečné informace (např. přístupový účet, použitý server)**	N/A
13.13	**Přílohy (screenshot, log, atd.)**	Tvorba fotek při testování na reálném zařízení, tvorba screenshots na virtuálním zařízení
***13.1**	**Pokud není automaticky pak: Unikátní ID chyby**	Řešeno autoinkrementací ID chyby
***13.2**	**Pokud není automaticky pak: Jméno testera**	Není nutno – pouze jeden tester
***13.3**	**Pokud není automaticky pak: Datum nalezení**	Doplněno do STR
***13.4**	**Snaha přiřazovat chyby k TS/TC, ve kterých byla nalezena**	Doplněno do STR
***13.5**	**Tvorba/doplnění manuálů**	N/A

6.3.13 Vyhodnocení výsledků testů

Tabulka 33 – Vyhodnocení výsledků - Výstup

Krok	Popis činnosti	Výstup
14.1	**Tvorba základní struktury STR (obvykle z STS)**	Chyby přepsány do šablony STR (vznik úpravou STS)
14.2	**Kontrola/doplnění testované verze**	Doplněno
14.3	**Kontrola/doplnění použitého testovacího prostředí**	Převzato z STS
14.4	**Kontrola/doplnění použitých testovacích dat**	Převzato z STS
14.5	**Kontrola/doplnění plánovaných testů**	Převzato z STS
14.6	**Kontrola/doplnění výsledků testů (Pass/Failed)**	Doplněno
14.7	**Kontrola/doplnění výsledků testů (ID + krátký popis nalezených chyb)**	Doplněno (viz. Příloha V)

14.8	**Kontrola/doplnění jmen jednotlivých testerů**	Převzato z STS
14.9	**Kontrola/doplnění dat spuštění testů**	Doplněno
***14.1**	**Možnost doplnění různých statistik/grafů z testování**	N/A
***14.2**	**Možnost doplnění reálné časové náročnosti jednotlivých činností**	N/A

6.3.14 Údržba testů

V tomto případě se nevykonává. Testy jsou vykonány pouze jednou a následně zálohovány a předány zákazníkovi. Při dalším testování se testy upraví dle nové specifikace.

6.3.15 Závěr testování

Tabulka 34 – Závěr testování - Výstup

Krok	Popis činnosti	Výstup
16.1	**Uzavření/předání všech vlastních úkolů na projektu**	Kontrola splnění všech testovacích úkolů
16.2	**Doplnění veškeré dokumentace (např. manuály)**	STS i STR kompletní – předáno zákazníkovi
16.3	**Sepsání zkušeností s projektem (tzv. Lessons Learnt)**	Zkušenosti zaznamenány do poznámek k testování mobilních aplikací
16.4	**Nastavení zapůjčených zařízení do původního nastavení**	Pomocí volby v menu, u virtuálního zařízení reset nastavení
16.5	**Vrácení zapůjčených zařízení (včetně příslušenství)**	Reálné zařízení vráceno
16.6	**Úklid svěřených projektových složek na serverech**	N/A
16.7	**Úklid dat na lokálním počítači**	Zálohovány testy i test data
16.8	**Případné předání funkce a školení nového testera**	N/A
***16.1**	**Tvorba/doplnění manuálů**	N/A

ZÁVĚR

Tato diplomová práce měla jako hlavní cíl najít a identifikovat rozdíly mezi softwarovým testováním a testováním aplikací pro mobilní zařízení, určit jejich optimální řešení a k tomuto testování vytvořit jednotný testing skeletonu, který bude popisovat jednotlivé činnosti testera během projektu.

Byla provedena literární studie na dané téma se zaměřením na principy a metodiku testování mobilních aplikací a jejich rozdíly proti testování softwaru pro platformu PC.

Pro základní identifikaci těchto rozdílů mezi nimi byla kromě zdrojů z internetu využita hlavně více než 3-letá osobní zkušenost v oboru testování. Díky tomu byly identifikovány jak rozdíly v přístupu k samotnému testování, tak například i rozdíly při definici uživatelských požadavků.

