Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Reporty používané v nástrojích pro řízení testů

Michal Grolmus

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Michal Grolmus

Informatika

Název práce

Reporty používané v nástrojích pro řízení testů

Název anglicky

Reports used in test management tools

Cíle práce

Cíle práce: Cílem diplomové práce je identifikovat silná a slabá místa v reportech nástrojů pro řízení testů a získané informace využít ke zlepšení open-source nástroje TestLink.
Obsah práce:
1. Představení testování jako součásti vývoje softwaru.
2. Seznámení s jednotlivými metodikami testování SWPorovnání nástrojů pro řízení testů a jimi používaných reportů.
3. Vyhodnocení kladů a záporů reportingu nástrojů na základě interview s test managery
4. Výběr nejdůležitějších reportů z výsledků vyhodnocení interview a teoretických poznatků
5. Implementace vybraných reportů do nástroje TestLink
6. Sepsání dokumentace k novým reportům

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Praktická část je zaměřena na návrh a rozšíření reportingového modulu nástroje TestLink. Na základě syntézy teoretických poznatků a výsledků vlastního řešení budou formulovány závěry diplomové práce.

Doporučený rozsah práce

60-80 stran

Klíčová slova

testování SW, řízení testů, test report, TestLink

Doporučené zdroje informací

Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press, 2008, 322 s. ISBN 978-0-511-39330-3

Farrell-Vinay, P.: Manage Software Testing. Auerbach Publications, 2008, 537 s., ISBN 978-0-8493-9383-9.

Kaner, C., James, B., Pettichord, B.: Lessons Learned in Software Testing: A Context-Driven Approach. Wiley Computer Publishing, 2002, 286 s., ISBN 0-471-08112-4.

Marick, B.: The Craft Of Software Testing, Subsystem Testing, Prentice Hall PTR, 1995, ISBN 0-13-177411-5.

Myers, G. J.: The Art of Software Testing, 2. vydání. John Wiley & Sons, 2004, 234 s., ISBN 0-471-46912-2.

Spillner, A., Linz, T., Schaefer, H.: Software Testing Foundations, 2. vydání, 2007, 296 s., ISBN 978-1-9339-5278-9.

Předběžný termín obhajoby

2017/18 ZS – PEF (únor 2018)

Vedoucí práce

Ing. Josef Pavlíček, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Konzultant

Ing. Petra Pavlíčková Ph.D.

Elektronicky schváleno dne 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 27. 11. 2017

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Reporty používané v nástrojích pro řízení testů" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. 11. 2017 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu bakalářské práce Ing. Josefu Pavlíčkovi, Ph.D. a odborné konzultantce Ing. Petře Pavlíčkové, Ph.D. za vstřícný přístup a cenné rady.

Reporty používané v nástrojích pro řízení testů

Reports used in test management tools

Souhrn

Diplomová práce se zabývá reporty využívanými v nástrojích pro řízení testů. Na základě studia a analýzy odborných informačních zdrojů v teoretické části jsou získány poznatky o nástrojích, reportech a metrikách užívaných při testování a řízení testů. Tyto informace jsou následně využity v praktické části k formulaci relevantních otázek, cílených na uživatele nástrojů pro řízení testů, za účelem zjištění požadovaných vylepšení reportingového modulu nástroje TestLink. Na základě získaných poznatků je nástroj rozšířen o chybějící reporty.

Klíčová slova: testování SW, řízení testů, test report, TestLink.

Summary

This diploma thesis follows up reports used in test management tools. Based on the study and analysis of professional information sources in the theoretical part, knowledge about the tools, reports and metrics used in test and test management is obtained. This information is then used in the practical part to formulate relevant questions addressed to the users of the test management tools to identify the required improvements to the TestLink reporting module. Based on the findings, the tool is expanded by missing reports.

Keywords: SW testing, test management, test report, TestLink.

Obsah

1 Úvod

Přestože se počítačům daří vykonávat čím dál tím více činností dříve zastávaných výhradně člověkem, stále se najdou úkony, kde je lidská práce nenahraditelná. Nejinak je tomu i u testování softwaru. Při opakovaném vykonávání testů je bezesporu rychlejší a méně náchylný k chybám počítač provádějící automatizované testy. Pro testovací případy, které je potřeba zkontrolovat jen jednou nebo dvakrát, se však nevyplatí takové testy psát. Je tedy jednodušší a většinou i levnější, když tyto testy provede manuální tester. Při rozsahu dnešních softwarových produktů jsou ale potřeba desítky takových testerů, kterým musí test manažer efektivně rozdělit práci a také je při ní kontrolovat.

S tím mu mají pomoci nástroje pro řízení testů, jejichž úkolem je sbírat data o průběhu testů a pomocí srozumitelné formy je manažerům předkládat. Takovou formou nejsou tabulky plné čísel, neboť v nich člověk jen těžko najde nějaký vzorec nebo trend. Daleko srozumitelnější jsou data vizualizovaná pomocí grafů.

TestLink je open-source nástroj který se od roku 2011 snaží přesně takovým způsobem pomáhat test manažerům po celém světě. Jeho kód byl od té doby stažen již téměř 600000krát, čímž se řadí mezi jedny z nejoblíbenějších nástrojů pro řízení testů (SOURCEFORGE, 2017). I na tak oblíbeném nástroji se však dá vždy něco vylepšit. Jak prohlásil sám zakladatel TestLinku Francisco Mancardi v rozhovoru na serveru Bitnami: „Myslím, že reportingový modul potřebuje více práce“ (BITNAMI, 2014).

2 Cíl práce a metodika

Hlavním cílem práce je rozšířit reportingový modul nástroje TestLink užitečnými reporty. Současně jsou stanoveny i dílčí cíle, mezi které patří:

- představení testování jako součásti vývoje softwaru;
- seznámení s jednotlivými metodikami testování SW;
- porovnání nejpoužívanějších nástrojů pro řízení testů a jimi používaných reportů;
- výběr nejdůležitějších reportů k implementaci;
- sepsání dokumentace k novým reportům.

V teoretické části práce bude kromě úvodu do problematiky testování softwaru a řízení testů provedena také důkladná analýza typů nástrojů používaných při testování, s důrazem na skupinu nástrojů pro řízení testů. Studium odborných informačních zdrojů navíc poskytne informace o reportech a metrikách využívaných v těchto nástrojích.

V praktické části práce, na základě syntézy teoretických poznatků a výsledků dotazníkového šetření, komunikace s uživateli nástroje a interview s test manažerem, dojde k porovnání nástrojů, jejichž reporting je považován za nejlepší a k výběru nejužitečnějších reportů. Následně bude vybrána vhodná knihovna pro vizualizaci dat a proběhne samotná implementace reportů do nástroje. Na závěr bude doporučeno další možné rozšíření reportingového modulu.

3 Teoretická východiska

3.1 Testování jako součást životního cyklu softwaru

Dnes je již testování standardní součástí životního cyklu softwaru a jeho jednotlivé fáze pokrývá od začátku až do konce. Existuje mnoho přístupů k vývoji, které se nazývají vývojové procesní modely a patří mezi ně například vodopádový, iterativní, spirálový nebo agilní model. I přes značné rozdíly mezi modely může být životní cyklus zpravidla rozdělen na šest fází: sběr a analýza požadavků, návrh architektury, implementace, testování, předání k užívání a údržba (EVERETT a MCLEOD, 2007).

V první fázi je potřeba zjistit, kdo a jak bude software používat, jaká data do něj budou vstupovat, z něj vystupovat a podobně. Tím budou nalezeny požadavky na systém, které musejí být dále analyzovány a podrobně specifikovány. Výstupem takového úsilí bývá dokument Specifikace požadavků. S tímto dokumentem již přijde do styku testovací tým a začíná s plánováním testů.

Druhá fáze slouží k návrhu architektury softwaru, případně k určení nároků na hardware, dle požadavků specifikovaných v předchozí fázi. Výstupem je specifikace návrhu systému. Testovací tým v tu dobu připravuje strategii testů, která zodpoví otázky typu: „Co a jak bude testováno?“.

Ve třetí fázi se na základě návrhu systému začíná se skutečnou implementací softwaru. Ten je rozdělen na moduly nebo třídy a postupně kódován vývojářským týmem. Tato fáze bývá zpravidla nejdelší, testovací analytici tak mají dost času na specifikaci detailních testů.

Čtvrtá fáze závisí silně na testovacím týmu. Jakmile je software naprogramován, testeři začínají spouštět definované testy a zaznamenávají nalezené chyby. Vývojový tým se snaží o jejich opravu a předání testerům zpět k přetestování. Během této fáze využívají testovací metodiky popsané v následující kapitole.

V páté fázi, tedy po úspěšném dokončení testů a opravě všech chyb, může být výsledný software předán k užívání koncovým zákazníkům. Ti si většinou software znovu prověří takzvanými akceptačními testy a v případě nálezu chyb vrátí k opravě.

Poslední šestá fáze se zabývá údržbou softwaru již používaného zákazníky. V této fázi je stále ještě možné, že se nějaké chyby objeví. I zde se tedy testovací tým musí zapojit a přetestovat nahlášené chyby a jejich opravy.

3.2 Metodiky testování

Jak již bylo řečeno v minulé kapitole, ve čtvrté fázi životního cyklu softwaru se využívají různé metodiky testování. Základní rozdělení těchto metodik je funkční a nefunkční testování. Za funkční testování se považují jednotkové, integrační, systémové a akceptační testy. Za nefunkční se pak pokládají výkonnostní a bezpečnostní testy, testy použitelnosti a kompatibility.

Jednotkové testy kontrolují nejmenší samostatné části softwaru, tedy například třídy, komponenty nebo moduly. Tyto testy si většinou píší sami vývojáři a při využití modelu vývoje softwaru řízeného testy se dokonce vytvářejí ještě před samotnou testovanou jednotkou. Integrační testy jsou používány ke kontrole spolupráce komponent a modulů. Pro tento druh testů se většinou využívají manuální i automatické testy v závislosti na složitosti komunikace mezi jednotkami. Systémové testy pak kontrolují funkčnost systému jako celku. V tomto typu testů se nejčastěji využívá řízení testů kvůli zvýšené aktivitě testerů při manuálním testování navržených testovacích případů. V popisovaném stavu se již většinou netestuje kód, ale pouze chování aplikace z pohledu koncového zákazníka. Jako poslední krok funkčního testování přichází akceptační testy vedené zástupci koncových zákazníků.

Výkonnostní testy kontrolují chování aplikace pod zvýšeným náporem dat nebo uživatelů. Také mohou sloužit k měření požadované rychlosti odezvy. Bezpečnostní testy ověřují znemožnění přístupu do aplikace neautorizovaným uživatelům. Testy použitelnosti pomáhají odhalit chyby v uživatelském rozhraní, které způsobují neintuitivní ovládání softwaru nebo neefektivnost a chybovost při využívání aplikace. Testy kompatibility mají za úkol ověřit, že je vyvinutý software možné bezchybně používat na všech určených platformách, operačních systémech, ve všech definovaných prohlížečích a na displejích všech požadovaných rozlišení.

3.3 Řízení testů

Řízení testů je velmi důležitá činnost, umožňující efektivní čerpání zdrojů, které jsou vždy omezené. Navíc je potřeba testy efektivně koordinovat, aby došlo k dodržení všech stanovených milníků.

3.3.1 Stupně nezávislosti

Každý člověk tvoří relativně slepě, nevnímajíc své chyby. Proto je více než důležité, aby byla práce vývojářů otestována. Níže jsou předloženy v bodech jednotlivé stupně nezávislosti, jež označují různou úroveň ovlivnění testů vývojem:

- nejnižší stupeň nezávislosti označuje test prováděný tou samou osobou, která navrhovala nebo vyvíjela danou část kódu;
- o něco více nezávislé testy jsou prováděny jinou osobou nespecializující se na testování, která ale patří do stejného vývojového týmu, například při párovém programování;
- dalším stupněm jsou testy prováděné jednotlivcem nebo menší skupinou specializovaných testerů přiřazených do vývojového týmu, s takovým stylem organizace se můžeme setkat v agilních týmech nebo při vývoji dle metodiky Scrum;
- vyšší stupeň nezávislosti je dosažen oddělením testovacího týmu od vývojového v rámci té samé organizace;
- za nejvyšší stupeň samostatnosti je považováno oddělení testovacího týmu do samostatné ekonomické entity nebo využití firem specializujících se na testování.

Důvody pro výběr určitého stupně nezávislosti je dle Borise Beizera například ochrana testerů nebo poskytnutí určitého stupně objektivity (BEIZER, 1995). Nezávislé testovací týmy mají také několik nevýhod. Přenesením odpovědnosti za kvalitu na testovací tým mohou vývojáři ztratit zájem o samostatné kontrolování kvality vlastní práce. Vyčleněním speciálního testovacího týmu mohou vzniknout komunikační problémy a časové prodlevy. Každý stupeň má své výhody i nevýhody, je tedy vždy důležité provést správný výběr dle specifik plánovaného projektu.

3.3.2 Role a odpovědnost

Na testovacím projektu můžeme rozlišit dvě hlavní role: test manažer a tester.

Test manažer

Role test manažera pokrývá plánování testů, kontrolu průběhu testů a reporting výsledků. V týmech menší velikosti se test manažer často zabývá i analýzou požadavků a tvorbou testů (HASS, 2014). Jeho hlavní náplň práce se tak týká:

- koordinace testovací strategie a plánu testů s manažery vývoje a dalšími zainteresovanými osobami;
- definice a přizpůsobení testovací strategie potřebám projektu a dodržení testovacích pravidel dané organizace;
- poskytnutí stanovisek z pohledu testovacího týmu s cílem přizpůsobit testování naplánovaným projektovým aktivitám a opačně, například integraci komponent s integračními testy;
- organizace testů při zvážení kontextu a rizik, odhad potřebného času, úsilí a nákladů za testy, stanovení délky testovacích cyklů, výběr vhodných testovacích metod a postupů, naplánování řízení opravy chyb;
- specifikace, přípravy a implementace testů;
- kontroly a vyhodnocení průběhu testů;
- provádění změn v testovacím plánu v závislosti na průběhu a výsledcích dílčích testů nebo v případě zjištěných problémů;
- výběru testovacích nástrojů a organizace zaškolení testerů v jejich používání,
- rozhodnutí o využití a rozsahu automatizovaných testů;
- navržení vhodných měr pro vyhodnocení kvality testovaného produktu;
- vytvoření reportů a shrnujících prezentací založených na získaných výstupech z testování.

Tester

Úkolem testera je generování testů, například převodem požadavků na testovací případy, vytváření testovacích dat a popis očekávaných výstupů, provádění testů a kontrola shody mezi očekávaným skutečným výsledkem testů. V případě, že se výsledky neshodují, založí tester záznam o chybě. Hlavní náplní práce je tedy:

- přezkum plánu testů a pomoc s jeho tvorbou, což je potřeba hlavně kvůli minimalizaci výskytu nerealistických očekávání;
- analýza, přezkum a vyhodnocení možnosti otestovat požadavky, specifikace a modely;
- implementace testovacího prostředí ve spolupráci se systémovým administrátorem a správci sítě;
- získání nebo příprava testovacích dat;
- provádění testů, zaznamenávání a vyhodnocování výsledků;
- využívání nástrojů pro podporu testování;
- vytváření automatizovaných testů, většinou za pomoci programátorů nebo specialistů zaměřujících se na automatizované testy;
- měření výkonu jednotlivých komponent i celého systému;
- přezkum testů navržených ostatními testery.

Uvedené činnosti nemusí být vždy prováděny jen testery z testovacího týmu. Vývojáři mohou například provádět integrační testy a uživatelé zákazníka testy akceptační.

Testovací tým

Pracovní náplň jednotlivých členů malých týmů může být promíchána z aktivit obou rolí. Naopak ve větších týmech je specializace jednotlivých členů daleko znatelnější. Lze tak rozpoznat další role:

- test analytik – zaměřuje se na funkční nebo nefunkční aspekty testování;
- technický test analytik – zabývá se testy výkonnosti, bezpečnosti nebo škálovatelnosti;
- databázový specialista – připravuje složitá testovací data;
- business specialista – kontroluje správné pokrytí požadavků testovacími případy;
- tester použitelnosti – stará se o uživatelské testování grafického rozhraní.

3.3.3 Plánování a odhady

Test manažeři ve spolupráci s projektovými manažery, případně s vedoucími jednotlivých testovacích týmů, musí odhadnout pracnost. Dle ní zorganizovat a naplánovat testovací aktivity včetně určení potřebných zdrojů a odůvodnění jejich pořízení či využití. Tento odhad je navíc prováděn vzhledem ke stále se měnícímu cíli (MAIDASANI, 2007).

Odhad pracnosti

Odhad pracnosti může být ovlivněn několika typy faktorů. Faktory spojené s testovacím procesem představují problémy s průběhem předchozích testovacích fází, změnami v požadavcích nebo vyšší než předpokládanou mírou chybovosti aplikace. Faktory hardwarové se vztahují k nástrojům pro podporu testů nebo testovacímu prostředí. Z pohledu lidských zdrojů ovlivňují pracnost schopnosti testerů a představy o jejich výkonnosti, podpora od vývojového týmu a vztahy mezi týmy. Mezi další faktory lze zařadit komplexnost softwaru, velké množství zainteresovaných entit, příliš mnoho funkcí nebo vícejazyčné distribuce. Důležitý je také celkový přehled o všech aspektech, které mohou vývoj aplikace ovlivnit a dostatečná znalost metod používaných pro odhad (MILI a TCHIER, 2015).

Metody pro odhad

Metody pro odhad bývají postaveny na základě metrik nebo zkušeností. Odhady založené na metrikách využívají statistické poznatky a výpočty v závislosti na již realizovaných projektech. Metoda COCOMO II (Constructive Cost Model) při zadání několika parametrů projektu vrátí odhad pracnosti, předpokládané trvání vývoje i přibližný počet řádků kódu. Na základě odhadu vývoje lze s přihlédnutím ke specifikům testovacích aktivit lehce odhadnout i pracnost testů (BOEHM, ABTS a kol., 2000). Odhady založené na počtu řádků kódu však mohou být značně nepřesné, neboť nepočítají s rozdíly mezi programovacími jazyky ani stylem programování jednotlivých vývojářů. Tuto nevýhodu se snaží minimalizovat odhady založené na funkčních jednotkách, tedy především na počtu funkcí, které musejí být vyvinuty. Poznatky z historických projektů pak ukazují, že na jednu funkci připadá dle typu projektu čtyři až sedm chyb (JONES, 2007). Při práci s uvedenými metodami se však vždy muselo počítat s určitou nepřesností při převodu odhadu vývoje na test. Proto byla představena metoda určená přímo pro odhad pracnosti systémových nebo akceptačních testů pod zkratkou TPA (Test Point Analysis), která počítá i s váhou a riziky jednotlivých testů (VAN VEENENDAAL, POL a kol., 2002).

Odhady založené na zkušenosti nepočítají s žádnými metrikami, ale pouze se zkušeností jednotlivých členů týmu zodpovědných za jim přidělené testovací aktivity. Takovým přístupem je eliminována možnost špatného použití metrik. Nejjednodušší metodou je prostý součet odhadů časů jednotlivých aktivit. Nevýhodou je jen velmi přibližná představa každého člena o celkové pracnosti. Vždy je proto k součtu odhadů připočítána

určitá časová rezerva. Další metoda požaduje po každém jednotlivci tři odhady: pesimistický, optimistický a ten, který by se měl nejvíce blížit skutečnosti. Jiné metody spoléhají na podobnost s již realizovanými projekty. Vždy je však potřeba přihlédnout k tomu, zda neprošel tým zásadní personální změnou nebo zda se nezměnilo testovací prostředí (CHEMUTURI, 2009).

Plánování testů

Při plánování testů je potřeba rozhodnout, co bude testováno, které role budou vykonávat jaké aktivity, kdy a jak budou tyto aktivity prováděny, jak se vyhodnotí výsledky testů a kdy se testování ukončí, k čemuž je důležité stanovit výstupní kritéria. Dále je nutné přiřadit definované úlohy a testy členům týmu včetně dalších potřebných zdrojů. Musí však být brána v potaz specializace každého jednotlivce a také jeho možná nedostupnost v případě čerpání dovolené nebo možného onemocnění.

