VYSOKÁ ŠKOLA EKONOMICKÁ V PRAZE

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informatika

Testování a test reporting v nástroji HP ALM

BAKALÁŘSKÁ PRÁCE

Student : Patrik Hoffmann

Vedoucí : doc. Ing. Alena Buchalcevová, Ph.D.

Oponent : Mgr. Zdeněk Grössl

2015

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 16. prosince 2015 .....

Patrik Hoffmann

Poděkování

Tímto bych rád poděkoval doc. Ing. Aleně Buchalcevové, Ph.D. za obětovaný čas při vedení mé bakalářské práce s uskutečněnými konzultacemi a dále i svojí rodině a přátelům za jejich podporu při studiu.

Abstrakt

Tématem práce je problematika testování softwaru, reportování a zajišťování kvality, což tvoří nedílnou součást vývoje softwaru.

Hlavní cíl práce je směrován na význam test reportingu a vytváření reportů v nástroji HP Application Lifecycle Management. Očekávaným přínosem je možnost využití tutoriálu jak vytvářet reporty a nástěnky v nástroji pro podporu testování. Dále dílčím přínosem práce je rozšíření znalostí z odborné literatury a článků v oblasti testování, zajišťování kvality a setkání s nástrojem HP Application Lifecycle Management.

Práce je rozdělena na části, které obsahují informace o pojmu kvalita softwaru, zadávání požadavků pro zhotovení produktu, rozebrání významu pojmů zajišťování a řízení kvality, verifikace a validace. Rozšiřuje znalosti v rámci oblasti testování softwaru vysvětluje pojmy a popisuje pozice v testovacím týmu. Součástí je i testovací dokumentace využívaná i pro reportování. Důležitou částí testování je test reporting a využívání metrik pro měření kvality. Poslední část je zaměřena na tutoriál pro vytváření reportů a nástěnek v HP Application Lifecycle Management, který může být užitečný pro členy testovacího týmu.

Klíčová slova

Testování softwaru, reportování, HP ALM, zajišťování kvality.

Abstract

Bachelor thesis is devoted to detailed analysis of an issue of software testing, test reporting and quality assurance which is an integral part of software development.

The main objective of this bachelor thesis is to elaborate on the importance of test reporting and report creation in HP Application Lifecycle Management tool. Expected contribution of this thesis is the possibility to utilize the information included as a tutorial for report and dashboard creation as a supportive role for testing. Additionally, the contribution of this thesis is also extension of the the knowledge from literature and articles regarding testing, quality assurance and working with HP Application Lifecycle Management tool.

The thesis is divided into three parts, which contain information about software quality, from assigning the requirement for product delivery to quality control and quality assurance and subsequent verification and validation. It expands knowledge in the area of software testing and explains various roles in the testing team. Thesis also includes testing documentation, which is utilized also for reporting. An important part of the testing is test reporting and employment of underlying metrics for quality measurement. The last part is devoted to the tutorial for report and dashboard creation in HP Application Lifecycle Management, which can be useful for the members of testing team.

Key words

Software testing, test reporting, HP ALM, quality assurance.

Obsah

1. Úvod

Součástí vývoje softwaru je i oblast testování. V době prvních programů bylo testování chápáno spíše jako dokazování správnosti fungování softwaru a v současné době je jeho cíl především v hledání chyb, které je nutné napravit co nejdříve, protože postupem času se náklady na opravu chyb rapidně zvyšují. Testování určuje, zda daný software splňuje uživatelské a technické požadavky, které jsou určeny stranou zadavatele. Proces testování je tedy jedním z klíčových faktorů vývoje, na kterém závisí dosahování kvality a spolehlivosti softwaru. Je prováděno prostřednictvím používání definovaných standardů a nástrojů s cílem zajištění kvality softwaru či systému vyvíjených různými testovacími týmy. Tyto týmy by před samotným procesem testování měly mít znalosti budoucího chování softwaru a také znalosti nástrojů, se kterými pracují.

1.1 Vymezení tématu práce a důvod výběru tématu

Tématem práce je problematika testování softwaru, zajišťování kvality a test reporting. Jedná se o nedílnou součást vývoje softwaru. Díky využití a kvalitního provedení těchto oblastí je možné dosáhnout konkurenceschopnosti na trhu produktů.

Téma bylo vybráno jednak z důvodu zájmu o oblast testování, částečně i z důvodu využití zatím poměrně krátce získaných praktických zkušeností pří mé účasti v testovacím týmu, kde bylo možno prakticky se seznámit s nástroji testování a poznat důležitost i komplikovanost testování pro zajištění kvality a spolehlivosti softwaru.

1.2 Cíle práce a způsoby dosažení

Bakalářská práce je zaměřena na test reporting s cílem vypracování tutoriálu pro vytváření test reportů s příklady v nástroji HP Application Lifecycle Management. Dílčím cílem je přiblížit základní aspekty testování a zajišťování kvality, rozebrat nejdůležitější pojmy, cíle a způsoby testování. Pro splnění cílů jsem čerpal z dostupných metodik a pramenů uvedených v přehledu použité literatury i dalších zdrojů a rovněž ze získaných praktických zkušeností autora v oblasti testování.

1.3 Struktura práce

Bakalářská práce je rozdělena na následující části. První část obsahuje informace o pojmu kvalita softwaru, přiblížení modelu zadávání požadavků pro zhotovení produktu, rozebrání významů pojmů zajišťování kvality a řízení kvality, verifikace a validace.

Další část seznamuje a rozšiřuje znalosti v rámci oblasti testování softwaru, popisuje běžné pozice v testovacím týmu a vysvětluje používané pojmy. Následují otázky, na které je dobré znát odpovědi již před započetím procesu testování.