Byla vypracována srovnávací studie, která obsahuje základní odlišnosti testování software a testování mobilních aplikací (Kapitola 2.3) a to včetně analýzy rizik s hodnocením jejich závažnosti a výskytu (Kapitola 3.7.3).

Hlavním výstupem celé práce je návrh testing skeletonu, který v jednotlivých svých procesech popisuje doporučenou činnost testera během projektu. Každý proces je popsán krátkým úvodem, očekávaným výstupem a také předpokládanou časovou náročností. Těla jednotlivých procesů jsou rozdělena do dvou částí – primární (povinné) a sekundární (nepovinné). Obě části jsou pak tvořeny kroky, které obsahují popis, důvod jejich zavedení a rizika při nedodržení.

Pro správný postup testing skeletonem byl vytvořen diagram, který znázorňuje propojení jednotlivých procesů a určuje tak jejich správnou posloupnost během testování na daném projektu.

Na závěr byl tento testing skeleton aplikován na black-box testování jednoduché aplikace pro mobilní zařízení s operačním systémem Android. Toto mělo za cíl především vyzkoušet nasazení testing skeletonu v praxi a doladit tak jeho případné nedostatky. Testování probíhalo paralelně na fyzickém zařízení (LG Optimus One) a na virtuálním emulátoru se shodnými parametry (AVD emulátor od Google). Díky tomu bylo zjištěno, že některé chyby se skutečně dají odhalit jen na reálném zařízení (např. špatné vykreslení při použití funkce zoom). Virtuální emulátor navíc běžel na průměrně výkonném počítači

velmi pomalu a znemožnil tak jakékoli testy výkonu. Během samotného testování bylo díky navrženému testing skeletonu odhaleno i několik dalších chyb. Ukázka použitého test designu včetně výsledků testů je k nalezení v přílohách. Tabulky pro definici uživatelských požadavků a jejich pokrytí testy jsou uvedeny v přílohách.

Po ukončení testování byly veškeré poznatky zaneseny zpětně do testing skeletonu a ten je tak nyní připraven k praktickém nasazení na skutečných projektech pro testování aplikací pro mobilní zařízení.

SUMMARY

The main aim of this thesis is to find and identify differences between software testing and testing of mobile applications, specify their optimal solution and create unified testing skeleton for this testing which describes the various tester's activities during the project.

The literature search with aim on principles and methods of mobile application testing and their differences from testing of PC software was conducted.

Not only internet sources but more than three-year experience with testing was used for basic identification of these differences.

The comparative study which contains basic differences between PC software testing and testing of mobile applications (Chapter 2.3) was designed. It contains risk analysis with evaluation of relevance and the occurrence (Chapter 3.7.3).

The main output of this thesis is design of testing skeleton which in its individual processes describes recommended tester's activity during the project. Each process is described by short introduction, expected output and assumed time severity. Bodies of individual processes are divided into two parts – primary (mandatory) and secondary (optional). Both parts are compiled from steps which are consist from description, reason of introduction and risk for non-compliance.

The testing skeleton diagram was created. This diagram describes connections between processes which determine correct sequence during testing on project.

In the end the testing skeleton was applied on black-box testing of simple application for Android mobile devices. The main aim of this was to try out the testing skeleton to improve its possible imperfections. Testing was performed in parallel on real device (LG Optimus One) and in virtual emulator with the same parameters (AVD emulator from Google). It has been found that some errors is possible to find on real device only (e.g. incorrect screen render during the usage of the zoom functionality). Because of slow running of virtual emulator the performance tests could not have been done. Thanks to the designed testing skeleton a few another errors were found. Preview of the used test design and test report (results included) are attached. Tables for definitions of user requirements and their test coverage are attached too.

When testing had been finished then all lessons learnt were backward entered into the testing skeleton. It is now ready for using on real mobile application testing projects.

SEZNAM POUŽITÉ LITERATURY

Monografie:

[1] PATTON, Ron. *Testování softwaru*. 1. vyd. Praha: Computer Press, 2002.

[2] GLENFORD J. Myers. *The Art of Software Testing*. John Wiley and Sons, Inc., 2nd edition, 2004.

Internetové zdroje (OS):

[3] *Google : Android* [online]. 2010. Dostupné z WWW: <http://www.android.com>.