3.3.4 Stanovení kritérií

Při organizování testovacích aktivit musí test manažer také naplánovat, kdy aktivita začne a kdy skončí. Podmínky pro začátek aktivity se nazývají vstupní kritéria a pro skončení výstupní kritéria (BLACK, 2009).

Vstupní kritéria

Tato kritéria definují dostupnost:

- adekvátní dokumentace ve formě požadavků, návrhu nebo manuálů, které umožní testerům pochopit očekávané chování testovaných komponent;
- testovaných komponent v odpovídající kvalitě, tedy po úspěšném dokončení předchozích fází vývoje nebo testů;
- testovacího prostředí a všech potřebných nástrojů plně využitelných testery;
- připravených testovacích skriptů;
- potřebných testovacích dat.

Výstupní kritéria

Výstupní kritéria definují, za jakých podmínek může být daná testovací aktivita považována za dokončenou a kdy lze komponenty podléhající testu předat do další fáze testů. Podmínky jsou většinou stanoveny ve formě metrik. Může se jednat o poměr mezi

úspěšně a neúspěšně provedenými testy nebo o stav pokrytí požadavků. Výstupní kritéria se dle velikosti projektu stanovují pro jednotlivé úrovně testů, jednotlivé testovací aktivity nebo pro celý systém. Nedodržení výstupních podmínek znamená nepředání dostatečně kvalitního softwaru, což povede v následujících fázích ke snížení efektivity díky vyššímu počtu chyb, které budou muset být opraveny a současně ke zvýšení nákladů. Při nedodržení kritérií u posledních fází, tedy u systémových nebo akceptačních testů, bude nekvalitní software doručen zákazníkům, čímž může dojít k finančním ztrátám. V těchto posledních fázích jsou proto vhodná tato kritéria:

- v průběhu posledních třech týdnů systémových testů nebudou provedeny žádné funkční změny v softwaru, vyjma změn spojených s opravou identifikovaných chyb;
- nedojde k zastavení nebo pádu žádného ze serverů nebo aplikací, nebude zaznamenána nestabilita softwaru;
- testovací tým dokončí všechny naplánované testy;
- vývojářský tým dokončí opravu všech závažných chyb, závažnost většinou určuje zákazník nebo oddělení prodeje či marketingu;
- stanovené metriky svědčí o stabilním a spolehlivém softwaru.

3.3.5 Řízení rizik

Riziko je pravděpodobnost výskytu události s negativními dopady. Takové události lze v rámci testovacích aktivit rozdělit na dva druhy. Rizika týkající se samotného softwaru, což jsou hlavně chyby a defekty, nebo průběhu projektu, například nedodržení naplánovaných termínů nebo překročení výdajů.

Naložit s riziky lze různými způsoby. Pokud neexistuje způsob, jak rizika omezit nebo jim čelit, případně je-li pravděpodobnost jejich výskytu nebo závažnost dopadu malá, lze hrozby přijmout včetně jejich následků. Pravděpodobnost výskytu rizik lze v některých případech zmírnit zavedením příslušných opatření. I v takovém případě je však nutné pokračovat v měření pravděpodobnosti výskytu rizik s cílem zhodnocení efektivity zmírňujících opatření. Stejně tak při použití nouzových opatření pro snížení dopadu rizika je potřeba dále zkoumat, zda musí být taková opatření nadále využívána (SICKINGER, 2010).

3.3.6 Řízení chyb

Hlavním úkolem testovacích aktivit je identifikace chyb softwaru, ale současně i zaručení vysoké kvality výsledného produktu. Nestačí tak chyby pouze nalézt, je důležité i zkontrolovat, že byly skutečně opraveny. Kvůli hladkému průběhu testů je navíc důležité chyby popsat dostatečně podrobně, aby s pochopením jejich výskytu neměli vývojáři problémy a dokázali je rychle lokalizovat a opravit.

Stejně jako při řízení rizik se i u chyb může vedení rozhodnout některé z nich neopravovat, pokud je jejich dopad minimální, nebo pokud jsou aktuálně jiné priority. Kvůli zajištění pozice na trhu tak často bývá uveden produkt s chybami, i za cenu budoucích volně dostupných opravných updatů.

Životní cyklus chyby lze zjednodušeně popsat jako tři na sebe navazující fáze. Ve fázi identifikace tester objevuje rozdíly mezi očekávaným a reálným výsledkem testu a posuzuje jejich příčinu a dopad. V akční fázi se určí, jaké postupy budou použity pro zpracování chyby. A v poslední fázi likvidační se rozhoduje o průběhu uzavření chybového hlášení (PUSULURI, 2008).

Chyby se mohou třídit dle různých kritérií: dopadu na uživatele, úsilí potřebného k opravě chyby, postihnuté části softwaru a dalších.

Identifikace chyby

Hlavní činností testovacích aktivit je srovnávání očekávaných a reálných výstupů získaných při kontrole softwaru. Pokud je mezi takovými výstupy objeven rozdíl, je důležité rozhodnout, zda se skutečně jedná o chybu, nebo o správnou funkčnost systému, která jen byla chybně popsána ve formě očekávaného výstupu. Při mylném rozhodnutí pak mohou vznikat chybová hlášení, jejichž popis považují vývojáři za správnou funkčnost. Mnoho nalezených chyb se tak nemusí týkat softwaru, ale samotného popisu testů, případně chybně připravených testovacích dat či testovacího prostředí.

Jakmile je chyba objevena, měla by být co nejrychleji reportována. Při popisu je vhodné minimalizovat počet kroků potřebných k jejímu opětovnému nasimulování. Na druhou stranu dostatečně detailní informace o původu chyby mohou s pomocí statistických nástrojů odhalit například shlukování chyb v nejvíce postižené komponentě nebo nejčastější zdroje chyb.

Reporting chyb

Chybové hlášení by na úvod mělo obsahovat unikátní identifikační kód a stručný popis, často jen ve formě názvu chyby. Pro určení priority opravy daného nálezu je vhodné popsat dopady chyby na uživatele a ostatní zainteresované osoby, případně na testovací dokumentaci. V detailním popisu chyby je pak potřeba uvést:

- použitá vstupní data, včetně požadovaných podmínek splněných před započetím testu,
- očekávaná výstupní data,
- skutečně získaná výstupní data z testu,
- přesný popis chyby,
- kroky testovacího scénáře a testovacího případu,
- přesný popis postupu nasimulování chyby,
- datum a čas nálezu chyby,
- nastavení testovacího prostředí, jak hardwarové tak softwarové,
- jména zúčastněných testerů.

Přesnost popisu chyby je velmi důležitá. Název části softwaru, ve kterém se chyba vyskytla, například titulek webového okna nebo název testované komponenty, musí přesně odpovídat. Stejně tak v případě chybové hlášky je nutné uvést její přesné znění.

Po upřesnění nálezu je vhodné se zamyslet nad podobností se zbytkem softwaru. Mohou tak být odhaleny chyby se stejnou strukturou i v jiných komponentách. Dále je potřeba určit, zda se jedná o první výskyt takové chyby, nebo zda již byla přítomna v předchozích verzích, ale nebyla objevena. Dle toho lze rozhodovat například o zlepšení navržených testů.

Řešení chyby

S chybovým hlášením je nakládáno podle jeho priority a způsobu opravy. Nejčastěji bývá uzavřeno ihned po opravě chyby. Může být však uzavřeno, aniž by došlo k opravě. Mezi takové případy patří například hlášení, která ve skutečnosti popisují správnou funkčnost systému a ne chybu, duplicitní hlášení nebo hlášení, která se týkají části softwaru od externích dodavatelů nebo zcela jiného projektu. Hlášení mohou být také odložena k opravě v příštích verzích softwaru, případně spojena s jinými chybami týkajícími se stejného problému nebo stejné části kódu.

3.3.7 Testovací dokumentace

Všechny testovací aktivity, jako je plánování testů, příprava testovacích dat, tvorba a provádění testů, ověřování a vyhodnocování nebo reportování výsledků, jsou podporovány psanou dokumentací. Na internetu lze najít mnoho šablon, jak by takové dokumenty měly vypadat. V této práci však budou představeny pouze dokumenty doporučené mezinárodní normou (IEEE 829, 2008).

Detailní testovací dokumentace má své výhody i nevýhody. Příliš podrobné dokumenty jsou málo flexibilní a při každé menší změně musejí být přepracovány, což se negativně promítne i na nákladech. Na druhou stranu s nimi dovedou pracovat i méně zkušení testeři, což může náklady snížit.

Níže jsou představeny typy dokumentů dle standardu. U každého z nich bývá doporučeno uvádět unikátní identifikátor umožňující na něj zpětně odkázat, včetně čísla verze. U rozsáhlejších dokumentů je vhodné přidat zkrácený úvod pro nastínění kontextu.

Plán testů

Doporučený obsah tohoto dokumentu se skládá z popisu elementů nebo objektů, které mají být otestovány, včetně jejich typu a charakteristik. Dále z popisu hlavních použitých testovacích principů, technik a metod, spolu s přiřazením do určitých testovacích aktivit nebo stupňů testů. Důležité je také stanovit kritéria úspěchu a neúspěchu, tedy popsat, dle jakých podmínek bude vyhodnocena úspěšnost naplánovaných testů. Není však důležitý jen konec testů, ale i jejich průběh, tedy stanovení podmínek, za kterých je lepší testy přerušit nebo naopak, kdy je možné s testováním pokračovat. Z pohledu zdrojů se v tomto dokumentu uvádí také požadavky na testovací prostředí, jak na hardware, tak softwarové nástroje. Součástí by měl být i popis rolí spolu s potřebným stupněm znalostí a dovedností včetně odpovědnosti členů týmu. V rámci plánu testů nesmí být opomenut ani přesný rozpis aktivit s daty začátků, jejich trváním a návaznostmi. Na závěr je vhodné sehnat souhlas všech zainteresovaných stran s obsahem plánu testů, například vývojový tým musí odsouhlasit, že do naplánovaného data stihne k testům předat software v požadované kvalitě.

Tvorba testů

Takových dokumentů může být i více, jsou detailnějším popisem aktivit a postupů uvedených v test plánu. Dopodrobna je v nich řešena testovací strategie, způsob tvorby testovacích případů, důvody jejich výběru nebo závislosti mezi nimi.

Testovací případy

Testovací případy se používají k jednoznačnému určení způsobu otestování komponent uvedených v dokumentech popisujících tvorbu testů. Obsahují určení vstupních hodnot, očekávaných výsledků včetně času odezvy softwaru, nároků na testovací prostředí, testovací data a verzi zkoumané aplikace nebo návaznost na ostatní testovací případy.

Postup testů

Postup testů, jindy nazývaný také jako testovací scénář, umožňuje spojovat testovací případy za cílem kontroly rozsáhlejších funkcí nebo jejich skupin. Dokument tedy obsahuje popis účelu tohoto spojení, předpoklady pro zahájení vybrané skupiny testů nebo dodatečné nároky na testovací prostředí.

Protokol o průběhu testů

Protokol neboli report, popisuje provedené testovací aktivity, jejich výsledky a poskytuje odhady založené na těchto výsledcích. Obsahuje detailní identifikaci testovaných komponent s odkazy na příslušnou testovací dokumentaci, včetně popisu rozdílů mezi plánovaným stavem komponent a skutečně testovaným stavem, s vysvětlením, proč k těmto rozdílům došlo. Stejně důležité je i uvedení těch částí nebo kombinací částí, které mohly být otestovány nedostatečně. Hlavním cílem tohoto dokumentu je však podrobný i souhrnný přehled výsledků testů, identifikovaných a vyřešených chyb, vyhodnocení všech testovacích aktivit a zdůvodnění splnění či nesplnění výstupních kritérií.

3.3.8 Monitorování a kontrola průběhu testů

Nejsnazší cestou jak vyhodnotit aktuální průběh testů je přesné změření stavu pomocí navržených metrik. V čase kontroly je znám naplánovaný počet testů, které měly být do daného data provedeny. Ten lze snadno porovnat s počtem již reálně provedených testů. Také jsou známy míry pokrytí rizik, požadavků i testovacích případů. Mimo objektivní míry

lze průběh testů posoudit i z hlediska subjektivního vnímání úrovně kvality softwaru zúčastněných testerů.

3.3.9 Reporting

Monitorování a kontrola průběhu testů by nebyla užitečná bez reportingu, díky němuž lze přehledně informovat všechny zainteresované strany o provedených testech a dát jim dostatek informací pro rozhodování o nadcházejících fázích vývoje. Je ale také potřeba umět rozlišit, jaké informace komu nebo kterým skupinám poskytnout. Například pro vývojáře a zákazníka nebude podstatný stejný druh informací.

Vývojový tým bude zajímat počet nalezených chyb, jejich závažnost a jimi postihnuté komponenty, data plánovaných začátků testovacích aktivit nebo stupeň kvality dodaných komponent, případně celého softwaru. Vedení projektu bude také zajímat stupeň kvality softwaru, navíc potřebuje vědět, zda naplánované využití zdrojů souhlasí s reálným využitím, či zda je potřeba plán upravit. To se může týkat nejen nedostatečné hardwarové infrastruktury, ale i podhodnocení počtu testerů. Zákazníky nebo uživatele bude mimo kvalitu softwaru zajímat i změna plánovaného data akceptačních testů, plánované spuštění na produkci nebo uvedení na trh, případně zpoždění v dodání určité funkcionality.

Reporting ale nesouvisí pouze s poskytováním informací ostatním zainteresovaným stranám, měl by předávat informace i členům testovacího týmu. Testeři nebo celé týmy testerů musí mít přehled o objemu zbývající práce, počtech nahlášených neopravených chyb nebo chyb opravených, určených k přetestování. Důležitou se stává i informace, která komponenta je nejvíce chybová, aby se upravil plán následných testovacích aktivit.

Ke správnému fungování testovacího týmu jsou důležité i reporty generované ostatními týmy zapojenými do vývoje. Od vývojového týmu bývají považovány za nejpodstatnější informace o plánovaných dodávkách komponent nebo změn ve funkčnosti či požadavcích na software. Při takových změnách stačí jen upravit odpovídající naplánované testy. Pokud se však jedná o zcela novou funkčnost, budou muset být vytvořeny nové testy a dojde k rozšíření plánu testů (DESIKAN a RAMESH, 2006).

3.4 Testovací nástroje

Dle slovníku testovacích pojmů certifikační organizace International Software Testing Qualifications Board (ISTQB) je testovací nástroj definován jako softwarový produkt, který podporuje jednu nebo více aktivit. Lze sem zahrnout: plánování a kontrolu, specifikaci, tvorbu počátečních souborů a dat, provádění testů a testovací analýzu. Obecně si lze testovací nástroj představit jako software, který je využíván k účinnějšímu, efektivnějšímu, snadnějšímu, rychlejšímu a přesnějšímu testování (ISTQB, 2014).

Testovací nástroje se používají z různých důvodů. Těmi hlavními však jsou: automatizování úloh vyžadujících strojovou preciznost a úloh pro člověka únavných nebo opakujících se, případně provádění úloh, které by člověk prováděl jen velmi přibližně, například testy rychlosti odezvy, které se pohybují v řádech setin sekund.

Nástroje mohou být použity k podpoře většiny testovacích aktivit, ať už je to tvorba testů, provádění testů, kontrola, reporting nebo celkové řízení testů. Obecně se dá říci, že pro jakýkoliv typ nástroje existuje jak jeho komerční verze, tak i nástroj s licencí open-source. U automatizovaných nástrojů většinou platí, že zvyšují spolehlivost testů, umožňují provádět testy bez přítomnosti lidí a zlepšují efektivitu opakovaných úloh.

Přesto ale použití nástrojů negarantuje zlepšení softwaru. Přidanou hodnotu totiž nástroj může poskytnout jen při úlohách, ke kterým je určen. Často se tak na projektu používá více nástrojů se specifickým použitím. U některých nástrojů je navíc jejich přínos patrný až po určitém počátečním úsilí, které by bez používání nástroje nemuselo být vynaloženo (DUSTIN, 2003).

3.4.1 Typy testovacích nástrojů

Testovací nástroje lze možné rozdělit podle různých kritérií, například podle:

- aktivity v rámci testovacího procesu nebo životního cyklu softwaru, ke které se vztahuje;
- typu testování, které podporují (manuální, automatizované,…);
- zdroje (komerční, shareware, open-source,…);
- použitých technologií;
- nebo kdo a s jakou testovací rolí je používá.

Tabulka 1: Typy testovacích nástrojů

Testovací nástroje	Aktivity	Uživatelé
Podpora pro statické testování		
Nástroje pro přezkum	Implementace a testování	Různí
Nástroje pro statickou analýzu	Implementace a testování	Vývojáři
Podpora pro přípravu testů		
Nástroje pro přípravu testovacích dat	Analýza a příprava testů	Testeři
Nástroje pro získání očekávaných výstupů	Analýza a příprava testů	Testeři
Nástroje pro tvorbu testů a generování test skriptů	Analýza a příprava testů	Testeři
Podpora pro provádění testů		
Nástroje pro porovnání výsledků testů	Implementace a testování	Testeři a vývojáři
Nástroje pro provádění testů	Implementace a testování	Testeři
Nástroje pro jednotkové testy	Implementace a testování	Vývojáři
Nástroje měřící pokrytí testů	Implementace a testování	Vývojáři
Nástroje pro bezpečnostní testy	Implementace a testování	Specialisté na bezpečnostní testy
Podpora pro kontrolu výkonnosti a údržbu		
Nástroje pro dynamickou analýzu	Implementace a testování	Vývojáři
Nástroje pro výkonnostní a zátěžové testy	Implementace a testování	Specialisté na výkonnostní testy
Nástroje pro monitorování	Implementace a testování	Různí
Nástroje pro testy použitelnosti	Implementace a testování	Specialisté na testy použitelnosti
Podpora pro řízení testů a testování		
Nástroje pro řízení testů	Všechny aktivity	Testeři
Nástroje pro správu zaznamenaných chyb	Implementace a testování	Testeři (+ jiní)
Nástroje pro řízení požadavků	Analýza a příprava testů	Business analytici (+ jiní)
Nástroje konfiguračního managementu	Implementace a testování	Různí

Zdroj: vlastní zpracování

3.4.2 Nástroje pro statické testování

Hlavním úkolem statického testování je zlepšit kvalitu softwaru především díky identifikaci chyb vývojářů v brzkém stádiu vývoje, kdy ještě není možné aplikaci spustit. Takové testování se tedy vztahuje především ke konvencím jmen proměnných, funkcí nebo rozhraní, k interní dokumentaci, pravidlům programování nebo jednotlivým částem softwaru. Mohou tak být prováděny ještě dlouho před využitím dynamických testovacích technik.

Velkou výhodou je také fakt, že takové testy bývají oproti dynamickým většinou levnější. Chyby jsou objeveny dříve, jejich opravy bývají levnější a nedochází tak v pozdějších fázích vývoje například k narušení práce více modulů najednou. Co víc, pokud jsou chyby objeveny již v dokumentech se specifikací softwaru, nemusí se ani do kódu dostat. To umožní testerům se při dynamickém testování více věnovat těm defektům, které mohou být nalezeny pouze při testování již funkční aplikace.

Návratnost investic přezkumu a statické analýzy je velmi vysoká. Proto může být překvapivé, že mnoho organizací nástroje pro tyto metody nepoužívá. Capers Jones uvádí, že například formální inspekce přispívají ke snížení celkových nákladů projektu a zkracují dobu vývoje o 15%, objem práce o 20% a v průměru je před dodáním softwaru identifikováno o 200% více chyb (JONES, 2007).