Práce pokračuje třetí částí, která se zabývá řízením testování, co je vhodné vykonat pro dostatečně kvalitní testovací proces. Součástí této kapitoly je i testovací dokumentace, která je často využívána i pro reportování.

Důležitá je pátá část bakalářské práce, která je zaměřena na teorii procesu test reportingu s uvedením metrik pro měření kvality. Zde jsou popsány některé používané metriky v praxi, které jsou podloženy vzorci pro výpočet. Metriky jsou základními prvky pro vytváření reportů.

Poslední část tvoří práce s nástrojem HP Application Lifecycle Management. Zde se nachází tutoriál jak vytvářet reporty v tomto nástroji. Dále jsou uvedeny některé příklady vytvořených reportů s jejich konfiguracemi a v závěru také návod jak vytvořit nástěnku z jednotlivých nadefinovaných reportů.

1.4 Předpoklady práce

Pro vypracování závěrečné části bakalářské práce bylo vycházeno z praktické činnosti v rámci práce v testovacím týmu.

1.5 Výstupy a přínosy práce

Přínosem bakalářské práce je možnost využití tutoriálu jak vytvářet reporty v nástroji pro podporu testování, a to především pro členy testovacího týmu a test manažery. Dále dílčím přínosem práce je rozšíření znalostí v oblasti testování, zajišťování kvality a setkání s nástrojem HP Application Lifecycle Management.

1.6 Rešerše literatury

V této kapitole je seznam prací s podobným zaměřením. Tento seznam byl sestaven na základě provedeného průzkumu. Rešerše je dělena do třech skupin, kterými jsou: akademické práce, odborné publikace a internetové články.

1.6.1 Akademické práce

Průzkum pro zjištění obdobných akademických prací byl proveden dle internetového vyhledávání a prohledávání v databázích jednotlivých univerzit. K tématu softwarového testování se vztahuje poměrně značná část prací, avšak nebyla nalezena práce, která by byla velmi podobná této práci. Tabulka č. 1 poskytuje seznam nalezených prací, autora a rok obhajoby.

Tabulka 1 - Seznam výsledků průzkumu akademických prací s obdobným tématem (zdroj: autor)

TYP PRÁCE	AUTOR - UNIVERZITA	NÁZEV	ROK OBHAJOBY
DP	BOROVCOVÁ, Anna - UK	Testování webových aplikací	2008
DP	BORŮVKA, Zdeněk - VŠE	Způsoby ověření kvality aplikací a systémů (metodika, nástroje)	2009
DP	FAUSTOVÁ, Tereza - VŠE	Nástroje na podporu testování	2009
DP	KRÁLOVÁ, Iveta - VŠE	Metodika testování podle mezinárodních praktik a standardů	2013
DP	LÍZNER, Tomáš - VŠE	Využití automatizace testování z hlediska nákladů a přínosů	2014
DP	PAVELKA, Tomáš - VŠE	Nástroje pro podporu testování a jejích praktické využití	2014
BP	RYBAK, Yauhen - VŠE	Testování softwaru	2012
DP	ŠPLÍCHALOVÁ, Marcela - VŠE	Metodika testování webových aplikací	2008
DP	VACHALEC, Vladan - VŠE	Testování a kvalita softwaru v metodikách vývoje softwaru	2014

Jednou z velmi přínosných akademických prací pro testování softwaru je diplomová práce Anny Havlíčkové (roz. Borovcová), která se zaměřuje na vysvětlení problematiky testování zejména potencionálním testerům. Popis problematiky je postaven na základě pracovních zkušeností autorky s testováním [1].

Práce zaměřující se spíše na zajišťování kvality vyvíjeného softwaru byla napsána autorem Zdeněkem Borůvkou a má název "Způsoby ověření kvality aplikací a systémů (metodika, nástroje)" [2]. Zmiňuje také jaké metodiky je dobré použít, aby došlo ke snížení rizika výskytu potencionálních chyb softwaru.

Pro seznámení s nástroji na podporu testování je zde práce [3]. V této práci je možné se dozvědět podle jakých kritérii je dobře vybrat nástroj na podporu testování. Tomáš Pavelka napsal práci podobného tématu [4], avšak jeho práce je více zaměřena na praktické využití nástroje HP Quality Center 11.

Další diplomová práce má název "Metodika testování podle mezinárodních praktik a standardů" od Ivety Králové [5], jejímž cílem je vytvoření metodiky v češtině, která vychází z mezinárodních praktik a standardů. Tato metodika je vytvořena na procesu testování definovaným mezinárodní organizací ISTQB.

Autor Tomáš Lízner [6] se ve své práci zabývá obdobně jako je tomu v předchozím případě vytvořením metodiky, která je vhodná pro automatizované testování. Ke konci práce vytvořenou metodiku ověří na praktickém projektu.

Pro praktické použití testovacího nástroje IBM Rational Functional Tester je vytvořena práce [7], jejímž cílem bylo vytvořit příručku pro použití. V úvodu práce jsou popsány teoretické oblasti testování.

Problematikou testování softwaru se zabývá práce Marcely Šplíchalové [8]. Hlavním cílem této práce bylo vytvoření jednotného metodického rámce pro menší testovací oddělení a prohloubit znalosti čtenářů o pojmu softwarová chyba.

Vladan Vachalec se ve své práci [9] zabývá pojmem kvalita softwaru a její dosažení a dále testováním při vývoji softwaru. Praktickou částí je rozšířena metodika MMSP (metodika pro malé softwarové projekty).