[4] *Apple : iOS* [online]. 2010. Dostupné z WWW: <http://www.apple.com/ios>.

[5] Android : Operační systém. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26.4.2008, last modified on 13.4.2011 [cit. 2011-04-22]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Android_%28opera%C4%8Dn%C3%AD_syst%C3%A9m%29>.

[6] *HelloAndroid.com* [online]. 2010 [cit. 2011-04-22]. HelloAndroid.com. Dostupné z WWW: <http://www.helloandroid.com/devices>.

[7] IOS : Apple. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 7.3.2008, last modified on 22.4.2011 [cit. 2011-04-22]. Dostupné z WWW: <http://en.wikipedia.org/wiki/IOS_%28Apple%29>.

[8] *IPhone Forum CZ* [online]. 2010 [cit. 2011-04-22]. IPhone Forum CZ. Dostupné z WWW: <http://forum.iphone.cz/>.

[9] List of iOS devices. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16.6.2009, last modified on 17.5.2011 [cit. 2011-04-22]. Dostupné z WWW: <http://en.wikipedia.org/wiki/List_of_iOS_devices>.

[10] *Microsoft Windows Phone CZ* [online]. 2011 [cit. 2011-04-22]. Windows Phone. Dostupné z WWW: <http://www.microsoft.com/cze/windowsphone/>.

[11] *BlackBerry Czech* [online]. 2011 [cit. 2011-04-22]. BlackBerry. Dostupné z WWW: <http://blackberryczech.cz/>.

[12] MAREČEK, Ivo. BlackBerry: co na té ostružině všichni vidí?. *MobilMania.cz* [online]. 5.2.2009, -, [cit. 2011-04-22]. Dostupný z WWW: <http://www.mobilmania.cz/clanky/blackberry-co-na-te-ostruzine-vsichni-vidi/sc-3-a-1121405>.

[13] *Symbian Nokia* [online]. 2011 [cit. 2011-04-22]. Symbian Nokia. Dostupné z WWW: <http://symbian.nokia.com/>.

Internetové zdroje (ostatní):

[14] *Testing Mobile Application is Different from Testing Traditional Application*. A white paper by Llonbridge, USA, 2006.

[15] ŠTRBÁK, Martin; VALIENTOVÁ, Silvia; GROSSL, Zdeněk. *Blog o testování* [online]. 2010. Dostupné z WWW: <http://cz.-testing.blogspot.com>.

[16] *Syllabus ISTQB CZ : Certifikovaný tester - učební osnovy* [online]. 2007 [cit. 2011-04-22]. Dostupné z WWW: <http://www.swtestovani.cz/store/ISTQB_CTFL_Syllabus_v2007-CZ-beta1.pdf>.

[17] *Trustica :* Poskytovatel *bezpečnostních služeb* [online]. 2009 [cit. 2011-04-22]. Trustica. Dostupné z WWW: <http://www.trustica.cz/penetracni-testy/>.

[18] *Normy jakosti* [online]. 2008 [cit. 2011-04-22]. Normy jakosti. Dostupné z WWW: <http://www.technickenormy.cz/normy-jakosti-a-environmentalniho-managementu/normy-jakosti-en-iso-9000>.

[19] *Qcom* [online]. 2011 [cit. 2011-04-22]. Qcom. Dostupné z WWW: <http://www.qcom.cz/spolecnost/pojmy-a-zkratky/#9000>.

[20] *EISO.cz* [online]. 2006 [cit. 2011-04-22]. EISO.cz. Dostupné z WWW: <http://www.eiso.cz/poradenstvi/nase-sluzby/ISO-20000/>.

[21] JANOTA PhD., Ing. David. QA General Presentation. In *CN Group Czech - QA* [online]. 2011 [cit. 2011-04-22]. Dostupné z WWW: <Interní materiál společnosti CN Group Czech>.

[22] *TestLink* [online]. 2011 [cit. 2011-04-22]. TestLink. Dostupné z WWW: <http://www.teamst.org/>.

[23] *MS Test Manager* [online]. 2011 [cit. 2011-04-22]. MS Test Manager. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/bb385901.aspx>.