Nástroje pro přezkum

Tyto nástroje poskytují podporu při plánování a sledování přezkumu, pro komunikaci a kolektivní přezkum nebo slouží jako repositář. Standard IEEE 1028-2008 seskupuje techniky přezkumu dle míry formálnosti, od nejformálnějších auditů a inspekcí po méně formální ověření z pohledu managementu nebo technického ověření (IEEE 1028, 2008). Přezkum se většinou týká čtyř oblastí, a to ověřování:

- shody s dokumenty vyšší úrovně, dle kterých byli přezkoumávané dokumenty vytvořeny (smlouvy, specifikace, požadavky);
- s ohledem na projektové dokumenty stejné úrovně (specifikace testovacích případů, testovací data, zdrojové kódy, rozhraní);
- s ohledem na standardy a normy, doporučené postupy;
- s ohledem na koncové využití (zda není navržené řešení málo podrobné).

Nástroje pro statickou analýzu

Nástroje spadající do této kategorie používají převážně vývojáři pro kontrolu, zda je kód v souladu s platnými programovacími a jmennými pravidly. Mají jim pomoci:

- odhalit chyby v raném stádiu vývojového cyklu a snížit tak náklady na jejich opravu;
- zlepšit spolehlivost kódu identifikací těch částí, které nemohou být pokryty dynamickými testy;
- identifikovat nejproblémovější komponenty podléhající shlukování chyb, jenž by mohly být odpovědné za většinu defektů;
- nalézt nebo dokonce předpovědět problémy díky přehledné vizualizaci dat;
- usnadnit refaktorování díky identifikaci duplicitních struktur a částí kódu.

Typickými chybami, které pomáhají nástroje pro statickou analýzu odhalit, bývají odkazy na proměnné s nedefinovanou hodnotou, chybně definované proměnné nebo proměnné, které nejsou nikde použity, nekonzistentní rozhraní mezi moduly a komponentami, nedosažitelné části kódů (například za podmínkou, která nemůže nikdy nastat), nekonečné cykly, bezpečnostní zranitelnosti, syntaktické chyby nebo nedodržení jmenných konvencí (HUTCHENSON, 2003). Pro podchycení výše uvedených chyb nemusí stačit jeden nástroj. Naopak je většinou nutné použít více nástrojů v různých fázích vývoje:

- při testování jednotlivých komponent, tedy ještě před integrací se provede: analýza toku řízení, analýza toku dat, výpočet cyklomatické složitosti, validace syntaxe, ověření jmenných a programovacích pravidel, křížová kontrola odkazů na proměnné;
- během testování integrace komponent lze aplikovat: kontrolu konzistence rozhraní;
- při testování výsledného systému je vhodné uplatnit: schéma funkčních volání.

3.4.3 Nástroje pro přípravu testů

Nástroje pro přípravu testovacích dat

Pro některé typy testů, například test funkčnosti stránkovací navigace u rozsáhlých seznamů, je potřeba disponovat velkým množstvím dat, které by bylo časově náročné připravit manuálně. K tomu jsou určeny nástroje pro přípravu testovacích dat, které zvládnou

požadované množství dat vygenerovat náhodně nebo podle předlohy, případně stáhnout a upravit z již existující databáze. V případě využívání reálných dat většinou umožňují jejich anonymizování, aby se předešlo jejich zneužití (ASHFAQUE, 2010).

Nástroje pro získání očekávaných výstupů

Tyto nástroje dodávají testerům data správných očekávaných výstupů pro porovnání s výstupy testovaného softwaru. Skupina těchto nástrojů je velmi široká, nemusí se jednat o nástroje ve formě specializovaného softwaru. Při přechodu na nový produkt mohou testeři pro porovnání využít například výstupy ze starého produktu. Pokud jde o testování početních operací, mohou být výsledky porovnány s výsledky z ověřeného algoritmu (AGARWAL, TAYAL a kol., 2010).

Nástroje pro tvorbu testů a generování test skriptů

Nástroje mohou s tvorbou testů pomoci jak při návrhu vstupních a výstupních dat, tak se samotným určením testovacích případů. Vstupní data dokážou navrhnout z:

- popisu požadavků – rozmezí vstupních hodnot, chybové hlášky;
- již hotového kódu – jakou kombinací vstupů se dá dostat k volání které funkce;
- grafického rozhraní – typ vstupních hodnot formulářových polí.

Z grafického rozhraní lze také získat testovací případy. Nástroj rozhraní prozkoumá a navrhne testy pro všechny typy ovládacích prvků nebo zjistí nefunkční odkazy. Pro různé platformy a prohlížeče navrhne všechny jejich kombinace k otestování.

3.4.4 Nástroje pro provádění testů

Nástroje pro porovnání výsledků testů

Pomocí těchto nástrojů lze automatizovat porovnávání očekávaných a skutečných výsledků testů. Výhody dynamického porovnávání spočívají v možnosti kontrolovat výsledky při samotném běhu testu, tedy například mohou zkontrolovat textaci chybové hlášky, jakmile se objeví na obrazovce. Oproti tomu nástroje porovnávající zaznamenané výsledky až po ukončení testů jsou určené k porovnávání obsáhlejších dat, například celých vygenerovaných souborů (GRAHAM, VAN VEENENDAAL a kol., 2008).

Nástroje pro provádění testů

Některé manuální testy vyžadují provádění stále stejných úkonů dokola, například vyplňování dlouhých formulářů. Tyto činnosti lze lehce automatizovat pomocí nástrojů pro provádění testů. Většina nástrojů funguje na principu nahrání manuálního testu ve formě znovu spustitelného kódu. Takto nahrané testy lze následně využít k opakovanému spuštění testu, ke snadné úpravě vstupních dat a obměně testu, k porovnání nebo zaznamenání výsledků testů, případně k měření odezvy mezi jednotlivými kroky testu. Nevýhoda takto nahraných testů spočívá v závislosti na shodné podobě aplikace v době nahrání a spuštění testu. I malá změna ve vzhledu může znehodnotit velkou část zaznamenaných testů (EVERETT a MCLEOD, 2007).

Nástroje pro jednotkové testy

Při kontrole i těch nejmenších funkčních částí systému se lze vyhnout nálezům mnoha chyb v pozdějším cyklu vývoje. S takovým testováním pomáhají nástroje pro jednotkové testy. Za jednotky mohou být považovány nejen funkce nebo procedury, ale i celá rozhraní či třídy. Testy pro tyto jednotky většinou píší vývojáři. Nástroje je umožňují po čase opětovně automaticky spouštět a zaznamenávat jejich výsledky, podobně jako nahrané manuální testy (DIETRICH, 2014).

Nástroje měřící pokrytí testů

Tyto nástroje pomáhají určit důkladnost testů. Nejprve vyhledají a spočítají všechny testovatelné položky, například řádky kódu, větvení programu v podmínkách nebo volání funkcí. Nad programem se poté spustí sada testů a nástroj zaznamená, jakých položek se testy dotkly a jakých nikoliv, případně spočítá poměr otestovaných a neotestovaných položek.

Nástroje pro bezpečnostní testy

V době, kdy velká část aplikací běží na serverech připojených k internetu a mohou tak být snadno napadeny, je velmi důležité testovat i zabezpečení aplikací. S takovými testy pomohou nástroje pro bezpečnostní testy. Některé vyhledávají nezabezpečené porty nebo slabá hesla, jiné se snaží aplikaci nabourat a zaznamenávají výsledky pokusů o takové narušení.

3.4.5 Nástroje pro kontrolu výkonnosti a údržbu

Nástroje pro dynamickou analýzu

Dynamická analýza je způsob testování vyžadující spuštěný program. Nástroje, které s tímto typem testů pomáhají, zkoumají, co se děje při běhu programu. Kontrolují například zacházení s pamětí, zda je správně uvolňována a nedochází k přetížení. Některé nástroje také pomáhají s odhalením nefunkčních odkazů a nejčastěji se využívají při integračních testech.

Nástroje pro výkonnostní a zátěžové testy

Aby se uživatelům s aplikací dobře pracovalo, musí běžet plynule a bez výpadků. K zajištění takového stavu slouží nástroje, které testují předpokládanou zátěž ale i větší množství uživatelů ve špičkách. Mimo zátěž způsobenou uživateli testují i zatížení způsobené větším množstvím dat, například nahráním extrémně velkého souboru. Kromě testování poskytují nástroje také informace o průměrných časech odezvy nebo době načítání dat (MOLYNEAUX, 2009).

Nástroje pro monitorování

Pomocí těchto nástrojů lze předejít větším škodám způsobených chybou a to jejím co nejrychlejším zaznamenáním a následnou opravou. Nástroje pro monitorování zvládají kontrolu serverů, sítí, databází i samotných aplikací. Jejich hlavním úkolem by mělo být: zaznamenávat události nastalé při běhu softwaru, hlásit závady administrátorům, hledat optimální nastavení a monitorovat provoz.

Nástroje pro testy použitelnosti

Testy použitelnosti jsou speciální kategorií, která nezkoumá přímo aplikaci, ale interakci uživatele s aplikací. Nástroje pro podporu takových testů umožňují natáčet uživatele i obrazovku, na které je test prováděn. Jiné tvoří mapu interakcí, tedy vyznačují místa, která uživatele na obrazovce nejvíce zaujmou. Existují také nástroje, které za pomoci speciální kamery dovedou snímat, kam se uživatel právě dívá (RUBIN a CHISNELL, 2011).

3.4.6 Nástroje pro řízení testů

Malé projekty se často obejdou bez jakýchkoliv nástrojů, maximálně využijí tabulkový editor. U větších projektů je však těžké si udržet přehled o všech naplánovaných

testech. K tomu pomáhají nástroje pro řízení testů, které pokrývají celý životní cyklus softwaru. I proto představuje tato kategorie širokou škálu nástrojů. Některé nástroje v sobě obsahují i nástroje z jiných kategorií (PINKSTER, VAN DE BURGT a kol., 2004). Nejčastěji podporují:

- psaní testů, přiřazování testů k požadavkům a testerům;
- plánování a řízení běhu testů;
- kontrolu průběhu testů;
- zaznamenávání výsledku testů;
- reportování průběhu testů pomocí metrik.

Nástroje pro řízení testů také získávají a poskytují informace komunikací s ostatními nástroji skrz rozhraní. Takovými nástroji mohou být nástroje pro spouštění automatických testů, pro řízení požadavků, pro správu chyb nebo nástroje konfiguračního managementu.

Nástroje pro řízení požadavků

Nástroje pro řízení požadavků nelze úplně pokládat za nástroje testovací, s testováním však silně souvisí (ADNAN a NAQVI, 2015). Testovací případy jsou tvořeny právě z požadavků a také se sleduje pokrytí požadavků testy. Některé nástroje pro řízení testů proto do sebe integrují i nástroje pro řízení požadavků. Hlavní úlohou takových nástrojů je:

- uchovávat požadavky a jejich vlastnosti;
- upozorňovat na nedefinované požadavky;
- prioritizovat požadavky za účelem upřednostnění při testech;
- propojovat požadavky s testovacími případy;
- poskytovat rozhraní pro sdílení informací s nástroji pro řízení testů.

Nástroje konfiguračního managementu

U větších projektů vzniká velké množství verzí různých modulů. Testovací tým může ztratit přehled, která verze má být právě testovaná a která je nasazena v testovacím prostředí. Nástroje konfiguračního managementu pomáhají se takovým problémům vyhnout (HAMZAH, 2013). Při testování pomáhají:

- uchovávat informace o verzích a sestavách softwaru stejně jako o verzích testů;
- sledovat propojenost verzí, operačních systémů, prohlížečů a knihoven;
- řídit plánování nových verzí;

- kontrolovat přístupy do systémů.

Nástroje pro správu zaznamenaných chyb

Chyby nalezené při testování musejí být nahlášený vývojářům, opraveny a přetestovány (DESAI a SRISTAVA, 2016). S celým tímto procesem pomáhají nástroje pro správu chyb, někdy také nazývané nástroje pro sledování chyb. Jejich hlavním úkolem je:

- ukládat chyby a jejich vlastnosti včetně příloh (např. printscreen);
- prioritizovat chyby a nastavovat jim status;
- přiřazovat nálezy vývojářům k opravě nebo testerům k přetestování a uzavření;
- reportovat metriky o chybách.

3.5 Reporty

Jak již bylo popsáno v kapitole 3.3.9, pro úspěšné dokončení projektu je potřeba průběžně měřit jeho kvalitu, nákladnost a efektivnost. S takovým úkolem mohou manažerům pomoci jednotlivé metriky, případně více metrik spojených do ucelených reportů.

3.5.1 Typy zobrazení metrik

Základní a odvozené metriky není obtížné v nástrojích pro řízení testů zobrazit, neboť jsou reprezentovány výpočtem s číselným výsledkem. Někdy je však pro získání informace potřeba porovnat více metrik mezi sebou, pro takové účely slouží tabulkové nebo grafické zobrazení (KAN, 2002).

Tabulkové zobrazení

Pomocí tabulek a seznamů se zobrazují skupiny metrik, jejichž vztahy by se těžko zobrazovaly graficky. Pokud by test manažera například zajímalo, jaké požadavky jsou pokryté testy slabě nebo vůbec, číselný výsledek metriky se vzorcem (9) by mu nikterak nepomohl. Oproti tomu Tabulka 2 mu poskytne přehlednou informaci, že „Požadavek 3“ ještě není pokryt žádným testem.

Tabulka 2: Nepokryté požadavky

Název požadavku	Počet přiřazených testovacích případů
Požadavek 1	2
Požadavek 2	5
Požadavek 3	0

Zdroj: vlastní zpracování

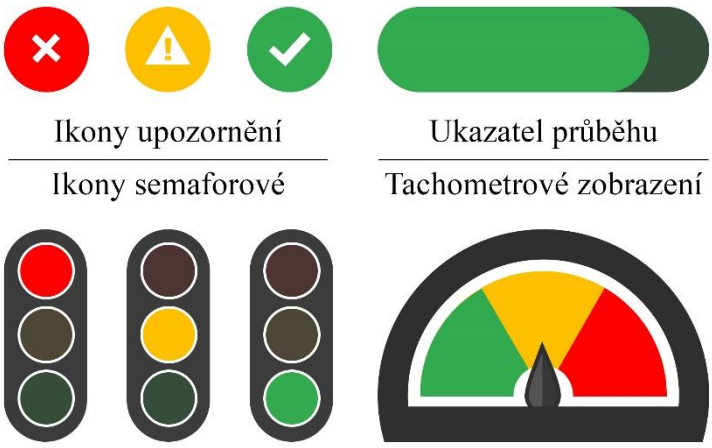
Grafické zobrazení

Ještě přehlednější než tabulky je grafické zobrazení, případně může být přehlednost tabulek vylepšena grafickými prvky zobrazující stav určité metriky. Některé základní metriky lze graficky zobrazit jedinou ikonou, např. úspěšnost testu. Pro jiné je vhodnější komplexnější zobrazení sloučených metrik pomocí grafů, např. zastoupení úspěšných, blokovaných a neúspěšných testů v celkovém počtu testů (KERZNER, 2013).

Zobrazení samostatných metrik

Nejjednodušší vizualizací jsou pravděpodobně **ikony upozornění**. Pomocí nich se zobrazuje více či méně žádoucí stav zkoumané metriky ve formě zelených, žlutých a červených ikon. Nejčastěji se používají jako výstižné oznámení stavu spolu s upřesňujícími informacemi, případně při potřebě zobrazit stavy většího množství zhuštěných informací. Bývá doporučováno kromě barvy rozlišit ikony také tvarem, neboť 10% mužů a 1% žen je barvoslepých. Rozšířením ikon upozornění jsou tzv. ikony semaforové.

Obrázek 1: Zobrazení samostatných metrik



Zdroj: vlastní zpracování

Ukazatel průběhu je vizualizací, která dovede poskytnout více informací. V základu může zobrazovat pouze stav průběhu sledované metriky, v kombinaci s barvou však může nabídnout i informaci o překročení určité hranice nebo přiblížení k určitému limitu. Bývají využívány především k zobrazení poměrné hodnoty vůči nějakému stanovenému cíli. Nepoužitelné jsou pro metriky, které nabývají kladných i záporných hodnot. V takovém případě je lepší použít tachometrové zobrazení.

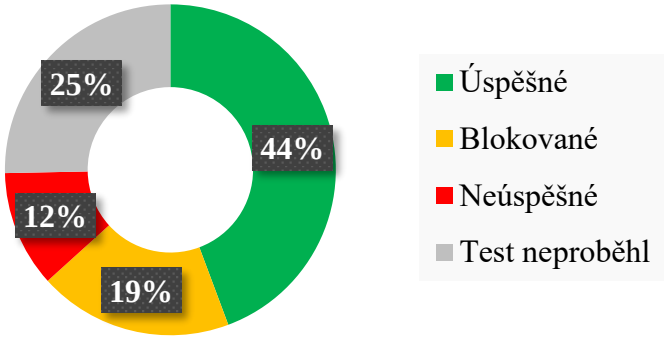
Zobrazení sloučených metrik

Koláčový graf, jinak také kruhový či výsečový, je ideální pro zobrazení menšího počtu metrik, které spolu souvisejí. U více než 6 hodnot bývá špatně čitelné proporční zastoupení jednotlivých hodnot. Existuje mnoho dalších variant tohoto typu grafu:

- prstencový – umožňuje do uvolněného místa uprostřed grafu vepsat doplňující informace o sloučených metrikách;
- roztříštěný koláčový – oddálením části grafu od zbytku upozorňuje na významnější skupinu hodnot;
- polární – pomocí více úrovní prstenců dokáže zobrazit více dat.

Graf 1 zobrazuje pomocí prstencového grafu čtyři metriky týkající se stavu testu, tedy procentní zastoupení úspěšných, blokovaných, neúspěšných nebo ještě neprovedených testů na celkovém počtu navržených testů.

Graf 1: Prstencový graf

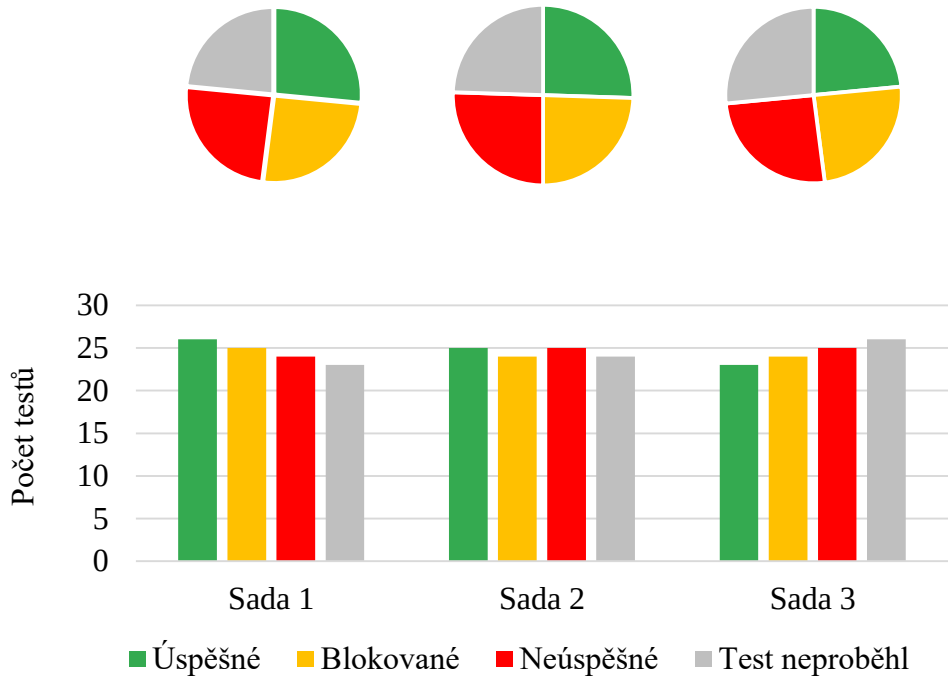


Zdroj: vlastní zpracování

Sloupcový graf je ideální pro větší množství souvisejících metrik, v případě skupinového sloupcového grafu dokáže zobrazit i více metrik roztříděných dle kategorií. Navíc má oproti koláčovému grafu daleko lepší čitelnost v případě menších rozdílů mezi

metrikami. Variantou sloupcového grafu je skládaný sloupcový graf, který stejně jako vícebarevný ukazatel průběhu dokáže zobrazit poměr jednotlivých metrik v celku. Grafu, který má opačné osy než graf sloupcový, se říká pruhový. Graf 2, zobrazující výsledek testů ve třech různých sadách, porovnává čitelnost koláčového a sloupcového grafu, kdy jsou rozdíly mezi sloupci oproti výsečím jasně patrné.