1.6.2 Odborné publikace

V oblasti softwarového testování se vyskytují publikace, které se zabývají především touto oblastí komplexně. V českém jazyce se mnoho literatury nevyskytuje, ale je třeba zmínit publikaci jakou je Řízení kvality softwaru - průvodce testováním [10]. Další publikací tentokrát pouze přeloženou je Testování softwaru [11]. Pro čerpání více informací bylo potřeba se zaměřit i na publikace v jazyce anglickém, mezi které patří Testing computer software [12], dále také Manage software testing [13] a v neposlední řadě Effective methods for software testing [14]. Pro kompletní zjištění informací byly tyto publikace často doplňovány internetovými zdroji.

1.6.3 Internetové zdroje

Pro doplnění knižních a akademických prací je třeba použít internetové zdroje. Tyto zdroje se skládají z částí jednotlivých odkazů na články, ale například se jedná i o konkrétní webové blogy, server, věnující se softwarovému testování. Mezi tyto hlavní tři zdroje patří: Testing Excellence [15], blog o pro pomoc při testování [16] a jako poslední web, který připravuje uchazeče na zkoušky pro certifikát ISTQB - istqbexamcertification.com.

2. Kvalita softwaru

V současné době se téměř každý člověk setkává se softwarem v každodenním životě. Trh produktů je plný konkurence a proto výrobci a vydavatelé, kteří chtějí být se svými produkty úspěšní, musí zajistit co nejlepší vlastnosti produktu ještě před jeho vydáním. Je zřejmé, že nedostatečně kvalitní software nebude tak hojně využíván, jako konkurenční více kvalitní software. Pro oblast softwarového testování je nutné uvést pojem "kvalita softwaru". Vztah mezi tímto pojmem a testováním softwaru je velmi důležitý, protože z velké části díky testování softwaru je zajišťována jeho kvalita. Tato kapitola se zabývá definicí pojmu kvalita, dle jakého modelu je možno roztřídit jednotlivé požadavky na produkt, vysvětluje rozdíly mezi pojmy zajišťování a řízení kvality a dále rozlišuje pojmy verifikace a validace.

2.1 Pojem kvalita

Význam samotného slova "kvalita" může být různorodý. Například Philip B. Crosby, byznysmen, který ovlivnil zejména oblast řízení kvality, definuje kvalitu jako "soulad s požadavky" [17]. Pravděpodobně nejvíce odborná definice slova "kvalita" je v normě ISO 8402-1986 ve znění *"The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs."* V češtině tedy jako souhrn vlastností nebo charakteristik produktu či služby, který souvisí se schopností splnit explicitně uvedené či implicitně předpokládané potřeby. Cílem by se dalo jednoduše říci, že kvalitou se rozumí uspokojený zákazník [18].

2.2 Model FURPS pro zajištění kvality

Model FURPS je vytvořený společností Hewlett-Packard a díky jemu je možno roztřídit jednotlivé požadavky na dodávaný produkt do kategorií tak, aby došlo k celému pokrytí všech oblastí funkcionálních a nefunkcionálních požadavků [19]. Mezi funkcionální požadavky se řadí funkcionalita systému či případně jeho komponent, popis vstupu, chování systému, výstupu. Naopak nefunkcionálními požadavky jsou myšleny požadavky na návrh, služby, výkonnostní požadavky na bezpečnost, dostupnost, stabilitu a dále například design, uživatelská přívětivost, dokumentace, licence, testovatelnost, cena atd. [20].

Název modelu FURPS je zkrácený tvar začínajících písmen kategorií požadavků [10],[19],[21],[22]:

- **Functionality** (funkčnost)

  Tato kategorie je zaměřena na hlavní funkčnost a schopnost programu. Jedná se o požadavky, které jsou vyžadovány ze strany zákazníka. Při nesprávném pochopení zadání vývojovým týmem může dojít k problémům typu opožděného dokončení či zvýšení nákladů. Proto je velmi důležité, aby ze strany zadavatele bylo zadání jednoznačné a přesně definované. Z druhé strany je třeba informovat zadavatele podklady o dostatečném porozumění zadání.

- **Usability** (použitelnost)

  Vnímání, zda je systém dostatečné efektivní pro koncového uživatele, který systém používá. Celkový dojem působení systému, snadnost použití, uživatelská přívětivost, estetika, dokumentace a podobně.

- **Reliability** (spolehlivost)

  Mezi bod spolehlivosti patří četnost a závažnost chyb, maximální doba nečinnosti, zda jsou potencionální selhání předvídatelné. Pro měření spolehlivosti se často používá metrika MTBF (mean time between failures), tedy střední doba mezi poruchami.

- **Performance** (výkonnost)

  Důležitou oblastí požadavků jsou požadavky na výkonnost za různých podmínek. Například v případě využívání softwaru, systému či aplikace více uživateli najednou ve stejném čase. Sledují se technické parametry, např. vytížení zdrojů OS, rozložení zátěže, maximální doba odezvy, spotřeba paměti.

- **Supportability** (rozšiřitelnost / podporovatelnost)

  Hodnocením je, zda systém je testovatelný, rozšiřitelný o nové vlastnosti, provozuschopný, přizpůsobitelný, zapojitelný do existujících procesů podpory a údržby.

Vzhledem k tomu, že model FURPS byl vytvořen v dávné historii, konkrétně první zmínky v roce 1986 a v literatuře se objevil až o rok později, v roce 1987 od autorů Robert Grady a Deborah Caswell v publikaci "Software Metrics: Establishing a Company - Wide Program" [5], v dnešní době se můžeme setkat spíše s verzí rozšířenou o další kategorie, která nese název FURPS+. Mezi tyto kategorie patří:

- **Design constraints** (omezení návrhu)

  Určitá omezení návrhu kladena zadavatelem na daný software, systém či aplikaci.

- **Implementation requirements** (požadavky na implementaci)

  Požadavky na určité programovací jazyky, standardy, speciální nástroje atp. Jsou určeny především pro vývojový tým.