[24] *Borland SilkTest* [online]. 2009 [cit. 2011-04-22]. Borland SilkTest. Dostupné z WWW: <http://www.borland.com/us/products/silk/silktest/index.aspx>.

[25] *IBM Rational Robot* [online]. 2010 [cit. 2011-04-22]. IBM Rational Robot. Dostupné z WWW: <http://www-01.ibm.com/software/awdtools/tester/robot>.

Internetové zdroje (Wikipedia a obrázky):

[26] Systém řízení jakosti. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 4.10.2007, last modified on 12.4.2011 [cit. 2011-04-22]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Syst%C3%A9m_%C5%99%C3%ADzen%C3%AD_jakosti>.

[27] *Tutorials Point* [online]. 2011 [cit. 2011-04-22]. Tutorials Point. Dostupné z WWW: <http://www.tutorialspoint.com/cmmi/cmmi-maturity-levels.htm>.

[28] Vzorkování. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 6.2.2005, last modified on 4.4.2011 [cit. 2011-04-22]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Vzorkov%C3%A1n%C3%AD>.

[29] HP QuickTest Professional. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 6.12.2007, last modified on 31.3.2011 [cit. 2011-04-22]. Dostupné z WWW: <http://en.wikipedia.org/wiki/HP_QuickTest_Professional>.

[30] Test automation. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 20.8.2004, last modified on 21.4.2011 [cit. 2011-04-22]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Test_automation>.

[31] *Android Developers* [online]. 2011 [cit. 2011-04-22]. Android Developers. Dostupné z WWW: <http://android-developers.blogspot.com>.

[32] *ThinkFree* [online]. 2011 [cit. 2011-05-09]. ThinkFree Office Manual. Dostupné z WWW: < http://help.thinkfree.com/android/en_us/android_guide.htm >.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

IT	Informační technologie
OS	Operační systém
SW	Software
HW	Hardware
FW	Firmware
TC	Test Case
TS	Test Suite
URD	User Requirements Definition
STS	System Test Specification
STR	System Test Report
SRS	System Requirements Specification
SPR	System Problem Report (bug, chyba v softwaru)
ROI	Return of Investments
QMS	Quality Management System
QA	Quality Assurance
XML	Extensible Markup Language
MS	Microsoft
TFS	Team Foundation Server
ISTQB	International Software Testing Qualifications Board
CMMI	Capability Maturity Model Integration
SDK	Software Development Kit
IDE	Integrated Development Environment
MD	Man Day = 8 hodin
UI / GUI	User Interface / Graphical User Interface

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OS ANDROID - JEDNOTLIVÉ VERZE A ODLIŠNOSTI

Verze 2.0/2.1 (Eclair)

Datum: 26. října 2009
Linuxové jádro: 2.6.29
Novinky:

- Optimalizována rychlost hardwaru
- Podpora pro více velikostí a rozlišení displeje
- Nové prostředí prohlížeče a podpora HTML5
- Mapy Google aktualizovány na 3.1.2
- Podpora pro Microsoft Exchange
- Podpora přisvětlovací diody
- Digitální zoom (fotoaparát)
- Podpora pro Bluetooth 2.1

Verze 2.2 (Froyo)

Datum: 20. května 2010
Linuxové jádro: 2.6.32
Novinky:

- Možnost instalovat aplikace na paměťovou kartu
- Adobe vydalo plugin Adobe Flash 10.1. Není integrován do systému, distribuce je řešena přes Android Market nebo přes stránky Adobe
- Díky JIT (Just-in-time) kompilátoru se podařilo zvýšit rychlost sytému na různých benchmarcích 2x až 5x. Dále je vylepšena správa paměti RAM
- Možnost vytvořit z telefonu WiFi hotspot, nebo sdílet internetové připojení přes USB kabel
- Dva nové režimy telefonu – „car mode“ a „night mode“ (režim v autě a noční režim)
- Přidána podpora pro OpenGL ES 2.0, vícebarevný trackball, vylepšena podpora pro Exchange, Bluetooth a přidána další vrstva vývojářského API

Verze 2.3 (Gingerbread)

Datum: 6. prosince 2010
Linuxové jádro: 2.6.35
Novinky:

- Podpora video formátu WebM pro HTML5 video
- Podpora pro Near Field Communication standard, který dnes podporují některé mobilní telefony
- Podpora SIP protokolu pro internetovou telefonii
- Lepší správa prostředků
- Podpora více kamer a nových senzorů
- Rozšíření podpory nativního kódu

Verze 3.0 (Honeycomb)

Datum: ? (finální SDK vydáno na konci února 2011)
Linuxové jádro: 2.6.36
Novinky:

- Nové grafické rozhraní (**optimalizováno pro tablety**)
- Přidán systémová lišta včetně notifikací
- Plně nastavitelná centrální plocha
- Přepracovaná klávesnice
- Zlepšená akcelerace 2D a 3D grafiky
- Podpora více procesorů
- Podpora USB klávesnic, …

Citováno z [31].

PŘÍLOHA P II: OS IOS - JEDNOTLIVÉ VERZE A ODLIŠNOSTI

Verze 4.0

Datum: Duben 2010
Novinky:

- Nové jméno
- Podpora multitaskingu
- Speciální funkcionality pro business sféru
- Vypuštěn tři dny před vydáním iPhone 4
- Verze 4.0.1 a 4.0.2 opravovaly chyby 4.0

Verze 4.1

Datum: Září 2010
Novinky:

- Opravy chyb
- Prodloužena životnost baterie
- Přidáno Game Centrum
- Nové nativní aplikace

Verze 4.2

Datum: Listopad 2010
Novinky:

- Opava závažné chyby WiFi
- 4.2.1 a 4.2.5 přinesly první podporu iPad zařízení, CDMA podporu

Verze 4.3

Datum: Březen 2011
Novinky:

- Vydán dva dny před zveřejněním iPad 2
- Nitro JavaScript pro Safari (až dvojnásobné zrychlení)
- iTunes Home Sharing (streamování obsahu iOS do reproduktorů počítače, TV apod.)
- Personal HotSpot

Citováno z [7].

PŘÍLOHA P III: SEZNAM ZÁKLADNÍCH NUTNÝCH ÚDAJŮ PRO PROJEKT

Specifikace systému by v každém případě měla obsahovat toto:

Číslo	**Název**	**Příklad**
1.	Typ OS	Android, iOS, Symbian, …
2.	Verze OS	Android 2.3, iOS 4.2.1
3.	Výrobce zařízení	HTC, LG, Sony Ericsson, …
4.	Model zařízení	HTC Legend, LG Optimus One, …
5.	Cílová země/jazyk	Velká Británie/EN, Česká republika/CZ, …
6.	Externí zdroje	GPS, Gyroskop, Fotoaparát, …
7.	Síťová optimalizace	WiFi, 3G, GPRS, …
8.	Velikost displeje/rozlišení	3,2“, 320x480, HVGA, …
9.	Otáčení aplikace při otáčení zařízení	Ano/Ne
10.	Způsob distribuce	Internet, FTP, Bluetooth, Paměťová karta, …
11.	Vzhled aplikace	Velikost tlačítek, Barevnost, Pozadí aplikace, Fonty, Velikost a rozlišení obrázků, …
12.	Vlastnosti cílového prostředí	Hluk, Tma, Vibrace, …
13.	Využití aplikací třetích stran	Facebook, Twitter, ICQ, Skype, …
14.	Využití bezpečnostních certifikátů	Pro bankovní služby, …
15.	Typ zpracovávaných dat	Statická/dynamická, Podporované formáty, …
16.	Je nutná SIM pro provoz?	Ano/Ne
17.	Opt.: Doplňující konfigurace zařízení	CPU, Paměť, Baterie, Displej, Velikost vnitřní paměti, …