Graf 2: Porovnání koláčového a sloupcového grafu



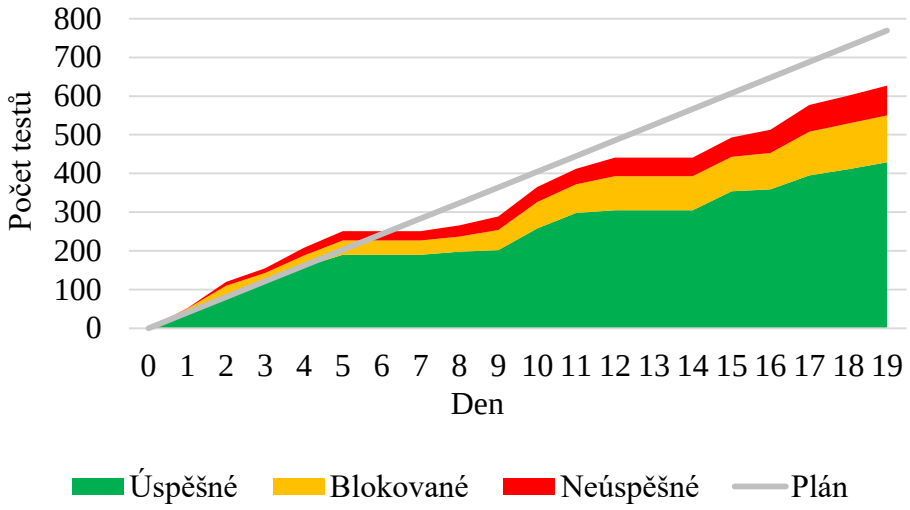
Zdroj: vlastní zpracování

Spojnicový graf, někdy také liniový či čárový, nachází uplatnění při pozorování metrik v průběhu času. Je tvořen spojením naměřených hodnot po určitých časových intervalech čárou, případně křivkou. Často slouží ke zkoumání trendů. Obdobně jako u sloupcového grafu je i u spojnicového možné využít skupinovou variantu pro zobrazení více porovnávaných metrik.

Plošný graf se využívá podobně jako skládaný sloupcový graf, tedy k zobrazení poměru metriky na sledovaném celku, je však podobně jako spojnicový graf určen ke zkoumání časových řad.

Někdy pro sdělení informací nestačí jen jeden typ grafu. Takový problém řeší **kombinované grafy**, které umožňují zobrazit metriky pomocí více typů. Graf 3: zobrazuje vývoj testů v čase pomocí plošného grafu a porovnává jej s plánem vykresleným liniovým grafem. Dá se z něj snadno vyčíst, že od druhého týdne jsou testy za plánem pozadu.

Graf 3: Kombinovaný graf



Zdroj: vlastní zpracování

3.5.2 Základní metriky

Základní metriky jsou takové, které umí test manažerům odpovědět na otázky typu: „Jak velký je projekt, kolik času zabere jej otestovat, kolik budou testy stát a kolik bude stát oprava nalezených chyb?“. Jedná se tedy o čas, náklady, testy a chyby (HUTCHENSON, 2003). V absolutním vyjádření mohou představovat:

- čas – plánovaná délka testů v hodinách/dnech, skutečná délka testů;
- náklady – hodinová mzda členů test teamu;
- testy – počet úspěšných/blokovaných/neúspěšných testovacích případů, celkový počet testovacích případů;
- chyby – počet nalezených/opravených chyb, počet lehkých/závažných chyb.

3.5.3 Odvozené metriky

Základní metriky jsou velmi užitečné, často ale samy o sobě nestačí. Obvykle se proto dále využívají pro výpočet složitějších odvozených metrik. Ty se pak podle zaměření dají seskupovat do kategorií (LEWIS, 2017).

Sledování a efektivita testů

Do této kategorie metrik lze zařadit ty, které pomáhají s určením efektivity testů při odhalování chyb. Pokud test manažer potřebuje znát procentní úspěšnost testů, použije vzorec:

$$\text{Procentní úspěšnost testů} = \left(\frac{\text{Počet úspěšných testů}}{\text{Počet provedených testů}}\right) \times 100 \quad (1)$$

Obdobný vzorec může využít i pro procentní zastoupení blokovaných nebo neúspěšných testů. Další vzorce spadající do této kategorie jsou:

$$\text{Procento kritických chyb} = \frac{\text{Počet kritických chyb}}{\text{Celkový počet nahlášených chyb}} \times 100 \quad (2)$$

$$\text{Průměrný čas opravy chyby} = \frac{\text{Celkový čas opravování chyb}}{\text{Celkový počet nahlášených chyb}} \quad (3)$$

Testovací úsilí

Metriky této kategorie jsou většinou ve formě průměrů a slouží k zodpovězení otázek ohledně vynaloženého úsilí na testování. Současně mohou posloužit jako vodítka pro plánování budoucích testů.

$$\text{Rychlost tvorby testů} = \frac{\text{Počet navžených testovacích případů}}{\text{Celkový čas tvorby}} \quad (4)$$

$$\text{Rychlost nalézání chyb} = \frac{\text{Počet nahlášených chyb}}{\text{Celkový čas testování}} \quad (5)$$

$$\text{Průměrná prodleva přetestování oprav} = \frac{\text{Celkový čas mezi opravou a přetestováním všech chyb}}{\text{Celkový počet chyb}} \quad (6)$$

Účinnost testů

Takové metriky pomáhají zjišťovat, jak dobře napsané jsou testovací případy, tedy zda dostatečně pokrývají testovaný produkt tak, aby dokázaly odhalit co největší množství chyb. Po dokončení testů a uvolnění produktu mezi uživatele se zpravidla ještě nějaké chyby objeví. Pro měření míry zachycení chyb se používá metrika:

$$\text{Procentuální zachycení chyb} = \frac{\text{Počet chyb nalezených při testování}}{\text{Celkový počet chyb (včetně chyb nalezených uživateli)}} \times 100 \quad (7)$$

Pokud je hodnota této metriky pro projekt například 80%, znamená to, že 20% chyb testy nezachytily. To pak musí vést k prošetření míst úniků a nápravným krokům. Přestože je málo pravděpodobné dosažení hodnoty 100%, testovací tým se vždy musí snažit o co nejvyšší hodnotu. Pokud se metrika používá mezi jednotlivými fázemi vývoje, může test manažer sledovat, zda se účinnost testů v čase zvyšuje.

Pokrytí testů

Skupina těchto metrik dokáže odpovědět na otázky týkající se pokrytí produktu navrženými testy, případně v jaké fázi se již testy nacházejí. Pro zjištění, kolik procent testů již proběhlo, je možné použít metriku:

$$\text{Procento uskutečněných testů} = \frac{\text{Počet již proběhlých testů}}{\text{Celkový počet navržených testů}} \times 100 \quad (8)$$

Při fázi návrhu testů je potřeba sledovat aktuální pokrytí požadavků testy. K tomu slouží metrika:

$$\text{Pokrytí požadavků} = \frac{\text{Počet pokrytých požadavků}}{\text{Celkový počet požadavků}} \times 100 \quad (9)$$

Náklady na testy

Čas členů týmu, infrastruktura a nástroje využité při testování jsou základními složkami testovacích nákladů. Projekty obvykle nemají neomezené rozpočty, je proto potřeba tyto náklady hlídat. Následující metrika pomůže s finančním plánováním.

$$\begin{matrix}\text{Předpokládaná}\\ \text{cena testů}\end{matrix} = \begin{matrix}\text{Předpokládaný}\\ \text{čas testů}\end{matrix} \times \begin{matrix}\text{Jednotková časová}\\ \text{cena všech zdrojů}\end{matrix} \tag{10}$$

Jako přínos testů je potřeba brát i nevyužité náklady na netestování, tedy náklady, které by vznikly, pokud by testy neproběhly. Do takových nákladů se počítá větší zátěž na zákaznických linkách, ztrátu zákazníků nebo zhoršení pověsti vývojové společnosti.

3.5.4 Metriky pro testovací tým

Tyto metriky pomáhají s kontrolou rozdělení práce mezi členy týmu a výkonností jednotlivých členů (LIMAYE, 2009).

Produktivita tvorby testovacích scénářů může být v menších týmech kontrolována generováním koláčového grafu, kdy každá výseč představuje **počet vytvořených případů** jedním uživatelem. Ve větších týmech je vhodnější seřazený sloupcový graf, ze kterého je ihned patrné, kdo je nejvíce či nejméně produktivní.

Při plánování průběhu testů je snadné mít přehled o rozdělení testovacích případů mezi členy týmu. Pomůže k tomu seřazený sloupcový graf zobrazující **počet přiřazených případů** jednotlivým testerům. V případě značného přetížení některých členů je pak nutné přerozdělit případy na méně vytížené členy týmu.

Při reálném průběhu testů lze odhalit nízkou výkonnost testera v porovnání se zbytkem týmu. K tomu lze opět využít sloupcový graf, který vykreslí pro každého testera metriku **počtu uskutečněných testů**. Stejně tak je možné kontrolovat členy s nejvyšším počtem chyb čekajících na přetestování nebo nejvyšším počtem neodhalených chyb v předchozím cyklu.

3.5.5 Metriky pro běh testů

Skupina metrik pro běh testů se zabývá především aktuálním stavem testů, případně jejich dosavadním vývojem (CRAIG a JASKIEL, 2002). Testovací případy mohou ve většině nástrojů nabývat jen několik stavů. Klasicky se jedná o úspěšný, blokovaný,

neúspěšný nebo neprovedený test, jen výjimečně jsou zaznamenávány i stavy čekající nebo nekompletní. Základní přehledy stavů testu se tak dají pohodlně zobrazit koláčovým grafem. V případě, kdy je zkoumán podrobnější stav, tedy například zastoupení jednotlivých stavů testů v závislosti na testovaných softwarových modulech, je potřeba metriky zobrazit pomocí skupinového sloupcového grafu, kdy sloupce reprezentují metriky a jsou seskupeny dle modulů. Za nejpraktičtější pro zobrazení dosavadního vývoje stavu testů lze považovat skládaný plošný graf, kde každá plocha představuje **počet testů s daným výsledkem** za minulé testovací dny.

3.5.6 Metriky pro chyby

Tato kategorie metrik pomáhá s kontrolou chybovosti softwaru a s lepším porozuměním původu nebo typu chyb (AINAPURE, 2009).

Koláčový graf lze využít pro zobrazení zastoupení chyb dle závažnosti, priority, stavu řešení nebo platformy původu, neboť se do grafu slučuje jen malé množství metrik. Například pro zastoupení chyb dle závažnosti budou výseče grafu představovat většinou pouze 4 metriky – poměr kritických, závažných, středních nebo nezávažných chyb.

Pro zobrazení metrik seskupených dle kritérií s více zástupci je praktičtější skupinový sloupcový graf. Ideální je pro vizualizaci typů chyb seskupených dle:

- testerů, kteří chyby nahlásili;
- modulu, ve kterém byla chyba nalezena;
- typu testu, při kterém byla nalezena;
- kombinace více seskupení.

Pro porovnání metrik v čase, například **počtu kritických chyb za určitý cyklus**, sprint nebo iteraci, je nejpraktičtější liniový graf.

4 Vlastní řešení

4.1 TestLink

TestLink je nástroj pro řízení testů, pokrývající zajišťování kvality po celý životní cyklus softwaru. Umožňuje vytvoření a úschovu požadavků, testovacích případů a uživatelů. Propojuje testovací případy s požadavky. Přiřazuje testovací případy uživatelům, zaznamenává výsledky testů a nabízí řadu reportů.

Nástroj je distribuován s licencí open-source, jeho zdrojový kód je tedy otevřený a poskytnutý volně k šíření i libovolným úpravám. Jedná se o webovou aplikaci spustitelnou téměř v jakémkoliv prohlížeči podporujícím JavaScript, HTML a CSS. Kód je napsán v jazyce PHP, lze jej tedy rozběhnout na libovolném serveru s podporou tohoto jazyka. Informace ukládá do databáze, administrátor může vybírat mezi podporovaným MySQL, MS-SQL 2000 nebo Postgres. Na adrese TestLink.org se nachází rozcestník s odkazy na kód na úložišti Github, živé demo nástroje, fórum a hlášení chyb. V době psaní této práce je k dispozici ve verzi 1.9.16 - Moka Pot, vydané v lednu roku 2017.

4.1.1 Instalace

Po stažení kódů a uložení na webserver následuje připravení prostředí na instalaci. Nejprve je nutné vytvořit novou databázi a připravit si k ní přístupy. Dále musí být pro TestLink nastaveno právo pro zápis do složek gui/templates_c, logs a upload_area. Pro webservery na operačních systémech Linux/UNIX lze použít příkaz „chmod“. Poté si lze vybrat mezi automatickou a manuální instalací.

Automatická instalace

TestLink nabízí administrátorům instalační skript, který pomůže s nastavením všech konfigurací včetně struktury a přístupu do databáze. Postup je velmi intuitivní a nezabere více než 30 minut. Skript se spouští z prohlížeče a sestává z pár jednoduchých kroků. Nejprve je nutné vybrat, zda se jedná o novou instalaci či upgrade z předchozí verze a zaškrtnout souhlas s licenčními podmínkami. V dalším kroku skript zkontroluje systémové požadavky a vyžádá si umístění, typ databáze a přihlašovací jméno s heslem do databáze.

Následně skript vytvoří strukturu databáze a prvního uživatele s přístupem administrátora. Po dokončení instalace se doporučuje smazat instalační složku.

Manuální instalace

Manuální instalace je doporučována pouze v případě, že automatickou instalaci nelze z nějakého důvodu spustit. Podrobný popis naleznete v instalačním manuálu (TESTLINK COMMUNITY, 2010).

Konfigurace

TestLink umožňuje uživatelům měnit některá nastavení. Změny se ukládají v souboru costum_config.inc.php, kde si uživatel může nastavit například jiné šablony stránek, barvy, textový editor, zda se má zobrazovat celá historie testů nebo jen poslední výsledek, definovat nové statusy testovacích případů, určit maximální velikost nahrávaných příloh a mnoho dalšího. TestLink lze pomocí konfiguračního souboru také propojit s nástroji pro správu chyb, mezi které patří Bugzilla, Mantis, JIRA, TrackPlus, Eventum, Trac, Fogbugz, Gforce a Redmine.

4.1.2 Uživatelské role

TestLink rozlišuje 5 základních uživatelských rolí. První uživatel vytvořený při instalaci obdrží roli administrátora. Tato role není editovatelná a má nastavená všechna oprávnění. Nástroj navíc administrátorům nabízí možnost vytvořit si role nové a přiřadit jim libovolné oprávnění.

Druhé nejvyšší oprávnění obdrží lídři. Na rozdíl od administrátorů nemají přístup ke správě uživatelů a rolí, nemohou spravovat pluginy, seznamy platforem a ani vytvářet nové testovací projekty. Mnohdy je tak žádoucí, aby byl test manažer současně i administrátor a roli lídrů obdrželi jen jeho podřízení, kteří mají na starost skupiny testerů.

Role test designéra již může pouze spravovat požadavky, testovací případy a klíčová slova. Role tester je určena pouze k zobrazení testovacích případů a provádění testů. Role host si může jen zobrazit testovací případy a výsledky testů.

Uživatelská oprávnění se přiřazují na úrovni testovacích projektů a plánů. Administrátor tak může přidělit testerovi práva jen k některým testovacím plánům z projektu.

4.1.3 Používání

Uživatelské rozhraní TestLinku vypadá velmi zastarale a logické rozvržení ovládacích prvků je neintuitivní. Ovládání tak pro nové uživatele není snadné.

Terminologie nástroje

Nástroj nabízí možnost lokalizace, lze vybrat i český jazyk. Ne všechny textové řetězce jsou však přeloženy. Kurzívou je uveden český překlad používaný nástrojem, v závorkách názvy v angličtině.

Hlavní jednotkou, pod kterou spadá vše ostatní, je *Testovací projekt* (Test Project). Ten může mít přiřazen *Požadavky* (Requirements) a *Testovací případy* (Test Cases). Testovací případy se spravují v modulu *Specifikace* (Test Specification). Mohou být seskupeny do *Testovacích sad* (Test Suites). Při plánování průběhu testů se testovací případy a sady přiřazují do *Testovacího plánu* (Test Plan), který je dále rozplánován na jednotlivé *Sestavení* (Test Builds). *Výsledky testů* (Test Results) jsou zaznamenávány v modulu s názvem *Provést* (Test Execution). Výsledné přehledy a grafy lze zobrazit v modulu *Výsledky* (Test Reports). Uživatelé a jejich oprávnění se spravují v části (Users/Roles), český překlad chybí.

Domovská obrazovka

Při prvním přihlášení dochází k automatickému přesměrování uživatele na vytvoření Testovacího projektu. Po dalším přihlášení, je-li již projekt vytvořen, se uživatel dostane na domovskou obrazovku. V horní části se nachází hlavní menu s odkazy na moduly Požadavky, Specifikace, Provést, Výsledky, Uživatelé, Události a Pluginy. Menu mimo jiné obsahuje též vyhledávač a rozbalovací pole pro výběr aktuálního projektu. Toto hlavní menu je zobrazeno i na všech ostatních stránkách. Domovská obrazovka, rozdělená do dvou sloupců, slouží jako rozcestník pro detailnější řízení testů. V levém sloupci najde test manažer správu testovacích projektů, požadavků a specifikaci testů. V pravém je umístěna správa testovacích případů, blok určený provádění testů a obsahu testovacího plánu. Tento rozcestník je uživatelům zobrazován na základě jejich oprávnění, obyčejným testerům se tak zobrazí pouze specifikace testů a jejich provádění.

Testovací projekt

Z bloku Správa testovacích projektů na domovské stránce se lze dostat k zakládání nových testovacích projektů. Každý projekt musí mít unikátní název a předponu, která je přidávána k číslu testovacích případů spadajících pod tento projekt. Ve stejném bloku jsou také odkazy na správu klíčových slov, platforem a inventáře. To svědčí o neintuitivnosti uživatelského rozhraní, neboť klíčová slova se používají až při tvorbě testovacích případů a platformy při plánování testů. Inventář představuje jednoduchý nástroj konfiguračního managementu, který byl popsán v kapitole 3.4.6. Umožňuje uchovávat základní informace o testovacím hardwaru.

Testovací případy a sady

Modul Specifikace funguje jako nástroj pro tvorbu testů představený v kapitole 3.4.3. Umožňuje vytvářet, editovat, kopírovat, mazat testovací případy a sady. Každý testovací případ má název, popis, případné přílohy a očekávaný výsledek. Lze mu také nastavit stav (zda se jedná o návrh, zda je předán ke kontrole či zda je již ve finální podobě), důležitost, očekávaný čas testu a typ provedení (automatické či manuální). Dále lze případy přiřazovat k požadavkům, případně je odkazovat na jiné související případy. Jako speciální atribut pro filtrování a seskupování případů slouží klíčová slova.

Pro navigaci mezi testovacími případy a sadami lze využít postranní levé menu se stromovou hierarchickou strukturou. Zde se dají sady a případy snadno třídit a přesouvat pomocí metody drag-and-drop. Toto menu se vyskytuje ve všech modulech pracujících s testovacími případy.

Testovací plány a sestavení

Pro plánování testů neexistuje samostatný modul, na veškerou správu týkající se testovacích plánů a sestav se lze prokliknout z pravého menu domovské obrazovky. Kromě názvu je možné testovacímu plánu přiřadit také popis, ve kterém je dobré uvést, co se v rámci tohoto plánu bude testovat, v jakém prostředí či na jaké platformě, možná rizika a testovací kritéria pro úspěšné ukončení testů. Testovací plán může být vytvořen i jako kopie z předešlých plánů, což ušetří čas s opětovným přiřazováním případů.