- **Interface constraints** (omezení na rozhraní)

  Definuje požadovaná omezení pro rozhraní a dále popisuje systémy, se kterými bude finální produkt interagovat.

- **Physical requirements** (požadavky na fyzické vlastnosti)

  Požadavky například na váhu, materiál, vzhled a rozměry v případě, že produkt je hardware.

2.3 Zajišťování a řízení kvality

Často se vyskytuje případ, kdy se lidé domnívají, že softwarové testování se shoduje s pojmem zajišťování kvality. Již zmíněné pojmy zajišťování kvality (Quality Assurance) a řízení kvality (Quality Control) je třeba v této problematice rozlišovat a pochopit. Následuje porovnání faktických informací k těmto pojmům [23]:

Definice

Quality Assurance: Soubor aktivit zajišťující kvalitu v procesech cyklu vývoje produktu.

Quality Control: Soubor aktivit zaměřených na identifikování defektů v již vytvořených produktech k zajištění kvality.

Zaměřenost a cíl

Quality Assurance: Předcházení defektům se zaměřením na procesy v rámci vytváření produktu. Cílem je zlepšit vývojové a testovací postupy tak, aby nedocházelo k výskytu defektů ve fázi vývoje.

Quality Control: Zaměřeno na nalezení a opravu defektů v již hotovém produktu, ale ještě před fází finálního vydání (releasu).

Jak toho docílit?

Quality Assurance: Vybudováním dostatečně kvalitního managementu a systému vyhodnocování adekvátnosti.

Quality Control: Nalézáním a eliminováním příčin vzniku problémů kvality skrze využívané nástroje.

Kdo je za to zodpovědný?

Quality Assurance: Každý v týmu, který je zahrnut ve fázi životního cyklu produktu.

Quality Control: Obvykle je zodpovědný určitý tým, který testuje produkt pro vyhledávání defektů.

Příklady

Quality Assurance: Verifikace - ověřování.

Quality Control: Validace / Testování softwaru.

2.4 Pojmy verifikace a validace

Dalšími pojmy v oblasti testování jsou verifikace a validace (V&V). Opět i v tomto případě dochází k nejasnostem chápání významu pojmů a častému zaměňování [11].

Neformálními, uživateli oblíbenými, definicemi Barryho W. Boehma [11] těchto dvou pojmů jsou:

- **Verifikace:** Vytvářím produkt správně?
- **Validace:** Vytvářím správný produkt?

Verifikací se rozumí proces, kterým je třeba se ujistit, zda daný produkt splňuje předem specifikované požadavky již ve fázi vývoje. Proces předchází validaci a je ověřován Quality Assurance týmem, který provádí prozkoumání, porady a kontrolu shody výstupů se vstupy. Předmětem těchto aktivit jsou různé plány, požadavky, specifikace pro návrh, zdrojové kódy, testovací případy a další.

Po verifikaci následuje proces validace, který slouží pro vyhodnocení produktu na konci fáze vývoje, kde se střetne konečný produkt s očekáváními zákazníka a jeho požadavky. Proces je prováděn testovacím týmem, který testuje výsledný produkt různými druhy testování. Zajímavým faktorem je, že chyby nalezené během validace jsou daleko nákladnější než chyby nalezené verifikací [24].

Pro objasnění pojmů verifikace a validace na jednoduchém praktickém příkladu je možné použít myšlenku autora internetového článku N. Belkhatouia [25]:

Při testování aplikace náležitosti říkají, že v aplikaci se nachází dvě pole ("Jméno" a "Příjmení") k vyplnění maximálně 64 znaky a dále tlačítko "Uložit". Zadání testu testerovi říká, aby vyplnil políčka "Jméno" a "Příjmení" a stiskl tlačítko "Uložit". V případě kontroly přítomnosti polí "Jméno", "Příjmení" a tlačítka "Uložit" by uživatel prováděl proces verifikace, a naopak, kdyby uživatel následoval kroky zadané případem užití, byl by to proces validace.

Vztah nadřazenosti a podřazenosti pojmů QA, V&V a testování je vyznačen na obrázku č. 1.

Obr. 1 - Schéma vztahu QA, V&V, testování (zdroj: [18], upraveno autorem)

3. Testování softwaru

Kapitola třetí se věnuje teorii testování softwaru. Uvádí jaké role a úkoly mají členové testovacího týmu, vymezuje základní pojmy testování, důvody a způsoby testování, které je třeba při čtení bakalářské práce znát.

Nedílnou součástí procesu řízení kvality produktu (Quality Control) je testování. K tomuto pojmu existuje spousta různých definic a názorů. Samotné testování není aktivita, která by byla vykonávána nahodile, proto tomu předchází analýzy a plánování ve formách návrhu testů. Definice testování dle knihy [10] je, *proces řízeného spouštění softwarového produktu s cílem zjistit, zda splňuje specifikované či implicitní potřeby uživatelů*. Například podle článku [26] je definici možné rozdělit do následujících osmi částí:

- **Proces** - testování je spíše proces, než jedna aktivita

- **Aktivity životního cyklu** - testování je součástí celého životního cyklu vývoje SW

- **Statické testování** - je možné testovat a hledat chyby i v případě, že zdrojový kód ještě není vytvořen. Statické testování je dokončeno během procesu verifikace a zahrnuje například kontrolu a analýzu dokumentů. Jedná se o efektivní způsob testování z hlediska vynaložených nákladů.

- **Dynamické testování** - v tomto případě je kód vytvořen a je vykonáno během procesu validace. Patří mezi to jednotkové testy, integrační testy a systémové testy.