PŘÍLOHA P IV: NA CO SI DÁVAT POZOR ANEB CO BY SE MĚLO TESTOVAT

Testování aplikace při …			**Typ testu**
Přerušení aplikace	Propojení mezi aplikacemi		Interrupt Test
	Hovor, SMS, slabá baterie, budík		Interrupt Test
	Vypnutí telefonu		Interrupt Test
	Minimalizace na pozadí		Interrupt Test
	Připojení/odpojení nabíječky		Interrupt Test
Přerušení kontaktu s externími zdroji	GPS, Gyroskop, Bluetooth		Interrupt Test
	Internet	Stahování	Network Test
		Ověřování kreditní karty	Network Test
		(Rychlost aplikace)	Network Test
	Operátorova síť		Network Test
„Look and Feel“	Gestikulace		Usability Test
	Klikatelnost linků		Usability Test
	Použití prstů, pointerů, kláves		Usability Test
	Dotaz při pokusu o připojení na net		Usability Test
	Ozvučení jen tam, kde k něčemu je		Usability Test
	Potvrzení při nevratných akcích		Usability Test
	Vzhled menu		Usability Test
	Vybrání poslední aktuální hodnoty při použití dropdown listu		Usability Test
	Zobrazení	Obrázky (rozlišení dle specifikace)	Usability Test
		Fonty	Usability Test
		Konzistentní tlačítka a barvy	Usability Test
		Zarovnání textu	Usability Test
	Upozornění a zprávy přes celý displej		Usability Test
	Minimum textu		Usability Test
	Nepoužívat psaní kapitálkami		Usability Test
Změna verze firmware	Podpora různých verzí OS		Compatibility Test
	Podpora různých verzí prohlížečů/přehrávačů		Compatibility Test
	Podporované formáty		Compatibility Test
	Změna nastavení telefonu	Klávesnice	Compatibility Test
		Jazyk	Compatibility Test

		Rozlšení	Compatibility Test
		Orientace displeje	Compatibility Test
		Změna času/časového pásma	Compatibility Test
	Umístění v telefonu	Instalace/Odinstalace	Compatibility Test
		Smazání aplikace	Compatibility Test
		Ukončení aplikace	Compatibility Test
	Možnost stáhnutí aplikace skrze app manager		Compatibility Test
Výkon	Plynulost aplikace při zátěži i bez		Performance Test
	Spotřeba operační paměti/baterie		Performance Test
	„Monkey“testy		Performance Test
Ostatní	Kontrola textu na překlepy		Functional Test
	Všechny vstupy validovány		Functional Test
	Správný sled obrazovek menu		Functional Test
	Kontrola notifikací v aplikaci		Functional Test
	Kontrola speciálních znaků		Functional Test
	Kontrola změny velikosti obrázku (například při načítání z kamery)		Functional Test

PŘÍLOHA P V: APLIKACE THINK FREE OFFICE – TEST DESIGN A TEST REPORT

Název TS	ČísloTestSuity: NázevTestSuity
Popis	Krátký popis dané Test Suity
VÝSLEDEK	PASS / FAIL
Nalezené chyby	ČísloSPR: Popis (Major/Minor/Cosmetic) – VD(virtualDevice)/RD(realDevice)

Výsledky testování ThinkFree Office verze v2.0.1110.01 na telefonu LG Optimus One (Android 2.2 Froyo, tovární nastavení) a virtuální emulátoru AVD od Google.

Název TS	TS1: Instalace/Odinstalace
Popis	Test úspěšné instalace a odinstalace
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS2: Vzhled jednotlivých obrazovek
Popis	Test, zda se jednotlivé obrazovky menu správně vykreslují a mají správný sled
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS3: Nastavení aplikace
Popis	Test změn v nastavení aplikace
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS4: Funkčnost tlačítek telefonu
Popis	Test správné funkčnosti jednotlivých tlačítek telefonu na různých obrazovkách
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS5: Procházení paměti telefonu
Popis	Test file manageru na listování adresářovou strukturou, řazení, správa složek
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS6: Rozbalování zip archivů
Popis	Testy na schopnost rozbalovat ZIP archivy
VÝSLEDEK	**FAIL**
Nalezené chyby	SPR1: Problém při rozbalování rozdělených archivů (MINOR) – RD,VD

Název TS	TS7.1: ThinkFree Write – otevření a čtení dokumentu
Popis	Test na otevření a procházení dokumentem (různé formáty), zoom, listování
VÝSLEDEK	**FAIL**
Nalezené chyby	SPR2: Rozhazuje formátování dokumentu (MAJOR) – RD SPR3: Občas nenačítá velké obrázky, které jsou přes celou stranu (MINOR) - RD