Přestože testovací případy v plánu většinou zůstávají stejné, testovaný software se mění, ať už díky opravám nebo novým funkcím. Je tedy dobrým zvykem pro každou novou verzi softwaru nastavit i nové testovací sestavení. Každý testovací plán může mít více

sestavení. K jejich založení se uživatel dostane opět z domovské obrazovky přes odkaz Správa sestavení. Zde může nové sestavení založit, popsat a nastavit mu koncové datum, takzvané datum vydání (Release date).

Plánování a přiřazování testů

K přiřazení testovacích případů do testovacího plánu se uživatel dostane z panelu Obsah testovacího plánu na domovské obrazovce. Je zde umístěn také odkaz, pod kterým se skrývá aplikace na přiřazení případů jednotlivým testerům. Nástroj umožňuje i hromadné přiřazení případů k testerovi i do plánu najednou, což značně ušetří čas. Pokud má zvolený tester v systému nastavenou e-mailovou adresu, je o přiřazení okamžitě informován a může začít s prací.

Provádění testů

V modulu Provést může tester načíst k němu přiřazené testovací případy. Ty se opět zobrazují v levé hierarchické struktuře testovacích sad. Při načtení testovacího případu se testerovi zobrazí všechny atributy případu včetně jeho zařazení a detailních kroků. Také si může zobrazit kompletní historii testování v předchozích sestaveních. Při zadávání výsledku testu je možné přidat i rozšiřující popis nebo přílohu a skutečnou délku trvání testu v minutách. Výsledky lze zadat i pro jednotlivé kroky případu. Pokud je v rámci testovací sady více případů, při uložení výsledku lze automaticky přejít na další.

4.1.4 Reporty

Pro reporty je určen samostatný modul Výsledky. V levém menu se nachází pole pro výběr testovacího plánu, pro který se mají reporty zobrazovat. Hned pod ním je seznam dostupných reportů. Seznam se mění v závislosti na tom, v jakém formátu se mají reporty vygenerovat. V nabídce uživatel nalezne formát:

- HTML, standardní zobrazení v prohlížeči, pro které jsou dostupné všechny reporty;
- MS Excel, nabízí 6 tabulkových reportů;
- MS Word, nabízí 3 reporty psaného textu kombinovaného s jednoduchými tabulkami.

Report testovacího plánu

Tento report je dostupný i ve formátu MS Word. Generuje celkový přehled o vybraném testovacím plánu. Součástí jsou podrobně vypsané atributy pro všechny obsažené testovací případy a sady. Před samotným generováním může uživatel vybrat, jaké části má dokument obsahovat a do jakých detailů má vypisovat vlastnosti případů.

Report průběhu testů

Obsahově je tento report shodný s předchozím, pouze přidává možnost výpisu výsledků testů a to i s rozdělením na sestavení, v nichž byly případy testovány. V případě výběru reportu průběhu testů v závislosti na sestavení pak lze vypsat jen ty výsledky, které se vybraného sestavení týkají.

Celkový přehled testovacího plánu

Tento report je dostupný pouze v HTML a MS Excel formátu. Jedná se o skupinu tabulek zobrazující metriky s určitým vztahem. První tabulka s názvem Celkový stav sestavení zobrazuje na základě sestavení tyto seskupené metriky:

- počet přiřazených testů;
- absolutní počet a procentuální zastoupení úspěšných / blokovaných / neúspěšných / ještě neuskutečněných testů;
- procentuální zastoupení uskutečněných testů;

Podobné metriky jsou pak v dalších tabulkách seskupeny ještě podle testovacích sad, klíčových slov a priorit.

Výsledky dle testerů a sestavení

Jedná se o tabulkový report dostupný pouze v prohlížeči. Využívá knihovnu pro snadné třídění a řazení dle sloupců a skrývání nepotřebných skupin. Na základě všech sestavení a všech testerů přiřazených do daného sestavení zobrazuje stejné seskupené metriky, jako report Celkový přehled testovacího plánu. Navíc však zobrazuje i celkový čas, jaký tester strávil prováděním testů ve vybraném sestavení. Při kliku na jméno testera se k výše uvedenému zobrazí i přehled všech jemu přiřazených testů se zaznamenaným výsledkem a metrikou počtu dní od přiřazení testu testerovi.

Přehled přiřazení testovacích případů

Pro přehled všech přiřazení testovacích případů do některého sestavení seřazených podle testerů slouží tento report. Některé případy se tak vyskytují i vícekrát pro každé zařazení. U případů je vypsán poslední výsledek testů, lze si zobrazit i historii výsledků. Z tohoto přehledu se dá také zaznamenat nový výsledek nebo se prokliknout ke specifikaci testu.

Přehled testování

Další tabulkové zobrazení nabízí v řádcích testovací případy seskupené dle testovacích sad a ve sloupcích všechny sestavení, do kterých byl případ zařazen, včetně výsledku testu v daném sestavení podle verze případu. U každé skupiny je uvedena metrika celkového počtu testovacích případů v dané sadě.

Další položka v menu odkazuje na podobný report, pouze ve formátu MS Excel.

Neúspěšné/blokované/neprovedené testovací případy

V menu se nachází pro všechny tyto tabulkové reporty samostatné položky. Kromě výsledku testovacího případu však zobrazují stejné informace. Testy jsou zde seskupovány podle sestavení. Ve sloupcích zobrazují verzi případu, uživatelské jméno testera, který test provedl, datum a čas provedení a poznámku k výsledku. U neprovedených testů je kromě výsledku testu a poznámky k výsledku uveden popis případu. Všechny tři reporty jsou dostupné i ve formátu MS Excel.

Testovací případy bez přiřazeného testera

Při plánování testů je užitečný report, zobrazující testovací případy, které ještě nebyly nikomu přiřazeny. Seskupuje případy dle testovacích sad a vypisuje k nim prioritu a popis případu. V tomto reportu se nezobrazují testovací případy, které nejsou přiřazeny v aktuálním sestavení, pokud byly některému testerovi přiřazeny v alespoň jednom minulém sestavení.

Grafy

Pod položkou Grafy se skrývají jediné tři grafické reprezentace dat. Prvním grafem je graf koláčový s názvem Celkové metriky. Ve výsečích zobrazuje procentuální zastoupení úspěšných, blokovaných, neúspěšných a neprovedených testů. Současně u popisku zobrazuje i celkový počet testů s daným stavem.

Zbylé dva grafy jsou grafy sloupcové skládané. První z nich zobrazuje sloupec pro každé klíčové slovo a každá část sloupce pak představuje počet úspěšných, blokovaných, neúspěšných a neprovedených testů s daným klíčovým slovem. V případě, že nastane situace přiřazení více klíčových slov k jednomu případu, bude zaznamenaný výsledek zobrazen ve více sloupcích. Druhý sloupcový graf představuje počet testů s daným výsledkem, seskupených dle testovacích sad nejvyšší úrovně.

Přehled podle požadavků

Tento tabulkový report zobrazuje úspěšnost systémových požadavků v závislosti na výsledcích testů, které tyto požadavky pokrývají. Přehled je seskupen podle sad požadavků a ve sloupcích vypisuje metriky pro procentuální zastoupení výsledků testů pokrývajících požadavek, včetně testů ještě neprovedených.

Chyby podle testovacích případů

V případě propojení TestLinku s některým z nástrojů pro správu chyb je k dispozici i report, který zobrazuje názvy chyb a jejich stav ve vztahu k testovacím případům. Případy jsou seskupeny dle testovacích sad. Nedílná součást reportu se zabývá výpisem metriky s celkovým počtem chyb, vztahujících se k vybrané testovací sadě.

Testovací případy s informacemi z vlastních polí

TestLink umožňuje uživatelům zakládat vlastní pole, která při tvorbě testů využijí pro detailnější popis testovacího případu nebo k doplnění chybějícího atributu. Tyto doplněné informace pak report vypisuje v závislosti na testovacích případech, seskupených do testovacích sad. K nim také doplňuje výsledky testů včetně data a času zaznamenání.

Testovací případy nepřiřazené k žádnému testovacímu plánu

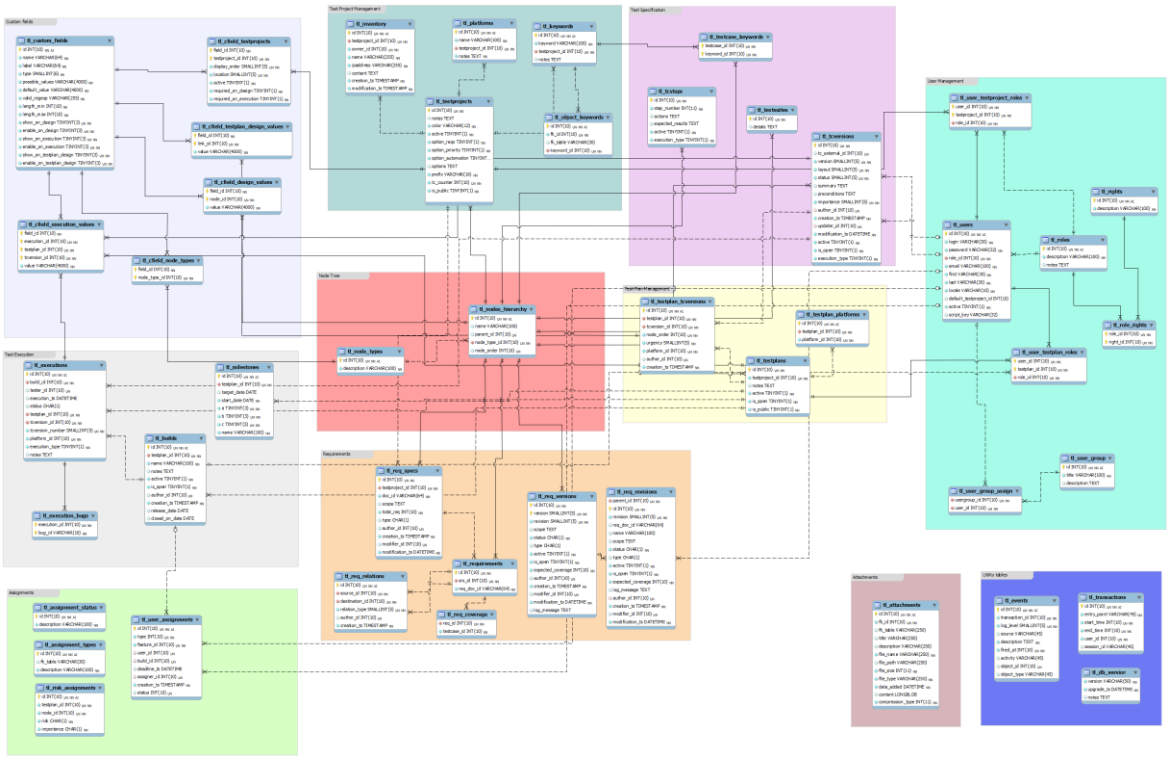
Jak název reportu napovídá, zobrazuje pouze ty případy, které ještě nebyly přiřazeny k žádnému testovacímu plánu. Seskupuje je dle testovacích sad a přidává k nim atribut důležitosti testovacího případu.

4.1.5 Struktura databáze

Databáze TestLinku ve verzi 1.9.16 obsahuje 58 tabulek. Obrázek 2 zobrazuje schéma databáze pro ilustraci její rozlehlosti. Tato práce nevyužije data ze všech tabulek, je

ale nutné seznámit se především s částí uchovávající hierarchii, testovací projekty, plány, sestavení, sady, případy a běhy testů.

Obrázek 2: Schéma DB nástroje TestLink



Zdroj: (TESTLINK, 2014)

Středobodem databáze je tabulka nodes_hierarchy, skládající se z 5 sloupců: identifikačního čísla, jména, typu a pořadí uzlu a identifikačního čísla uzlu nadřazeného. Typ uzlů je uchováván v tabulce node_types. Nejvyšším uzlem v hierarchii je projekt. Kromě základních uzlů se v hierarchii vyskytuje i typ uzlu pro verze testovacích případů a jednotlivé kroky testů. Pro uzly nižší úrovně se objevuje v tabulce více záznamů lišících se hodnotou nadřazeného uzlu pro každý takový uzel. Například testovací případ má zde záznam pro nadřízenou testovací sadu i testovací plán.

Tabulka testprojects uchovává informace o testovacím projektu. Důležité pro vykreslování reportů jsou především sloupce active, prefix a is_public. Tabulka testplans obsahuje podobná data pro testovací plán a navíc identifikační číslo nadřazeného projektu. Tabulka builds ukládá informace o sestaveních a číslo nadřízeného testovacího plánu. Je to

také jediná tabulka, která uchovává jméno objektu v sobě. Všechny ostatní se musí na jméno dotazovat tabulky node_types.

Tabulka executions obsahuje výsledky testů ve sloupci status. Každý běh testů se zde zaznamenává v závislosti na testovacím plánu, sestavení a testerovi. Výsledky se následně přiřazují k verzím testovacích případů uložených v tabulce tcversions. Zde jsou k dispozici všechny atributy testovacích případů, opět vyjma názvu, ten je uchován v hierarchické tabulce.

4.2 Výběr reportů k implementaci

Pro správný výběr reportů k implementaci je nutné zjistit, jaké reporty v TestLinku uživatelům chybí, jaké se nejčastěji využívají v testovacích nástrojích s lepším reportingovým modulem, případně jaké si uživatelé nejčastěji vytvářejí sami kvůli absenci v nástroji, který používají. S takovým výběrem pomohlo autorovi dotazníkové šetření, komunikace na fóru nástroje a interview s test manažerem.

4.2.1 Dotazníkové šetření

TestLink je celosvětově velmi populární nástroj. Od svého vzniku byl jen z úložiště Sourceforge stáhnut již téměř šestsettisíckrát (SOURCEFORGE, 2017). Více než 70% stažení pochází mimo Evropu. Nejčastěji ze Spojených států amerických (16%), Indie (14%), Číny (12%) a Brazílie (7%). Protože chtěl autor této práce zajistit co nejlepší výběr reportů, bylo potřeba oslovit uživatele ze všech koutů světa. Zvolil proto elektronickou formu dotazníkového šetření, která je lehce distribuovatelná. Ke zpracování dotazníku byl využit webový nástroj Google Forms. Autor s ním měl zkušenosti již z dřívějška a splňoval všechny požadavky na druh zadávaných otázek, nepovažoval tedy za nutné vybírat nástroj pomocí vícekriteriální analýzy. Z důvodu celosvětové distribuce byl dotazník sestaven v anglickém jazyce.

Většina reportů bývá využívána pouze test manažery, např. report zobrazující přiřazená hlášení dle uživatelů je však často využíván i řadovými testery. Cílová skupina je proto tvořena jakýmkoliv účastníkem testovacího procesu, který přišel do styku s některým nástrojem pro řízení testů. Za účelem kontroly cílové skupiny byly do dotazníku vloženy otázky na délku působení v testovacím procesu a název nástroje, který respondent využívá.

Dotazník byl rozdělen na 5 souvisejících stránek. Respondenti, kteří odpověděli, že nemají zkušenost s nástrojem TestLink, vyplňovali jen 3 z nich.

Strana 1 – Úvod

Na první straně dotazníku byl uveden název výzkumu „Užitečné reporty a grafy v nástrojích pro řízení testů“. Níže se zobrazoval krátký úvodní text, který obsahoval poděkování za zájem o vyplnění, vysvětlení, že sesbíraná data budou použita pro tuto práci a pro výběr reportů k implementaci do nástroje TestLink a dále následovala informace o délce vyplnění 2 až 5 minut v závislosti na poskytnutých detailech.

Strana 2 – Respondent pracuje v odvětví testování softwaru

Druhá strana s podnadpisem „Něco o vás“ nejprve zjistila přibližné místo působiště dle světadílu, ve kterém respondent pracuje. Další dvě otázky sloužily pro kontrolu cílové skupiny a upřesnění, o jak zkušeného respondenta se jedná. Nejprve měl vybrat odpověď na otázku „Jak dlouho již pracujete v odvětví testování softwaru?“. Nedala se však vybrat odpověď, že v takovém odvětví nepracuje. Všichni respondenti tak jsou minimálně řadovými testery. Druhá otázka se zajímala o délku působení na pozici test manažera. Zde již bylo možné vybrat odpověď, že na takové pozici nikdy nepracoval.

Strana 3 – Respondent má zkušenost s nástroji pro řízení testů

Otázka „Již jste někdy používal TestLink (open-source nástroj pro řízení testů)?“ fungovala jako rozcestník, zda respondent bude vyplňovat další stranu týkající se TestLinku. Následující otázka byla opět určena pro kontrolu cílové skupiny, tedy zda respondenti používají nějaký jiný nástroj pro řízení testů. Odpověď umožňovala výběr více nástrojů nebo přidání vlastního nástroje pomocí textového pole. Nástroje byly vybrány podle průzkumu webu qatestingtools.com. Ten sestavil přehled nástrojů, které byli doporučeny na stránkách Software Testing Help, ISTQB, Code Rewind, Guru 99, EasyQA, Quora a G2 Crowd. Jednoduchým ohodnocením dle pořadí doporučení bylo vybráno 11 nástrojů (QATESTINGTOOLS, 2017). Stejný seznam byl použit i pro otázku „Který z nástrojů má nejlepší funkce pro vytváření přehledů?“. Odpovědi na tuto otázku budou využity k inspiraci u konkurence s lepším reportingovým modulem, než jakým disponuje TestLink.

Strana 4 – Jen pro uživatele TestLinku

Na této straně byla respondentům osvěžena paměť seznamem všech reportů, které TestLink nabízí. Otázky zněly: „Který report nebo graf Vám v TestLinku chybí?“ a „Které tabulkové reporty by v TestLinku měly být zobrazeny ve formě grafů?“. Na obě otázky bylo možné odpovědět delším textem. Takový přístup sice není v kvantitativních dotaznících doporučován, v tomto případě šlo ale spíše o kvalitu. Vzhledem k existenci velkého množství reportů a metrik, využívaných při testování softwaru, by nebylo možné je obsáhnout pomocí otázky s výběrem odpovědi.

Strana 5 – Užitečné reporty a grafy

První otázka na této straně „Které reporty nebo grafy si připravuje samy, protože je Váš nástroj neposkytuje?“ měla za úkol zjistit, zda i ostatní nástroj mají při reportování nedostatky, které uživatele trápí. Tato i následující otázka byla opět mířena spíše kvalitativně a respondenti měli možnost se rozepsat. Druhá otázka „Které reporty nebo grafy rádi a často používáte?“ mířila na určení priority implementovaných reportů. Jako pojistku proti možné neochotě při kvalitativním vyplňování textových polí byla k této otázce připojena i kvantitativní varianta s výběrem více možností. Reporty v odpovědích byly seskupeny do kategorií základní reporty, odvozené reporty, reporty pokrytí testů, reporty pro testovací tým, reporty pro sledování průběhu testů a reporty pro chyby.

Samostatná distribuce dotazníku

K distribuci využil autor práce několik kanálů. Díky známosti lokální komunity profesionálních testerů [pro]Test! se podařilo získat odpovědi 14 respondentů.

Na sociální síti Facebook byli požádáni členové skupin Software Testing (19136 členů), Software Testing Studio (18792 členů), Software Testing Group (16886 členů), Testing Jobs (16118 členů), Quality Assurance Forum (3744 členů) a The QA Testing Channel (3443 členů). Na síti Reddit byl dotazník vyvěšen ve skupinách QualityAssurance (4535 členů), Softwaretesting (3738 členů), SoftwareTestingViews (216 členů) a Software_testing (124 členů). Na serveru Quora.com byl dotazník uložen do popisku otázky „Které reporty vám schází ve vašem nástroji pro řízení testů?“. Pokud budeme předpokládat, že členové mohou být současně ve více skupinách, i tak by mohl být dosah dotazníku v řádech desetitisíců testerů. Přesto se z těchto sítí podařilo získat jen 13 respondentů. Bohužel aplikace Google Forms neumožňuje rozlišit, z které sítě respondenti pocházeli,

autor se tak nemohl zaměřit jen na cennější z nich. Jsou ale k dispozici celková čísla prokliků odkazu na dotazník. Ze všech komunikačních kanálů se k formuláři dostalo 312 čtenářů.