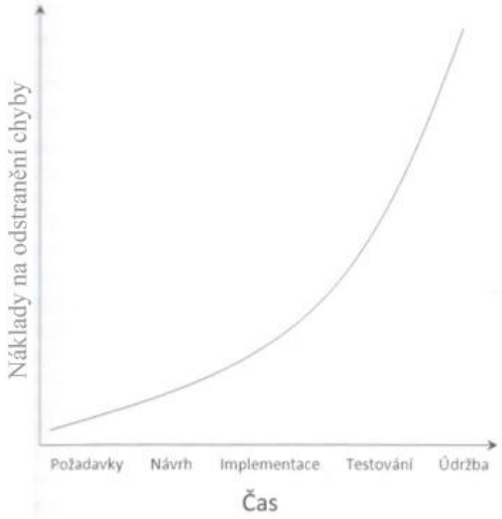
- **Plánování** - jak již bylo zmíněno výše, testování není nahodilou aktivitou a je třeba určit co se chce vykonat

- **Příprava** - kvalitní příprava je také výsledkem úspěšného testování. Je třeba konkrétně určit výběrem podmínek a tvorbou testovacích případů co se bude testovat.

- **Vyhodnocení** - během vyhodnocování se kontrolují výsledky testování a posuzuje se daný software kompletačními kritérii. Díky této části je možné zjistit, zda testování softwaru bylo kompletně ukončeno a zda produkt zdárně prošel testy.

- **Produkt a související soubory** - je dobré si uvědomit, že otestováním kódu, testovacích požadavků a specifikace návrhu není testování ukončeno. Neměla by se opomenout kontrola dokumentů souvisejících s produktem, jako jsou například materiály určené pro uživatele ke školení.

Proces důkladného otestování softwaru by rozhodně neměl být testovacím týmem podceňován. Jak je známé praxí, tester je také pouze člověk a ne stroj, proto je dobré jednotlivé testy opakovat ve více kolech. Testy, které byly vykonány testerem "A", by ve druhém kole měly být otestovány testerem "B". Dle grafu (obr. 2) z knihy [10] je křivka mezi náklady a chybou nalezenou v určité fázi vývoje softwaru vzrůstající. To znamená, že fáze vývoje, ve které byla chyba nalezena je velmi důležitým faktorem na vynaložení nákladů. Čím dříve je chyba testery odhalena, tím více se ušetří za náklady vynaložené k odstranění.



Obr. 2 - Vztah mezi náklady a fázemi vývoje (zdroj: [10], upraveno autorem)

3.1 Testovací tým

Tato podkapitola informuje kdo všechno je součástí testovacího týmu a jaká je jejich role, zodpovědnost a jaké jsou vyžadované dovednosti pro danou pozici.

Manažer testování (Test Manager) je nejzodpovědnějším členem testovacího týmu a zodpovídá za správné vedení celého projektu. Často bývá členem hlavní komunikace se zákazníkem, organizuje a účastní se schůzí před a po testování, má hlavní slovo pro nábor nových členů do testovacího týmu, které dále zaškoluje a dohlíží nad podřízenými. Mezi jeho

zodpovědnosti patří plánování testování, dosažení naplánovaných cílů a strategií testování, výběr nástrojů a metrik, které budou pro proces použity. Je potřeba, aby rozuměl testovacím metodám a procesům, měl dostatečné zkušenosti s řízením, znal osvědčené praktiky pro manuální a automatizované testování, porozuměl byznysové části projektu a požadavkům zadaných ze strany zákazníka. [10], [27]

Vedoucí testování (Team Lead) je jako předešlá pozice také manažerská pozice, ale v technické oblasti. Vedoucí testování je členem týmu často při práci na větších projektech, kdy je daný projekt rozdělen na menší části. Mezi jeho úkoly patří vytvoření testovacího plánu, který je později schvalován či odmítnut test manažerem. V rámci svého týmu zadává úkoly, má přehled o podřízených, dohlíží na jejich práci. Požadavky na jeho manažerské dovednosti nebudou tak vysoké, jako je tomu u pozice test manažera, ale na druhou stranu je od něho vyžadováno více technických schopností, protože se může podílet na samotném procesu testování a tvorbě testů. [10], [27]

Analytik testování (Test Analyst / Test Engineer) je pozice, bez které by nemohlo dojít k procesu testování, protože analytik vytváří testovací scénáře s jednotlivými testovacími případy, které jsou následně testery otestovány. Této aktivitě se říká test design. Jeho úkol není snadný, je třeba znát veškerou funkčnost programu a příslušné dokumentace. Analytik například píše testy pomocí čtení z UML diagramů a reportuje stav vedení o jeho práci. [10]

Tester má za úkol vykonávat jednotlivé testy, které jsou napsány analytiky. Tato pozice může být různě zaměřena. Nejznámější je dvojí rozdělení pozice pro tzv. automatizované a manuální testování. Požadavky na testera automatizovaného testování jsou daleko vyšší než na testera pro manuální testování. Softwarový tester je velmi kreativní osobou, která se snaží přijít věcem na kloub. Odhaluje a nahlašuje nalezené chyby softwaru vývojovému týmu, které jsou následně navráceny testerovi pro znovuotestování, dokud nebude funkčnost softwaru taková, jaká je uvedena ve specifikaci. [10], [11]

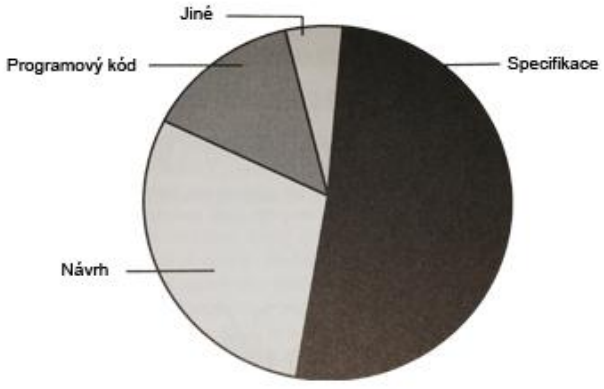
3.2 Základní pojmy testování

V oblasti informačních technologií je mnoho pojmů, které se používají v originálním anglickém znění i v češtině. Pouhé přeložení pojmů z angličtiny do češtiny pravý význam nevyjasní. Z toho důvodu bude dobré si alespoň část základních pojmů vysvětlit v této kapitole.