Název TS	TS7.2: ThinkFree Write – nabídková lišta
Popis	Otestování zobrazení nabídkové lišty, její funkčnosti, vzhledu
VÝSLEDEK	**PASS**

Nalezené chyby	žádné

Název TS	TS7.3: ThinkFree Write – vyhledávání textu
Popis	Otestování schopnosti programu vyhledávat daný text v dokumentu
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS8.1: ThinkFree Calc – otevření a čtení dokumentu
Popis	Test na otevření dokumentu (různé formáty), jeho procházení, přepínaní mezi listy, sortování tabulek, zoom
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS8.2: ThinkFree Calc – nabídková lišta
Popis	Otestování zobrazení nabídkové lišty, její funkčnosti, vzhledu
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS8.3: ThinkFree Calc – vyhledávání textu
Popis	Otestování schopnosti programu vyhledávat daný text v tabulce
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS9.1: ThinkFree Show – otevření a čtení prezentace
Popis	Test na otevření dokumentu (různé formáty), jeho procházení, přepínaní mezi slides, zoom
VÝSLEDEK	**FAIL**
Nalezené chyby	SPR4: Při zoomu rozhazuje grafiku prezentace (MINOR) - RD

Název TS	TS9.2: ThinkFree Show – nabídková lišta
Popis	Otestování zobrazení nabídkové lisšty, její funkčnosti, vzhledu
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS10.1: ThinkFree PDF – otevření a čtení PDF
Popis	Test na otevření dokumentu (různé formáty), jeho procházení, přepínaní mezi pohledy, zoom
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS10.2: ThinkFree PDF – vyhledávání textu
Popis	Otestování schopnosti programu vyhledávat daný text v PDF
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS11: Přerušení aplikace
Popis	Test na stabilitu aplikace při přerušení příchozím hovorem, SMS, budíkem, připojení/odpojení nabíječky, při minimalizaci na pozadí
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS12: Rychlost aplikace
Popis	Test, zda se aplikace nezasekává při práci s většími dokumentů či při více zatíženém zařízení
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS13: Test změny nastavení přístroje
Popis	Změna jazyka, změna typu klávesnice
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS14: Gesta
Popis	Test funkčnosti gest při listování dokumentem – slide, tap, …
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS15: Potvrzení nevratných akcí
Popis	Potvrzení při vymazání dokumentu ze zařízení
VÝSLEDEK	**FAIL**
Nalezené chyby	SPR5: Při smazání dokumentů nevyskočí potvrzovací okno (MINOR) – RD,VD

Název TS	TS16: Usabilty testy
Popis	Minimum textu na stránce, bez použití kapitálek, zarovnání na čitelnou plochu
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS17: Spotřeba
Popis	Měření spotřeby energie při práci s aplikací
VÝSLEDEK	**PASS**
Nalezené chyby	žádné

Název TS	TS18: Ostatní dokumenty
Popis	Pokus o otevření neznámých dokumentů, poškozených dokumentů a dokumentů jiných znakových sad
VÝSLEDEK	**FAIL**
Nalezené chyby	SPR6: Pád (zamrznutí) aplikace při otevírání poškozených souborů (MAJOR) – RD,VD

Popis typů chyb:

FATAL – Velmi závažná chyba, která způsobuje nefunkčnost (pád či zamrznutí). Aplikace není schopná vykonávat svůj hlavní účel.
MAJOR – Závažná chyba, kdy některá operace s aplikací končí výjimkou či vážně omezí funkčnost celé aplikace nebo její části.
MINOR – Jedná se o běžnou chybu, kdy některá funkcionalita se chová mimo definovanou specifikaci aplikace. Ostatní části aplikace jí nejsou vážně ohroženy.
COSMETIC – Kosmetická chyba - vyskytuje se převážně v GUI aplikace (například ve formě překrývajících se grafických prvků). Nijak neovlivňuje funkčnost aplikace.

Popis výsledků testů:

PASS - Všechny testy v rámci TS proběhly bez nalezení chyby
FAIL – Během testů v rámci TS byla nalezena jedna nebo více chyb