Distribuce pomocí specialistů

Ze samostatné distribuce tak autor práce získal pouze 27 respondentů. Rozhodl se proto využít služby specialistů na průzkum trhu. Všechny oslovené společnosti, vyjma společností Survey Monkey a Toluna, dokázaly zajistit distribuci dotazníku s 11 otázkami a s výše popsanou cílovou skupinou. Jako jediné kritérium pro rozhodování tak zbyla cena za skutečného respondenta (tedy za jednou vyplněný dotazník). Tabulka 3 ukazuje, že společnost Global Survey Market byla jednoznačně cenově nejvýhodnější. Autor s ní proto uzavřel smlouvu na distribuci dotazníku a sesbírání 70 kompletních odpovědí, aby tak disponoval celkovým počtem přibližně 100 odpovědí.

Tabulka 3: Přehled cen za průzkum trhu

Společnost	**Cena za 1 vyplnění dotazníku**
Global Survey Market	1$
Google Surveys	10$
QuestionPro	13$
Survey Gizmo	21$
Respondent.io	27$
CheckMarket	29$
AYTM	32$
Toluna	nemá k dispozici cílovou skupinu
Survey Monkey	nemá k dispozici cílovou skupinu

Zdroj: vlastní zpracování

Vyhodnocení dotazníku

Celkem tedy dotazník vyplnilo 97 respondentů. 22 záznamů však autor vyhodnotil jako neplatné nebo neužitečné. Důvody byly následující:

- nesmyslně vyplněná delší doba na pozici test manažera než celková zkušenost s testováním softwaru;
- žádná zkušenost respondentů s nástroji pro řízení testů;
- jako zkušenost uvedena práce s nástrojem pro správu chyb, ne s nástrojem pro řízení testů;

- za nástroj s nejlepším reportingovým modulem vybrán TestLink, přestože v předchozí otázce respondent vyplnil, že tento nástroj nikdy nepoužíval;
- vybraný nástroj s nejlepším reportingovým modulem se neshodoval s vybranými nástroji, které respondent označil jako že s nimi má zkušenost.

K vyhodnocení tedy zůstalo 75 plnohodnotných vyplnění. Nejvíce respondentů pocházelo z Evropy (19) a Afriky (19), dále z Austrálie (10), Asie (9), Severní Ameriky (9) a nejméně z Jižní Ameriky (3). 6 respondentů neuvedlo, na jakém světadílu pracují. Šetření se zúčastnilo 69 test manažerů. 15 z nich mělo zkušenost z této pozice kratší než dva roky, dvou až pětiletou zkušenost mělo 41 z nich a zbylých 13 pracovalo na této pozici šest až deset let. Žádný z test manažerů neměl více než desetiletou praxi. Z 6 respondentů bez manažerské zkušenosti měl první méně než dva roky zkušeností, druhý šest až deset let a zbylí čtyři působili v testování softwaru od dvou do pěti let.

S nástrojem TestLink mělo zkušenost 61 respondentů, ti tedy vyplňovali i čtvrtou stranu dotazníku s kvalitativními otázkami. Jako report, který TestLinku schází, uvedlo několik dotázaných přehled podle požadavků. Tento report však je součástí nejnovější verze TestLinku, jednalo se tedy nejspíš o uživatele starších verzí. Oprávněné stížnosti pak uváděli, že v reportingovém modulu postrádají:

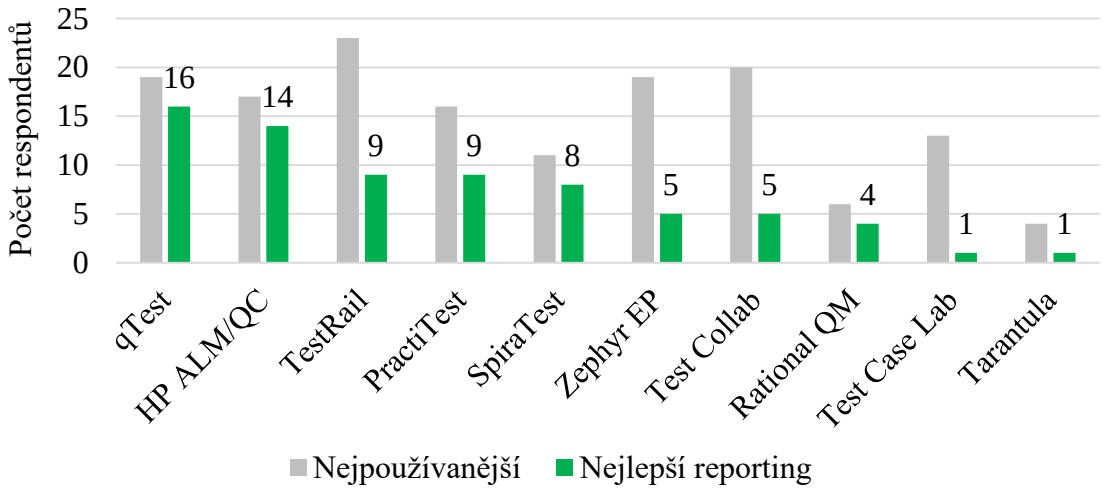
- graf průběhu testů včetně porovnání s plánem;
- graf výsledků testů seskupených dle priority;
- možnost vytvořit si vlastní grafy za pomoci naprogramovaného průvodce.

U otázky, které tabulkové reporty TestLinku by bylo vhodné převést do grafické podoby, byla nejčastější odpověď „chyby dle testovacích případů“. Autor však považuje za vhodné přenechat generování takových grafů nástrojům pro řízení chyb napojených na TestLink. Další návrhy uváděli graf výsledků testů seskupených dle priority, stejně jako u předchozí otázky. Také by uživatelé měli rádi k dispozici graf zobrazující chybné výsledky testů, bohužel neuvedli, podle jakých kritérií by měly být chybné výsledky roztřízeny. Není tak jasné, jak takový graf vykreslit.

Při vyhodnocení používání ostatních nástrojů pro řízení testů se ukázalo, že ty nejpoužívanější nebyly současně považovány za nástroje s nejlepším reportingovým modulem. Pro účely této práce je však důležitější právě spokojenost s reportováním, Graf 4 proto ukazuje nástroje seřazené podle spokojenosti. Nástroje, které sice uživatelé používají, ale nemyslí si, že mají nejlepší reportingový modul, nejsou v grafu uvedeny (mezi takové

nástroje patří i TestLink). Za nejlepší označili respondenti qTest, druhé místo obsadil nástroj HP ALM/QC (nově Micro Focus) a o třetí místo se dělí TestRail a PractiTest. Nejvíce respondentů mělo zkušenost s nástroji TestRail, Test Collab, qTest a Zephyr EP.

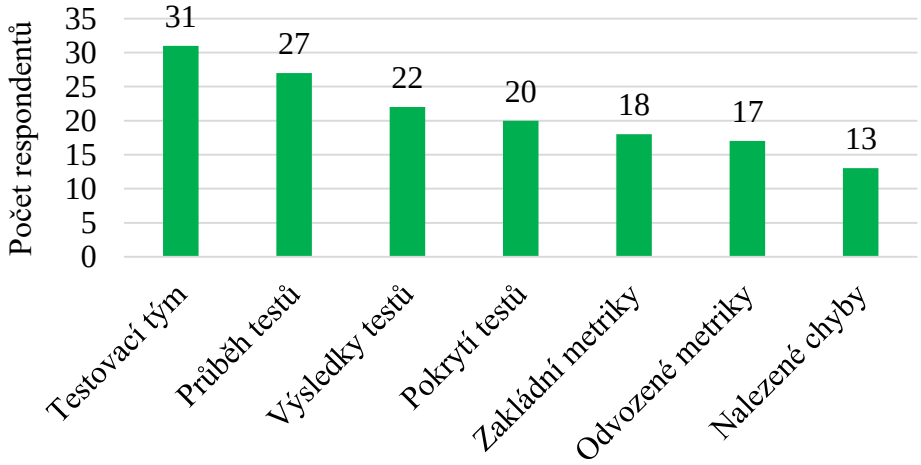
Graf 4: Nejpoužívanější nástroje a nástroje s nejlepším reportingem



Zdroj: dotazník, vlastní zpracování

Na otázku „Které reporty nebo grafy si připravujete sami, protože je Váš nástroj neposkytuje?“ odpověděla většina respondentů, že jejich nástroj nabízí všechny reporty, které potřebují. I tak se ale našlo několik nespokojených, kteří uvedli absenci grafu průběhu testů, grafu pokroku v testech dle uživatelů a reportu přehledu jednotlivých uživatelů.

Graf 5: Nejčastěji používané skupiny reportů



Zdroj: dotazník, vlastní zpracování

Graf 5 zobrazuje skupiny reportů označené respondenty jako nejčastěji používané. Nejvíce hlasů dostaly reporty s metrikami pro testovací tým, průběh testů a výsledky testů.

4.2.2 Komunikace na fóru nástroje

Jako druhý způsob získání informací od uživatelů využil autor této práce oficiální fórum nástroje TestLink. V rubrice konzultující nejnovější verzi nástroje založil nové vlákno, ve kterém položil dvě zásadní otázky, a to:

- který report nebo graf vám v TestLinku chybí;
- využívali jste v jiném nástroji pro řízení testů nějaký užitečný report nebo graf, který byste rádi využívali i v TestLinku.

Uživatelé na fóru však nejsou moc aktivní a tak bylo získáno jen několik návrhů. Jedním z návrhů byl burndown chart, který pomocí liniového grafu vykresluje počet ještě neprovedených testů a porovnává jej s plánem. Jedná se tedy o opak grafu průběhu testů, kdy je vykreslován celkový počet již provedených testů s jakýmkoliv výsledkem. Druhý návrh popisoval tabulkový report, zobrazující všechny výsledky testů za všechny testery v jednom sestavení. Šlo by tak o opak reportu Přehled testování, kde by místo výpisu posledního výsledku za všechna sestavení byly uvedeny všechny výsledky za poslední sestavení. Třetí návrh představil 2 metriky, o které by se daly tabulkové reporty zobrazující výsledky testů rozšířit. Jednalo se o datum posledního neúspěšného výsledku testu a o volatilitu, tedy počet neúspěšných výsledků testů jednoho případu za celou dobu všech sestavení.

4.2.3 Interview s test manažerem

Kvůli možnosti doplnit kvalitativní otázku v případě nejasné odpovědi, využil autor této práce jako třetí možnost interview s test manažerem, který disponuje více než desetiletou praxí v oboru. Současně se jednalo o člověka, který se svým týmem již delší dobu využíval pro řízení testů TestLink. Zkušenost měl také s nástroji společnosti Hewlett-Packard.

Největší přínos pro TestLink spatřoval v doplnění grafu průběhu testů včetně porovnání s plánem. Doporučil využít možnosti nastavit konečné datum sestavení a rozpočítat chybějící běhy testů na zbývající dny. TestLink by si také měl vzít příklad z průvodce vytvořením vlastních grafů, který je v HP ALM/QC obsažen.

4.2.4 Porovnání reportingu konkurenčních nástrojů

Ze získaných informací vyplynulo, že by si TestLink měl vzít příklad z reportingových modulů nástrojů qTest, HP ALM/QC, TestRail a PractiTest. Proto budou představeny jejich přednosti.

qTest

Tento nástroj pochází z dílny společnosti QASymphony. Skládá se z několika modulů, z nichž pro tuto práci jsou důležité především qTest Manager a qTest Insights. První z nich slouží pro řízení testů a sám o sobě obsahuje mnoho reportů, které si může uživatel dle nabídky upravit a uložit. Pokud je to vhodné, je vždy vykreslen nejprve graf se zvýrazněním důležitých metrik a pod ním přehledná tabulka s daty, ze kterých graf pochází. Reporty jsou přehledně rozděleny do 4 kategorií. Jedná se o reporty pro testovací případy, požadavky, chyby a průběh testů. Celý proces generování a ukládání je velmi intuitivní. Z kategorií si uživatel vybere report, například analýzu běhů testů. Nástroj mu nabídne:

- datové rozmezí pro vykreslení grafu (posledních 7 nebo 30 dnů, od začátku projektu až do dneška, výběr podle začátečního a koncového času);
- vlastnosti pro filtrování výsledků (například status, prostředí, přiřazení);
- typy vybrané vlastnosti, které se mají zobrazit (například jen úspěšné testy nebo přiřazení jen určitým testerům);
- dodatečné pole pro filtrování (například jen manuální testy).

Po nastavení reportu jej uživatel může zobrazit, nebo rovnou uložit s vlastním názvem. Při zobrazení se vykreslí přehledný skládaný sloupcový graf se zvýrazněnými metrikami, týkající se vybraných typů vlastnosti. Pod grafem pak uživatel nalezne tabulku s detailně vypsanými hodnotami. V ostatních reportech lze volitelně měnit grafy dle modulů, typů testovacích případů, priority a důležitosti chyb, stavu chyb, sestavení, testerů, testovacích sad nebo jednotek času (QASYMPHONY, 2017).

Druhý modul qTest Insights se zaměřuje pouze na reporty ze všech ostatních modulů. Na rozdíl od TestLinku tak umožňuje vykreslování grafů i na základě více projektů nebo tvorbu sdílených obrazovek sestavených z vlastních reportů. Modul nabízí i několik předpřipravených tematických obrazovek, týkajících se kvality testů, pokrytí požadavků a práce na tvorbě testů a požadavků. Většina grafů je navíc interaktivních, například při vykreslení skládaného sloupcového grafu lze jednotlivé části sloupců skrývat nebo při

vykreslení koláčového grafu se lze klikem na jednu z výsečí dostat k seznamu případů, které daná výseč zastupuje. Modul též zahrnuje tvorbu vlastních grafů, kde je navíc kromě standardních typů grafů k dispozici i korelační diagram nebo heatmapa.

HP ALM/QC

Nástroj pro řízení testů společnosti Hewlett-Packard (nově Micro Focus) je znám pod starším názvem Quality Center. Od verze 10 se ale stal součástí nástroje určeného k integraci činností celého životního cyklu softwaru s názvem Aplication Lifecycle Management, zkráceně ALM (MICRO FOCUS, 2017). Reporty jsou v nástroji využívány ve dvou modulech. Analysis View slouží k přípravě reportů a Dashboard k jejich zobrazení v sestavách.

Reporty se vytvářejí pomocí průvodce, anglicky nazývaného Graph Wizard. Ten umožňuje vytvořit reporty přednastavené, nebo pomocí 5 kroků navolit zcela vlastní. Při vytváření vlastního grafu je potřeba vybrat základní vykreslovanou entitu. Na výběr má uživatel komponenty, chyby, požadavky, běhy testů, nastavení testů a testovací případy. Podle entity je pak možné vybrat druh grafu. Zde se v nabídce nachází graf přehledový, průběhový a grafy stáří nebo trendu. Graf stáří je nabízen jen pro chyby. V dalším kroku se dá nastavit filtr zobrazovaných dat. Na výběr je z 28 filtrovaných vlastností, mezi které patří například přiřazený uživatel, identifikační číslo související chyby, priorita, důležitost, status, projekt a další. Filtry lze pomocí logických operátorů seskupovat nebo vybírat jen některé. V dalším kroku je možné vybrat, jaká data budou zobrazena na jaké ose, případně dle jakých pravidel se mají seskupovat. Průvodce nabízí mnoho možností, jak vizualizovat prakticky jakákoliv data.

Po vytvoření reportu se zobrazí náhled a lze jej přidat do přehledně rozdělené hierarchie reportů buďto jako privátní nebo sdílený s testovacím týmem. Průvodce se snaží typ grafu nabídnout co nejvhodněji k vybranému typu dat, pokud však uživatel není spokojen, může typ změnit na záložce zobrazení. Většina reportů lze převést na sloupcové, skládané sloupcové, koláčové nebo liniové grafy. V nabídce se nachází i tabulkové zobrazení. Každý typ grafu je navíc možné měnit i graficky. Prostorové grafy změnit zpět na 2D zobrazení, barvu výsečí nebo sloupců libovolně měnit včetně barvy obrysů, posouvat umístění legendy a další.

K přehlednému zobrazení vytvořených grafů slouží modul Dashboard. V něm je možné vytvořené grafy skládat do přehledných celků a určovat jejich rozložení. Celé takto

vtvořené celky lze následně exportovat do formátu PDF. Celkově je reporting nástroje ALM velmi dobře propracovaný.

TestRail

Nástroj pro řízení testů od společnosti Gurock se jmenuje TestRail. Reporty má přehledně rozdělené do kategorií: testovací případy, chyby, výsledky testů, celkové přehledy a uživatelé. Reporty se nezobrazují přímo, ale uživatel musí nejprve potvrdit jejich základní nastavení nebo si nastavení upravit a uložit si report pod vlastním názvem. Zobrazení reportů je možné upravit filtry, výběrem komponent i zvolením sloupců pro tabulkové reporty. Pro každý nový report lze také nastavit možnost sdílení s ostatními členy týmu. Report může být zobrazen rovnou s aktuálními daty, nebo lze naplánovat jeho vytvoření do budoucna, případně jej generovat nastavením cyklu (např. každý týden v pátek odpoledne). Současně s nastavením generování lze také označit členy týmu, kterým má být pravidelně zasílán emailem. Takto vytvořený report se pak zobrazuje na hlavní stránce reportingového modulu, odkud může být rychle upraven.

Report se zobrazuje jako graf a pod ním tabulky s daty, ze kterých byl graf vykreslen. Například přehled aktivity při tvorbě testů zobrazí nejprve skupinový pruhový graf, kdy každý pruh zobrazuje jeden ze stavů testovacího případu pro každý den ve zvoleném období. Pod grafem se nachází první tabulka, ve které je v řádcích uveden den a k němu sledované metriky. V druhé tabulce je uveden přehled testovacích případů seskupených dle stavů vykreslených v grafu (TESTRAIL, 2017).

Na domovské obrazovce nástroj nenabízí možnost její úpravy nebo přidání dodatečných reportů. I tak ale zobrazuje alespoň důležité údaje o průběhu testů za poslední dny a základní přehled aktivity na projektu.

PractiTest

Tento nástroj nabízí hned na domovské obrazovce 8 reportů, které se každých 15 minut samy aktualizují. Jinak je ale jeho reportingový modul zcela prázdný bez jakéhokoliv náznaku možností, které nabízí. Vše odhalí až formulář pro vytvoření nového reportu. Zde je možnost vybrat základní entitu nového reportu z možností: požadavek, testovací případ, výsledek testu, chyba. Dále lze vybrat, zda a podle čeho se má entita vyfiltrovat. Požadavky mohou být filtrovány dle statusu. Testovací případy, výsledky testů a chyby navíc i dle přiřazení testerovi. Také lze vybrat jen určité testovací sady. Poté je nutné zvolit, zda se data

zobrazí ve formě přehledu nebo tabulky. Ve druhém případě lze i zaškrtnou, které sloupce s jakými údaji se mají vypsat. Na závěr se dá k reportu vybrat i některý z grafů z domovské obrazovky, což může uživatele mást, že se každá část výsledného reportu nastavuje na jiném místě (PRACTITEST, 2017).

Grafy se tedy dají vytvářet pouze přes úpravu úvodní obrazovky. I zde jsou pro graf na výběr entity jako při vytváření reportu. Následně ale uživatel musí vybrat, zda chce data vykreslit ve formě koláčového, sloupcového nebo liniového grafu, případně jak vyfiltrovat zobrazená data. U sloupcového grafu lze určit, jaká data se budou zobrazovat na ose X a jaká na ose Y.

4.2.5 Finální výběr reportů

Jediný report, který uživatelé uvedli ve všech třech sledovaných kanálech, byl průběh testů v porovnání s plánem. Ten tedy bude implementován do nástroje TestLink jako samostatný report. Za další nedostatek TestLinku, ale velkou přednost konkurenčních nástrojů, považovali dotázaní možnost vytvořit si vlastní graf pomocí průvodce. V druhé části implementace tedy bude vytvořen základní průvodce, který umožní vygenerovat grafy dle vybraných kritérií.