Bug (chyba, defekt) - je trhlina v softwaru, která vznikla ve fázi vývoje například vlastní chybou programátora. Testeři v procesu testování mají za úkol tyto bugy vyhledávat a co nejrychleji je nahlašovat, aby došlo k brzké opravě za nižší náklady (obr. 2). Nalezený bug tester popíše srozumitelně do tzv. bug reportu, aby programátor jasně pochopil o co se jedná. Následuje informování testera ze strany programátora o opravení bugu, tester musí problém opětovně otestovat, zda vše funguje dle specifikace. Nemělo by se zapomínat na přetestování již prošlých testů, protože se může stát, že opravením jedné chyby vznikne chyba druhá. Ron Patton ve své knize [11] popisuje situace, při kterých dochází k nalezení bugu:

- *Software nedělá něco, co by podle specifikace produktu dělat měl.*
- *Software dělá něco, co by podle údajů specifikace produktu dělat neměl.*
- *Software dělá něco, o čem se produktová specifikace nezmiňuje.*
- *Software nedělá něco, o čem se produktová specifikace nezmiňuje, ale měla by se zmiňovat.*
- *Software je obtížně srozumitelný, těžko se s ním pracuje, je pomalý, nebo - podle názoru testera softwaru - jej koncový uživatel nebude považovat za správný.*

Další zajímavou skutečností Ron Patton [11] oznamuje, že proběhlo mnoho studií na téma z jakého důvodu vznikají chyby a co je tedy hlavní příčinou vzniku (obr. 3)



Obr. 3 - Hlavní příčiny vzniku chyb (zdroj: [11])

Test Case (testovací případ) - je dokument, který se skládá z testovacích dat, předpokladů a očekávaných výsledků. Případ je vytvořen v konkrétním testovacím scénáři za účelem ověření, zda je činnost v souladu se stanoveným zadáním [28]. Pro ujasnění za použití reálné situace má tester například zkontrolovat, zda je funkčnost tlačítek v aplikaci v souladu s požadavky. Nejprve je třeba splnit předpoklady pro daný testovací případ, protože tlačítka

mohou být zobrazeny pouze pro určitou část uživatelů. Po splnění si tester přečte popis, který mu oznámí, že má vyplnit dvě pole a stisknout tlačítko "Uložit". Dalším krokem je si přečíst, jaký je očekávaný výstup testovacího případu. V této situaci po stisknutí tlačítka dojde k uložení zadaných dat a následnému stažení PDF souboru. Pokud vše je v pořádku podle testovacího případu, tester označí tento případ jako úspěšný (passed), v opačném případě označí případ jako neúspěšný (failed) a reportuje chybu.

Test Data (testovací data) - jsou skutečná nebo fiktivně vytvořená data, která jsou využívána v rámci testování produktu [29].

Test Design - je pojem, kterým se označuje činnost analytiků testování, tedy tvorba testovacích scénářů a testovacích případů.

Test Log - je soubor, kde je zaznamenáván průběh testovacích případů, zda je daný případ úspěšný či neúspěšný (passed / failed).

Test Script (testovací skript) - je označení pro skupinu uspořádaných testovacích případů [10].

3.3 Důvody a způsoby testování

Lidé jsou stále pouze osoby, na které působí při práci mnoho vlivů, jako například stres, únava a další. Z toho důvodu je téměř nemožné být při výkonu práce tak soustředěný, aby člověk někdy neudělal nějakou chybu. Chyba může vypadat ze začátku jako nevýznamná, avšak později se prokáže, že se stala nebezpečnou a velmi nákladnou. Proto je potřeba svou práci kontrolovat pro brzké odhalení a následné opravení. Problém je v tom, že člověk se v kontrolování opakuje, tzn. jednou přehlédne například malinkou část, tzv. slepé místo softwaru, kde se zrovna chyba vyskytuje a neodhalí ji, proto zkusí překontrolovat oblast znovu, ale opět chybu přehlédne. Pro nejefektivnější odhalení chyb je vhodné zkusit předat část své práce někomu, kdo se v dané oblasti vyzná a zkusí ji zkontrolovat po svém. V tomto případě jde tedy o testování více lidmi, kde každý člověk se dívá na danou věc jiným pohledem. Existuje několik důvodu proč testovat. Článek [30] zmiňuje 7 hlavních důvodů proč je testování softwaru důležité:

1) Je důležité pro identifikování vad a chyb, které vznikly ve fázi vývoje.

2) Úspěšné otestování uspokojuje zákazníka.

3) Díky dodávání kvalitních produktů klientům se dosahuje jejich důvěry.

4) Je nezbytné pro poskytnutí vysoce kvalitního produktu nebo například produktu, který sníží náklady zákazníka.

5) Bez provedeného testování daný produkt nebude tak výkonný.

6) Po úspěšně dokončeném procesu testování je snadnější určit, že se aplikace nebude potýkat se selháními, které by mohly v budoucnu znamenat daleko vyšší náklady na opravu.

7) Je třeba testovat a dělat kvalitní software a být tak konkurenceschopný.

Pro testování softwaru je možné využít mnoho různých technik. Existují techniky, které jsou zaměřeny na určitou fázi testování. Bohužel však neexistuje jen jedna technika, kterou by bylo možné použít pro kompletní otestování a nalezení všech existujících bugů, proto klíčem úspěchu je použití více technik. Peter Farrell-Vinay uvádí [13] základní rozdělení technik do následujících bodů, které by bylo vhodné přiblížit.