4.3 Knihovny pro vizualizaci dat

Pomocí SVG, JavaScriptu a CSS lze v prohlížeči vykreslit libovolná data. Je ale zdlouhavé a namáhavé vše nastavit ručně. Proto existují knihovny pro vykreslování grafů, které spoustu práce odvedou za jejich uživatele. V této kapitole bude představena knihovna, kterou používá TestLink. Bude popsáno, co vše umí. Bohužel to nebude dost pro účely této práce. Proto bude zvolena nová, výkonnější knihovna pro implementaci vybraných reportů.

4.3.1 Aktuální knihovna - pChart

TestLink používá pro vykreslení grafů knihovnu pChart. Jedná se objektově orientovaný PHP framework. Liniové grafy zvládne vykreslovat i s plynulými křivkami. Sloupcové grafy nabízí ve skupinové, skládané i normalizovaně poměrné podobě. Stejně tak plošné grafy mohou mít i skládanou podobu. Koláčové i prstencové grafy vytváří v 2D i 3D prostoru. Také dokáže znázornit data pomocí radarových grafů a grafů polárních oblastí.

Z dalších typů grafů jsou zastoupeny bublinové, zónové, svíčkové grafy určené pro data kapitálových trhů nebo grafy rozptylu (PCHART, 2017).

Také grafické možnosti frameworku jsou rozsáhlé. Umožňuje stínovat objekty a nastavovat jim různou průhlednost, upravovat rozlišení os nebo vykreslit více os s různými jednotkami a měnit zobrazení legend. Podporuje navíc podmíněné formátování, které se hodí například u plošných grafů.

4.3.2 Nově použitá knihovna - Highcharts

Norská společnost Highsoft nabízí široké portfolio produktů pro vizualizaci dat. Highstock je knihovna určená pro zobrazování finančních dat. Highmaps je pro změnu určena pro vykreslení geografických dat. Pro tuto práci je však nejzajímavější produkt Highcharts. Ten v době psaní této práce obsahuje prozatím 37 typů grafů a v plánu je jich několik dalších. Kromě všech základních typů, jako jsou koláčové, sloupcové, liniové a plošné grafy, nabízí i grafy využívající přepočet na plynulé křivky, krabicové diagramy, histogramy, Paretův diagram, Sankeyův diagram a mnoho dalších. Typy grafů se také dají libovolně kombinovat do společného zobrazení.

Co se týče vzhledu, může si uživatel nastavit prakticky cokoliv. Kromě barev a obrysů i stíny, přechody nebo grafické vzory. Vše lze nastavit buď přímo při definici grafu, nebo pomocí stažených přednastavených vzhledů a vlastních CSS pravidel. Graf může mít více os i vlastní rozmezí hodnot. Popisky lze nastavit jak statické vedle grafu, tak i dynamické při najetí myší na libovolný bod nebo výseč dat. Všechny textace lze upravovat pomocí HTML tagů. Pro koláčový a sloupcový graf je k dispozici 3D zobrazení a možnost hierarchicky zobrazovat data, kdy při kliku na výseč se překreslí graf podrobnějšími daty dané výseče. Knihovna také nabízí jednoduché nastavení přibližování dat (takzvané zoomování). Velmi silnou vlastností při použití ve webových aplikacích je responzivnost celé knihovny, tedy schopnost přizpůsobit vykreslované elementy různým velikostem displejů.

Mezi pokročilé funkce patří nastavitelnost výkonnosti, kdy vykreslování upřednostňuje zobrazení dat před grafickým zpracováním, možnost překladu všech ovládacích prvků do jiných jazyků, podmíněné vkládání SVG objektů do grafů nebo generování dat do formátu PNG, JPEG, SVG a PDF.

Velkým kladem knihovny je nepřeberné množství funkčních příkladů grafů shromážděných na webu, které uživateli pomohou se rychle zorientovat a nastavit ty správné vlastnosti. Druhým kladem oproti frameworku pChart je možnost ovládat uživatelské kliknutí do prostoru grafu a dle toho mu nabídnout dodatečné nebo detailnější informace.

4.4 Implementace

Testlink používá pro zobrazení dat v prohlížeči šablonovací systém Smarty. Ten umožňuje do HTML kódu vkládat speciální znaky a příkazy a oddělit tak aplikační logiku od prezentace dat. Šablony jsou ukládány v adresáři /gui/templates a pro reporty je zde vytvořena samostatná složka s názvem /results. Pro každou stránku je zde potřeba vytvořit šablonu. Její funkční protiklad, tedy logika v jazyku PHP, je ukládána ve složce /lib a snaží se kopírovat složkovou hierarchii šablon. Pro logiku reportů je proto i zde vytvořena složka /results.

Pro zobrazení odkazů na dostupné reporty slouží soubor resultsNavigator.php umístěný ve složce /gui/templates/results. Ten načítá seznam odkazů z pole reports_list definovaného v konfiguračním souboru custom_reports.cfg.php. Každý záznam musí obsahovat název, adresu url souboru s PHP kódem reportu, formát zobrazení a indikátor, kdo může daný report zobrazit. Název se udává jako proměnná, která pak může být přeložena do požadovaných jazyků. Jazykové překlady jsou uložené ve složce /locale. V základu pracuje TestLink s anglickým jazykem. Pokud tedy nemá uživatel nastaven jiný jazyk, překládá názvy ze souboru string.txt ve složce /locale/en_GB.

4.4.1 Report průběhu testů a porovnání s plánem

Report průběhu testů, anglicky zvaný progress report, zobrazuje pomocí plošného skládaného grafu výsledky testů za každý den od začátku testování až do jejich předpokládaného ukončení. V našem případě se bude jednat o testy náležející vybranému testovacímu plánu až do nejpozdějšího data ukončení nastaveného u aktuálního sestavení. Plán spuštěných testů na každý den tak bude vypočítán takto:

$$\text{Počet spuštěných testů na den} = \frac{\text{Celkový počet testů přiřazených testovacímu plánu}}{\text{Počet dnů od zahájení testování do data ukončení}} \quad (11)$$

Za spuštěné testy se považuje jakýkoliv test s úspěšným, blokovaným i neúspěšným výsledkem. Pro dodržení plánu je tedy nutné aby součet testů s tímto výsledkem byl každý den větší nebo roven plánovanému počtu spuštěných testů, tedy aby skládané části plošného grafu přesahovaly linii plánu. Současně se podíl úspěšných testů musí s blížícím se koncem testů zvětšovat a před tímto koncem ideálně dosáhnout 100%.

Chyby, kvůli kterým jsou testy označeny za neúspěšné, musí být do plánovaného ukončení testů opraveny a testy znovu spuštěny. V případě úspěšné opravy se i výsledek testu změní na úspěšný a databáze bude obsahovat dva záznamy toho samého testu. V grafu však každý den může být uveden jen jeho nejnovější stav.

Získání dat z databáze

TestLink bohužel nenabízí žádnou funkci pro získání počtu spuštěných testů za každý den, neukládá ani takové počty nikam do databáze. Autor této práce ale nechtěl zakládat novou tabulku, do které by se taková data automaticky ukládala, protože by každý, kdo by chtěl nový report používat, musel kromě nahrání nových souborů ještě pracovat s databází. Zvolil tedy přístup, kdy se potřebná data poskládají z výsledků v databázi již zaznamenaných. Tímto způsobem ale lineárně narůstá časová náročnost dotazu, neboť pro každý další den vykreslený v grafu musí dotaz projít všechny dny předchozí, zda některý ze spuštěných testů ten den nebyl pouhým přetestováním a nezapočítal tak jeden test dvakrát.

Pro získání dat z databáze byla v souboru testplan.class.php umístěném v adresáři /lib/functions vytvořena nová funkce get_progress_report_data. Jako parametry přijímá identifikační číslo projektu a testovacího plánu. Obrázek 3 zobrazuje MySQL dotaz pro získání dat, který se z této funkce volá. Zjišťuje, zda je projekt, plán i sestavení aktivní a zda sestavení náleží do plánu a plán do projektu. Následně získává všechny zaznamenané výsledky testů ze zvoleného plánu. Ty seskupuje podle data spuštění testu a zaznamenaného výsledku a takto seskupené záznamy sčítá. Vzniká tak tabulka o pěti sloupcích, kterou funkce posílá v návratové hodnotě s těmito údaji:

- ex_date – datum běhu testů ve formátu YYYY-MM-DD;

- status – výsledek skupiny testů ve formě písmena, p = passed/úspěšný, b = blocked/blokovaný, f = failed/neúspěšný;
- number – počet výsledků s daným statusem v daný den;
- test_project – název testovacího projektu;
- test_plan – název testovacího plánu.

Obrázek 3: MySQL dotaz - získání dat pro graf průběhu testů

```
SELECT ex_date, status, COUNT(*) AS number, test_project, test_plan
FROM (
   SELECT exec_date AS ex_date, e2.tcversion_id, e2.status, test_project,
          test_plan
   FROM (
      SELECT DATE(execution_ts) AS exec_date
      FROM   {$this->tables['executions']}
      GROUP  BY exec_date) AS dates
      RIGHT JOIN (
         SELECT exec.*,nh_tproj.name AS test_project,
                nh_tplan.name AS test_plan
         FROM   {$this->tables['executions']} AS exec,
                {$this->tables['testplans']} AS tplan,
                {$this->tables['builds']} AS builds,
                {$this->tables['testprojects']} AS tproj,
                {$this->tables['nodes_hierarchy']} AS nh_tproj,
                {$this->tables['nodes_hierarchy']} AS nh_tplan
         WHERE  exec.testplan_id = tplan.id
                AND tplan.testproject_id = tproj.id
                AND exec.build_id = builds.id AND nh_tproj.id = tproj.id
                AND nh_tplan.id = tplan.id AND builds.active = 1
                AND tplan.active = 1 AND tproj.active = 1
                AND tproj.id = '1' AND tplan.id = '2') AS e1
         ON dates.exec_date >= DATE(e1.execution_ts)
      JOIN t15executions AS e2
      ON dates.exec_date >= DATE(e2.execution_ts)
         AND e1.tcversion_id = e2.tcversion_id
         AND e1.testplan_id = e2.testplan_id
   GROUP  BY dates.exec_date,e1.tcversion_id,e2.execution_ts
   HAVING e2.execution_ts = Max(e1.execution_ts)) AS results
GROUP  BY results.ex_date,results.status
ORDER  BY results.ex_date
```

Zdroj: vlastní zpracování

Protože je pro bezpečnost aplikace žádoucí, aby případný útočník neznal názvy databázových tabulek, umožňuje TestLink při instalaci přidat před název prefix. Aby pak byl kód univerzální a počítal s tímto prefixem, musí být všechny dotazy do databáze přeloženy pomocí kódu {$this->tables['nazev_tabulky']}.

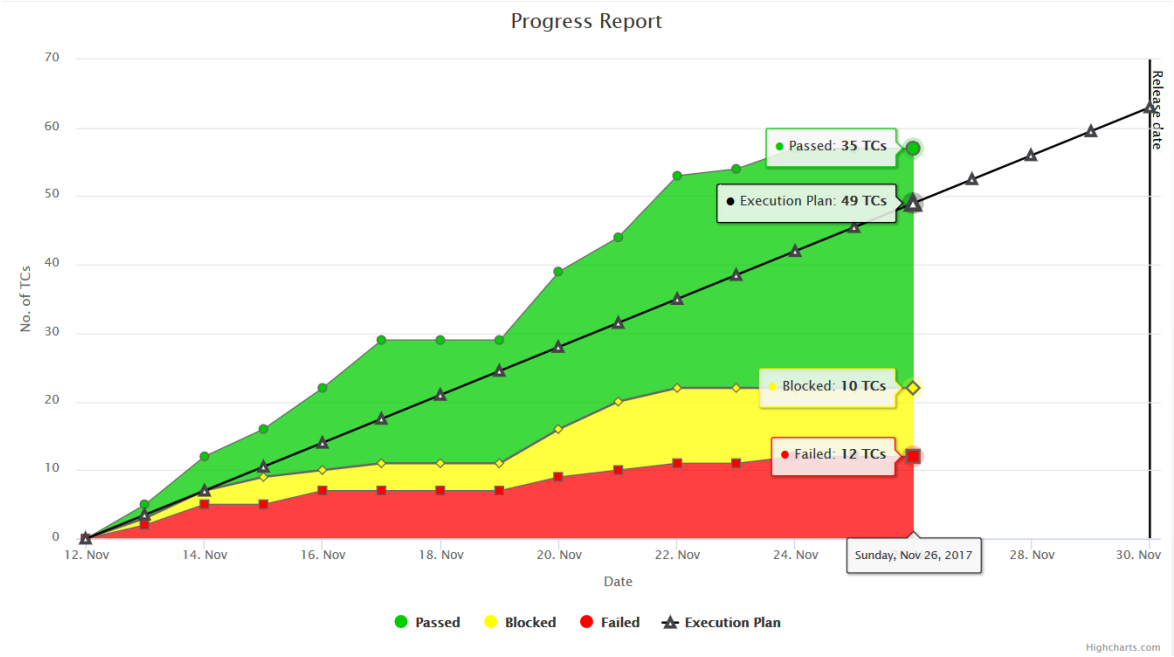
Vykreslení dat

Pro vykreslení dat je potřeba nový soubor s logikou získání dat a soubor šablony, který data zobrazí. Pro reporty týkající se výsledků testů používá TestLink na začátku souboru jmennou konvenci results, celý název souborů tedy bude resultsProgressReport. Identifikační číslo aktuálně testovaného projektu ukládá TestLink do proměnné $_SESSION['testprojectID'] a číslo testovacího plánu lze získat z $_REQUEST['tplan_id']. Tyto hodnoty jsou předávány výše definované funkci get_progress_report_data. Celkový počet testovacích případů přiřazených danému plánu lze získat voláním funkce count_testcases s číslem testovacího plánu. Datum předpokládaného ukončení testů na aktuálním sestavení vrací funkce getBuildByCriteria. Obě tyto funkce jsou taktéž součástí třídy testplan.

Protože testy nemusí probíhat každý den (například o víkendech), funkce get_progress_report_data často nevrátí výsledky pro všechny dny. Taková data by pak při vykreslení grafu způsobila hluchá místa. Proto je pro tyto netestovací dny nutné dopočítat počet již spuštěných testů. Ten se bude vždy rovnat počtu zaznamenanému nejbližší předchozí den testů. Pro získání data všech dnů slouží upravená funkce date_range získaná od uživatele Alioygur (STACKOVERFLOW, 2017). Ta vrátí pole všech dat ve vybraném časovém rozmezí. Pokud je den vykreslování grafu menší než předpokládané ukončení testů, bude se graf vykreslovat až do data ukončení. Knihovna Highchart využívá JavaScript, který používá jiný formát časových dat než MySQL. Před dokončením pole je tedy ještě potřeba data upravit na formát Date.UTC(YYYY,MM,DD). Následně je potřeba založit novou instanci šablonovacího systému Smarty a předat jí všechna důležitá data pro vykreslení.

Šablona resultsProgressReport.tpl obsahuje jediný element div s identifikátorem container a nastavenou výškou 600px, který slouží pro vykreslení grafu. Zbytek souboru obsahuje Javascriptový kód knihovny Highcharts pro vykreslení grafu. Je zde definovaný typ grafu area (plošný), titulek a zobrazení popisku osy y. Nastavením atributu plotOptions/area/stacking na hodnotu normal docílíme zobrazení plošného grafu skládaného. Jednotlivé série dat jsou definovány jménem, odkazem na část pole s daty a barvou plochy.

Obrázek 4: Náhled reportu průběhu testů



Zdroj: vlastní zpracování

4.4.2 Průvodce vytvořením vlastních reportů

Průvodce bude zkonstruován na základě inspirace konkurenčními nástroji. Bude tedy možné libovolně přidávat další grafy a ty již vytvořené znovu měnit a přegenerovávat. Základním výběrem bude entita, k níž bude možné přiřadit vztah a seskupovat data podle vlastností.

Soubor a šablona byly nazvány customCharts.php a customCharts.tpl. Logika v PHP souboru je v tomto případě poměrně chudá, neboť má na starost pouze načíst šablonu s tlačítkem, které bude umožňovat přidání nového grafu. Šablona obsahuje i JavaScriptový kód potřebný pro napojení událostí na tlačítka a AJAXové volání na server. Každé stisknutí tlačítka pro přidání grafu vyvolá nový formulář včetně plochy pro vykreslení grafu. HTML kód formuláře a grafu je umístěn v souboru inc_newChart.php ve složce /lib/results. Aby bylo rozlišeno, s jakou plochou pro graf se právě pracuje, je tomuto souboru předávaná i zvyšující se celočíselná hodnota, která je přidána za název id všech důležitých HTML elementů.

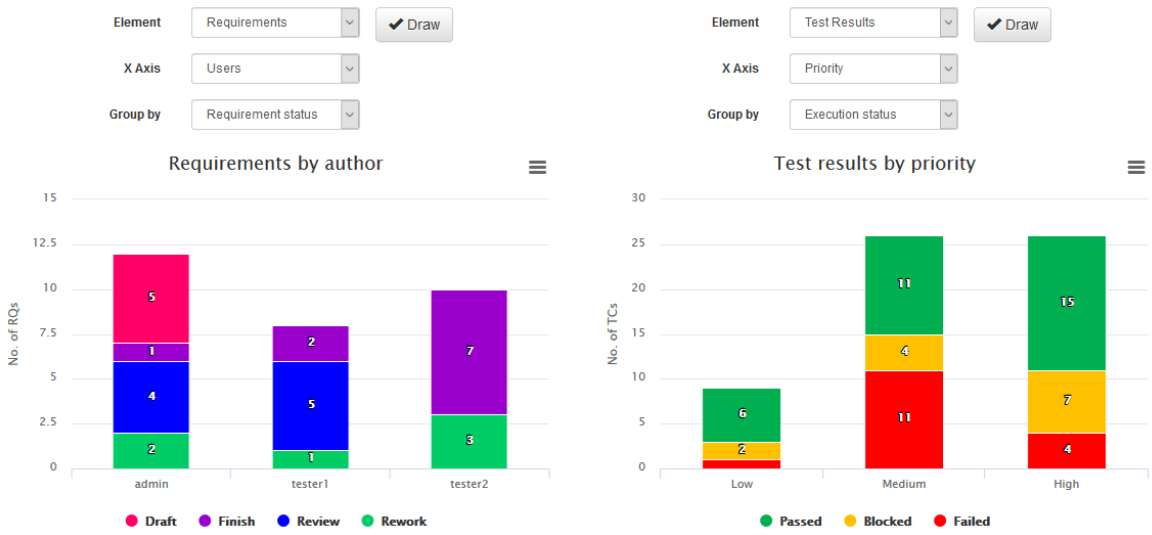
Pokud uživatel vybere požadované hodnoty z formuláře a stiskne tlačítko pro vykreslení grafu, odešle se na server požadavek pomocí AJAXového volání s parametry

předanými pomocí metody GET. Na serveru spravuje požadavky skript s názvem customChartData.php, uložený ve stejné složce jako ostatní logické soubory průvodce. Tento skript se pomocí funkce switch rozhodne, jaká data má vrátit pro vykreslení. Výstup tohoto souboru je ve formátu JSON a obsahuje kompletní nastavení pro požadovaný graf, včetně popisků, barev a podobně. Šablona přijme zpět toto nastavení a vykreslí podle něj graf do vybrané plochy.

V souboru, který vrací zpět data pro vykreslení, jsou připravené funkce, které usnadní budoucí přidávání grafů. Jedná se především o funkci sqlToJson, která výsledky z databáze přeformátuje pro výstup vhodný pro knihovnu Highcharts. Dále jsou ale k dispozici i funkce pro převod typů nebo stavů jednotlivých elementů do uživatelsky čitelné podoby.