Black-box (metoda černé skříňky) - po skončení vývoje naprogramováním je třeba posunout produkt testovacímu týmu, který má za úkol odhalovat chyby produktu. V praxi často nastávají vedlejší účinky, kde při opravení jedné chyby mohou vzniknout chyby nové. Metoda černé skříňky je metoda testování, kdy tester testuje pomyslnou černou skříňku, do které nevidí. Tester vidí pouze vstupy a výstupy. Během testování tester nemá žádné povědomí o tom, jak systém funguje uvnitř, to znamená, že při testování nevidí do zdrojového kódu, proto je zde důležitá přítomnost přesné specifikace. Je označována jako metodou funkcionální či metodou za přítomnosti specifikace. Metoda je vhodná k použití pro všechny úrovně testování, od testování komponent, až po akceptační testy [13], [31], [32].

White-box (metoda bílé skříňky) - je opakem metody černé skříňky, nejméně nákladná je v případě, kdy se testují programy, které jsou psané v strukturovaném jazyce, proto se označuje jako metoda strukturální. K procesu testování v tomto případě mají testeři náhled do implementace programu a nezjišťují co je výsledkem, ale jakým způsobem se dosáhlo výsledku. Jako tomu je i v předešlém případě, testování metodou bílé skříňky je možné použit na mnoho úrovní testování, jako například testování komponent, systémové testování a akceptační testování [13], [33].

Statické testování - tímto pojmem se rozumí kontrolování něčeho, co není zcela vytvořeno či momentálně neběží. Jedná se například o zkoumání a manuální kontrolu dokumentů.

Testování začíná v brzkém stádiu životního cyklu produktu a je ukončeno během procesu verifikace. Pokud se jedná o testování softwaru, v tomto případě testování není zapotřebí používat počítač. Ron Patton dává za příklad jednoduchou životní situaci, kdy se člověk chystá staticky otestovat svůj automobil, to znamená, že se chystá zjistit, zda kola drží, zda má automobil lak ve správném stavu, otevře kapotu auta a kontroluje hladinu kapalin a podobně [11], [34], [35].

Dynamické testování - je testování za chodu softwaru či produktu. Software je testován procházením a kontrolováním na počítači za pomocí jednotkového, integračního nebo systémového testování. Dynamické testování je zhotoveno ve fázi validace. Je možné kontrolovat například funkční chování softwaru, využívání paměti a procesoru a celkový výkon systému. Pro porovnání rozdílu mezi dynamickým a statickým je zde i situace, kdy člověk testuje automobil. V dynamickém testování automobilu je myšleno kdy člověk zkouší hlavní funkčnost, tedy nastoupí do automobilu a provede zkušební jízdu [11], [35], [36].

3.4 Zahájení a ukončení procesu testování

Před předložením finálního produktu zákazníkovi je třeba dosáhnout určité kvality. Bez testování se dosažení kvality neobejde. Na otázku kdy začít testovat pravděpodobně neexistuje žádná konkrétní odpověď jakou by mohlo být například kterým dnem začít, záleží tak na mnoha vlivech. Z předešlých kapitol a praktických znalostí je však známé, že je třeba začít co možno včas, aby došlo k vyvarování nalezení chyb v pozdějších fázích vývojového cyklu, kdy je oprava na chyby daleko nákladnější, než ve fázích raných. Na druhou stranu začít s testováním moc brzy je také problémem, protože software nemusí být ještě stabilní a bude se tak v něm vyskytovat příliš mnoho chyb. Toto testování by tak zbytečně marnilo čas testerů a zvyšovalo náklady na projekt.

Autor Peter Farrell-Vinay ve své knize [13] uvádí několik situací, ve kterých je vhodné začít s procesem testování:

- Po skončení fáze programování kódu, kdy každá část bude testována samostatně.
- Na konci integrační fáze, kde dojde k otestování celého systému, aby systém prokázal, že každý jeho prvek funguje tak, jak je požadováno.
- Co nejdříve, kdy se stane systém dostatečné stabilním a dostupným, následuje testování zda je chování uspokojující.

Proces testování by se neměl podcenit, proto je opravdu nutné provést důkladnou kontrolu produktu. Často daleko výhodnější je odložení releasu (data vydání produktu), než uspěchání procesu testování. Jako na otázku, kdy začít s testováním, není ani zde na otázku, kdy s testováním skončit, jasná odpověď, avšak je možné najít některé zajímavé odpovědi [13]:

- Při uvědomění si, že systém obsahuje mnoho chyb a další testování by bylo zbytečnou ztrátou času testerů.
- Počet nalezených chyb se rapidně zmenšuje a blíží se k očekávanému počtu nalezených chyb.
- Proběhly všechny testy.
- Ukázalo se, že každý prvek softwaru pracuje tak jak má, dle uvedených požadavků.
- Při prokázání, že systém funguje správně dle specifikace.
- Počet a závažnost chyb klesla na uspokojivou úroveň.
- Testování bylo úspěšně dokončeno a prošly všechny regresní testy, smoke testy a testy důvěry.

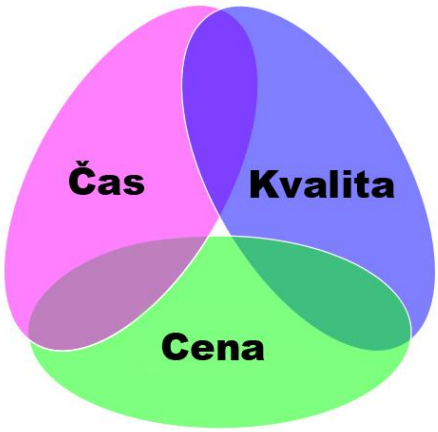
4. Řízení a dokumentace testování

Tato kapitola poukazuje na důležitost řízení testování a dále uvádí příklady dokumentace s náležitostmi, které jsou nutné dodržovat pro zajišťování kvality. Testovací dokumentace má významnou roli v procesu test reportingu.