Průvodce byl zaznamenán v souboru custom_reports.cfg.php jako položka menu Custom Charts. Pomocí frameworku Bootstrap je responzivně rozdělený na dvě poloviny. Obrázek 5 zobrazuje náhled průvodce, kdy si uživatel vytvořil graf s počtem požadavků dle autora pro každý status požadavku a v pravé polovině graf s počtem výsledků zaznamenaných testů pro každou prioritu seskupený dle stavu výsledku testu.

Obrázek 5: Náhled průvodce vytvořením grafu



Zdroj: vlastní zpracování

4.4.3 Dokumentace k novým reportům

Report průběhu testů a porovnání s plánem najde uživatel v reportingovém modulu pod položkou Progress Report. Vykreslený graf zobrazuje počet spuštěných testů pro každý den projektu a porovnává jej s přímkou plánu testů v závislosti na datu ukončení sestavení. Klikem na typ výsledku testu v legendě lze zobrazovat, nebo naopak skrývat ta data, která uživatele zajímají.

K průvodci vytvořením vlastních reportů se uživatel dostane přes položku Custom Charts. V počátečním stádiu se zobrazí pouze tlačítko pro přidání nového grafu. Po jeho stisknutí se uživateli zobrazí formulář s rozklikávací nabídkou. V poli typu Element jsou na výběr buďto Požadavky nebo Výsledky testů. Tento výběr určí základní data pro vykreslení. V poli X Axis je možné vybrat dodatečný vztah k elementu. Pro požadavky je na výběr vztah s autorem požadavku nebo typem požadavku. Výsledky testů mohou být rozděleny dle uživatele, který test prováděl, priority testu, nebo sestavení do kterého daný test náležel. Pole X Axis i pole Group se dynamicky mění v závislosti na vybrané entitě. V poli Group by je pak na výběr buďto status požadavků výsledků testů. Vždy se tedy vykreslí počet vybraného elementu v závislosti na proměnné osy x, rozdělené dle pole Group by. Celý výběr stačí potvrdit tlačítkem Draw a zvolená data se vykreslí hned pod formulářem.

Po vykreslení grafu jej lze změnit přenastavením vybraných hodnot ve formuláři a opětovným vykreslením tlačítkem Draw. Pro přidání nového grafu stačí zmáčknout tlačítko Add Chart, které se vždy nachází za posledním vykresleným grafem.

5 Zhodnocení výsledků a doporučení

Dotazníkové šetření, komunikace s uživateli nástroje i interview s test manažerem přineslo hodnotné poznatky o nedostatcích reportingového modulu nástroje TestLink, stejně tak o preferencích uživatelů konkurenčních nástrojů. Porovnáním těchto nástrojů byly získány další informace a inspirace pro konečný výběr reportů k implementaci. Jako nejpotřebnější byl zvolen report průběhu testů v porovnání s plánem a průvodce vytvořením vlastních grafů. Správný výběr knihovny pro vizualizaci dat a seznámení se se strukturou databáze TestLinku a jeho šablonovacím systémem umožnilo hladký průběh samotné implementace.

Budoucí rozšíření práce se samo nabízí. Do průvodce může být implementováno více entit, rozšířena možnost filtrování nebo umožnění změny typu grafu vykreslených dat. Porovnávané konkurenční nástroje ukazují, kolik možností vizualizace testovacích dat nabízí.

6 Závěr

Tato práce se zabývala reporty v nástrojích pro řízení testů. V souladu s dílčími cíli byly v rámci studia a analýzy odborných zdrojů získány informace o metodikách testování, řízení testů, nástrojích, reportech a metrikách využívaných při testování softwaru. Získané poznatky a vybrané komunikační kanály byly využity ke zjištění nedostatků reportingového modulu nástroje TestLink a preferencí uživatelů nástrojů pro řízení testů v oblasti reportování. Tyto informace se promítly do výběru nejpotřebnějších reportů, které byly následně implementovány do TestLinku. Tím byl naplněn i hlavní cíl práce. Díky reportu zobrazujícímu průběh testů v porovnání s plánem tak budou mít test manažeři snadnější plánování nejnáročnější fáze životního cyklu softwaru. Průvodce vytvořením vlastních grafů navíc zlepšuje zanedbanou grafickou reprezentaci dat a připravuje nástroj na snadné rozšíření o další užitečné reporty a grafy.

7 Citovaná literatura

ADNAN, F. a NAQVI, I. H., 2015. *Role of software requirements management tools in rework & software project success*. GRIN Verlag. ISBN 9783668052505.

AGARWAL, B. B., TAYAL, S. P. a GUPTA, M., 2010. *Software Engineering and Testing*. Sudbury: Jones and Bartlett Publishers. ISBN 9781934015551.

AINAPURE, B. S., 2009. *Software testing and quality assurance*. 2. Pune: Technical Publications Pune. ISBN 9788184315660.

ASHFAQUE, A., 2010. *Software Testing as a Service*. Boca Raton: Taylor & Francis GRoup. ISBN 9781420099577.

BEIZER, B., 1995. *Black-Box Testing Techniques for Functional Testing of Software and*. John Wiley & Sons, Inc. ISBN 0471120944.

BITNAMI, 2014. *Bitnami Open Source Leaders Interview Series: Francisco Mancardi from TestLink* [online] [cit. 2017-03-15]. Dostupné z: http://blog.bitnami.com/2014/10/bitnami-open-source-leaders-interview_28.html

BLACK, R., 2009. *Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software …*. 3. Wiley. ISBN 9780470404157.

BOEHM, B. W. a kol., 2000. *Software Cost Estimation with COCOMO II*. Pearson Education. ISBN 0137025769.

CaSTB.org, 2014. [ISTQB] In: *Czech and Slovak Testing Board - Standard glossary of terms used in Software Testing* [online]. 28. 3. 2014, verze 2.3 [cit. 2017-02-18]. Dostupné z: http://castb.org/wp-content/uploads/2014/05/istqb_glossary_of_testing_terms_v2.3.pdf

CRAIG, R. D. a JASKIEL, S. P., 2002. *Systematic Software Testing*. London: Artech House Publishers. Stefan P. Jaskiel.

DESAI, S. a SRISTAVA, A., 2016. *SOFTWARE TESTING : A Practical Approach*. 2. Patparganj: PHI Learning. ISBN 9788120352261.

DESIKAN, S. a RAMESH, G., 2006. *Software Testing: Principles and Practice*. New Delhi: Dorling Kindersley. ISBN 9788177581218.

DIETRICH, E., 2014. *Starting to Unit Test: Not as Hard as You Think*. BlogIntoBook.com. B00KIZ6JAC.

DUSTIN, E., 2003. *Effective Software Testing: 50 Specific Ways to Improve Your Testing*. Boston: Pearson Education, Inc. ISBN 9780201794298.

EVERETT, G. D. a MCLEOD, R., 2007. *Software Testing: Testing Across the Entire Software Development Life Cycle*. Hoboken: John Wiley & Sons, Inc. ISBN 9780470146347.

GRAHAM, D. a kol., 2008. *Foundations of Software Testing: ISTQB Certification*. 2. London: Cengage. ISBN 9781844809899.

HAMZAH, H., 2013. *Software Configuration Management Tools*. UMP.

HASS, A. M., 2014. *Guide to Advanced Software Testing*. 2. London: Artech House, Inc. ISBN 9781608078059.

HUTCHENSON, M. L., 2003. *Software Testing Fundamentals: Methods and Metrics*. Indianapolis: Wiley Publishing Inc. 047143020X.

CHEMUTURI, M., 2009. *Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project …*. Fort Lauderdale: J. Ross Publishing. ISBN 9781604270242.

IEEE 1028, 2008. *IEEE Standard for Software Reviews and Audits*. IEEE. Třídící znak 1028.

IEEE 829, 2008. *IEEE Standard for Software and System Test Documentation*. IEEE. Třídící znak 829.

JONES, C., 2007. *Estimating Software Costs: Bringing Realism to Estimating*. 2. New York: McGraw-Hill Companies. ISBN 0071483004.

KAN, S. H., 2002. *Metrics and Models in Software Quality Engineering*. 2. Boston: Addison Wesley. ISBN 9780201729153.

KERZNER, H., 2013. *Project Management Metrics, KPIs and Dashboards*. 2. New York: John Wiley & Sons, Inc. ISBN 978-1-118-52466-4.

LEWIS, W. E., 2017. *Software Testing and Continuous Quality Improvement*. 3. London: Taylor & Francis Ltd . ISBN 9781351722209.

LIMAYE, M. G., 2009. *Software Testing: Principles, Techniques and Tools*. New Delhi: Tata McGraw-Hill Education Private Limited. ISBN 9780070139909.

MAIDASANI, D., 2007. *Software Testing*. New Delhi: Laxmi Publications. ISBN 9788131802823.

MICRO FOCUS, 2017. *ALM Software Trial and Demos* [online] [cit. 2017-10-21]. Dostupné z: https://software.microfocus.com/en-us/software/alm-software-development-testing/try-now

MILI, A. a TCHIER, F., 2015. *Software Testing: Concepts and Operations*. New Jersey: John Wiley & Sons, Inc. ISBN 9781118662878.

MOLYNEAUX, I., 2009. *The Art of Application Performance Testing*. Sebastopol: O'Reilly Media, Inc. ISBN 9780596555436.

PCHART, 2017. *Features* [online] [cit. 2017-11-24]. Dostupné z: http://www.pchart.net/features

PINKSTER, , a kol., 2004. *Successful Test Management: An Integral Approach*. Logica CMG. ISBN 9783540447351.

PRACTITEST, 2017. *Free trial* [online] [cit. 2017-11-01]. Dostupné z: https://prod.practitest.com/

PUSULURI, N. R., 2008. *Software Testing Concepts And Tools*. New DElhi: Dreamtech Press. ISBN 9788177227123.

QASYMPHONY, 2017. *Get Started Today!* [online] [cit. 2017-10-13]. Dostupné z: https://www.qasymphony.com/qtest-trial-qascom/

QATESTINGTOOLS, 2017. *Best Test Management Tool 2017* [online] [cit. 2017-10-25]. Dostupné z: http://www.qatestingtools.com/testing-tool-article/best_test_management_tool_2017

RUBIN, J. a CHISNELL, D., 2011. *Handbook of Usability Testing*. 2. John Wiley & Sons, Inc. ISBN 9781118080405.

SICKINGER, J., 2010. *Risk management in software quality assurance: Risk-Based Testing*. Norderstedt: GRIN Verlag. ISBN 9783640999941.

SOURCEFORGE, 2017. *TestLink Download Statistics* [online] [cit. 2017-11-26]. Dostupné z: https://sourceforge.net/projects/testlink/files/TestLink%201.9/stats/map?dates=2011-01-01+to+2017-11-17

STACKOVERFLOW, 2017. *PHP: Return all dates between two dates in an array* [online] [cit. 2017-02-14]. Dostupné z: https://stackoverflow.com/questions/4312439/php-return-all-dates-between-two-dates-in-an-array

TESTLINK, 2014. *Document TestLink database schema documentation* [online] [cit. 2017-10-14]. Dostupné z: http://mantis.testlink.org/view.php?id=4004

TESTLINK COMMUNITY, 2010. In: *Installation & Configuration Manual* [online]. 01. 03. 2010, verze 2.15 [cit. 2017-11-17]. Dostupné z: https://github.com/TestLinkOpenSourceTRMS/testlink-code/blob/testlink_1_9/docs/testlink_installation_manual.pdf

TESTRAIL, 2017. *TestRail Free Trial* [online] [cit. 2017-10-28]. Dostupné z: https://secure.gurock.com/customers/testrail/trial/

VAN VEENENDAAL, E., POL, M. a TEUNISSEN, R., 2002. *Software Testing: A Guide to the TMap Approach*. Harlow: Pearson Education Limited. ISBN 0201745712.

8 Seznamy

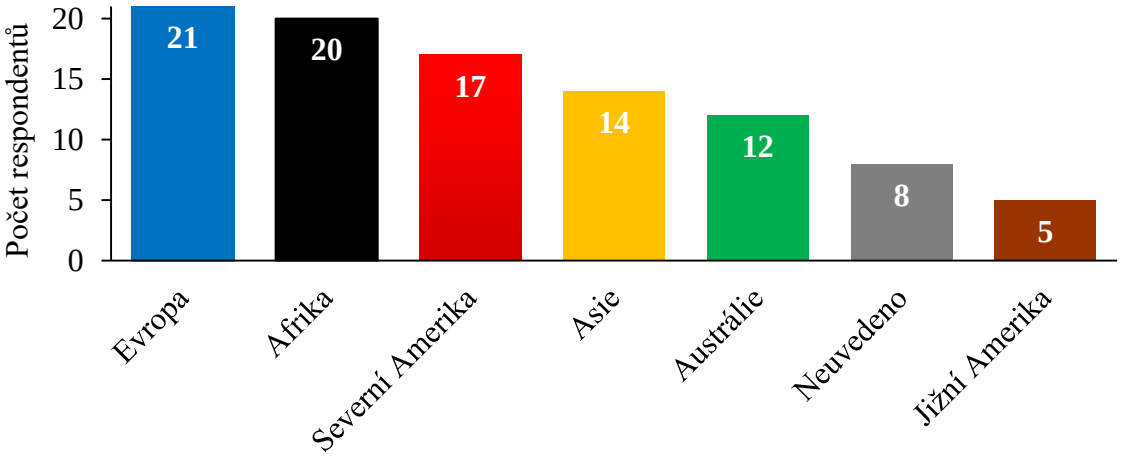
8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam grafů

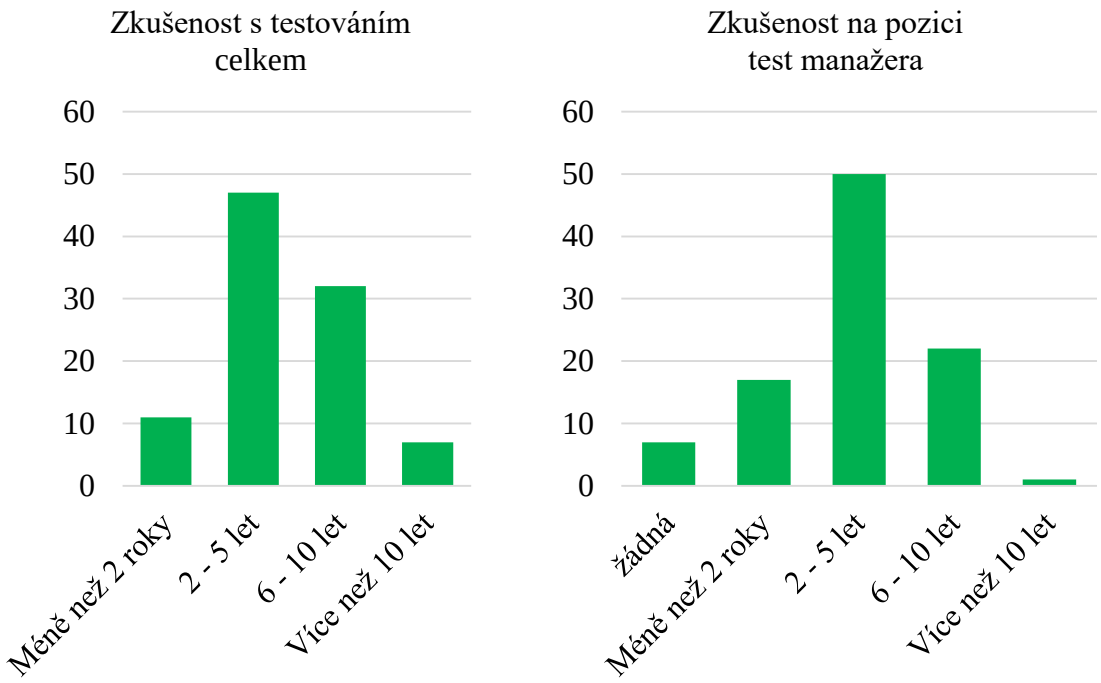
9 Přílohy

Příloha 1: Účastníci dotazníkového šetření dle kontinentu



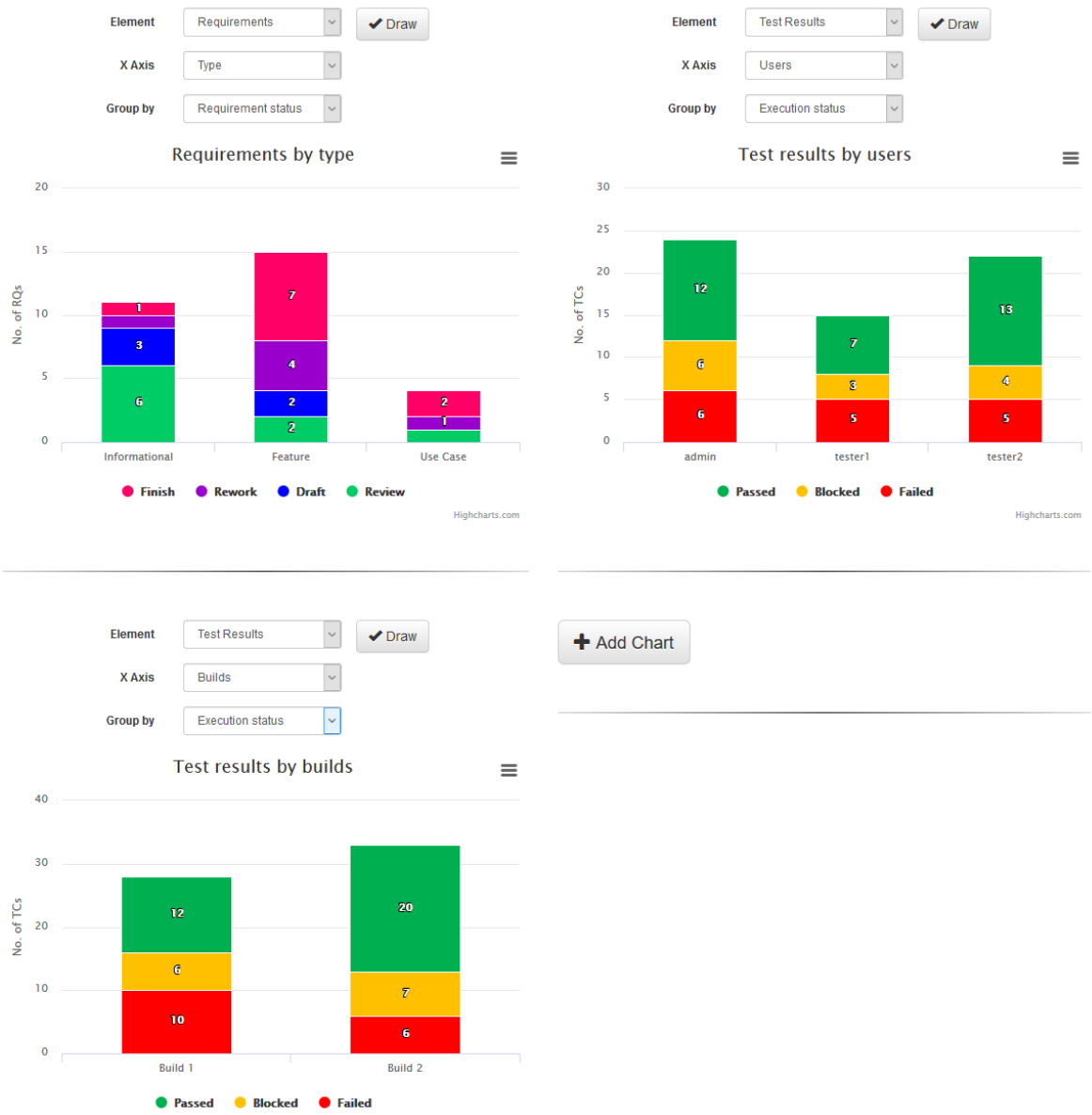
Zdroj: vlastní zpracování

Příloha 2: Zkušenost účastníků dotazníkového šetření s testováním



Zdroj: vlastní zpracování

Příloha 3: Náhled dalších grafů vytvořených v průvodci



Zdroj: vlastní zpracování