Řízení je důležité i v oblasti této problematiky - testování. Je třeba, aby někdo plánoval procesy, řídil členy týmu, rozhodoval nad situacemi, rozdával pokyny, spravoval a motivoval vlastní tým, komunikoval ve všech směrech, jak s podřízenými, nadřízenými, tak i se zákazníky. Definice pojmu management je následující: *"Organizování a koordinování byznys aktivit, za účelem dosažení vymezených cílů"* [37].

Pro kvalitní řízení testování (Test Management) je třeba být dostatečně připraven a vše mít naplánované. Plán pro testování se odvíjí od zadaných požadavků zákazníka. Ze situací z reálného světa by pro zákazníky bylo nejvhodnější žádat produkt, který bude dodán co nejdříve, za nejnižší možnou cenu při té nejvyšší kvalitě [10]. Tento požadavek je nesplnitelný ze strany zhotovitele, protože je dokázaná vzájemná závislost mezi časem, kvalitou a cenou. Dle tzv. trojúhelníku "Vyber jakékoliv dva" (obr. 4) má zadavatel pouze tři možnosti výběru, jak požadovaný produkt bude vypadat [38]:

- *Produkt, který bude kvalitní a rychle vytvořen, cena v tomto případě bude vysoká.*
- *Produkt, který bude vytvořen rychle a za příznivou cenu, nebude příliš kvalitní.*
- *Vytvoření produktu, který bude levný a s vysokou kvalitou, bude trvat příliš dlouho.*



Obr. 4 - Trojúhelník "Vyber jakékoliv dva" (zdroj: [38], upraveno autorem)

4.1 Faktory úspěšného řízení

Řízení a proces testování není prováděno pouze jednou osobou, ale týmem. Pro dosažení těch nejlepších výsledků je třeba zajistit dostatečně kvalitní tým. Role, které se vyskytují v rámci týmu jsou popsány v minulých kapitolách. Je důležité si uvědomit, že cílem není individuální úspěch členů, ale dosažení týmových cílů. Tato kapitola naznačuje, podle čeho by vedení mělo sestavovat svůj tým a dále jak s ním pracovat.

Dle článku [39] pro vytvoření efektivně pracujícího týmu je třeba zohlednit tyto oblasti:

Znalosti podřízených

Především praktické znalosti o softwarovém testování jsou znalostmi, které se nedají jednorázově naučit. Vzdělávání se v této oblasti je stále probíhající proces, člověk se učí především praxí. Větší výhodu má tedy tester, který má zkušenosti s mnoha odlišnými projekty a nebyl zaměřen jen na určitou část. Takový tester má například znalosti různých softwarových systému a praktické zkušenosti s odlišnými fázemi testování. Při výběru testera, dáváme přednost tomu, který nemá problém s identifikací požadavků specifikace, vytvářením testovacích případů, analyzováním rizik a konáním testů. Naopak při náboru na manažerské pozice hledáme toho, kdo má zkušenosti s plánováním testování, řízením lidí, reportováním výsledků klientům a podobně [39].

Mezilidské vztahy

Velmi důležitou vlastností testerů a dalších podřízených je umět jednat s okolím a celkově vystupovat na veřejnosti. Člověk s těmito vlastnostmi nemá problém komunikace, pomoci nováčkům se zapadnutím do kolektivu či zaškolením. Testeři by měli mít mezi sebou kladné vztahy, aby dosahovali vysoké efektivity práce. Tyto vztahy je možné zlepšovat organizováním různých mítinků, porad nebo brainstormingů, kde se diskutuje o určité problematice a následně jsou účastníky vytvořeny nové nápady pro zlepšení [39].

Nezávislost testování

Na projektu by měly být i takové osoby, které jsou v rámci procesu testování nahraditelnými. Z důvodu toho, že člověk může své chyby opakovat je třeba, aby ho zastoupila jiná osoba, která bude na danou věc pohlížet ze svého pohledu. Příklady nezávislého testování jsou následující [39]:

- Testování vývojáři, kteří programovali kód
- Testování vývojáři, kteří daný kód neprogramovali
- Testovaní testery, kteří jsou součástí týmu
- Testování testery, kteří jsou z jiných týmů
- Testování vykonané externí skupinou testerů (outsourcing)

Motivace

Jednou z klíčových vlastností manažerů je vědět jak motivovat své podřízené, aby dosahovali co nejlepších výsledků. Kvalita odvedené práce nemotivovaného člověka se dá jen těžko porovnávat s kvalitou práce motivovaného člověka. Špatnou motivací testerů může být i například špatné řízení týmu manažery. Zvýšením motivace není pouze navýšení finančních odměn, ale je třeba i dle [10] provádět následující úkony: uznávat a chválit podřízené za správné odvedení práce, zjišťovat zpětnou vazbu, dávat prostor k debatě, možnosti sebezdokonalování a časové svobody. Pro zvýšení výkonu je možné vytvářet i soutěže mezi členy nebo vyhodnocování žebříčků zaměstnanců. Samozřejmostí jsou odměny za úspěchy v těchto soutěžích.

Komunikace

Základním pravidlem všech bodů je komunikace. Dle osobního názoru člověk, který má méně technických zkušeností, ale je výborný v komunikaci a projevu, má daleko vyšší šanci na uplatnění a zajímavé finanční ohodnocení než člověk, který je mistr ve svém technickém oboru, ale je ostýchavý a postrádá tyto schopnosti. Komunikace v oblasti softwarového testování je velice důležitá. Proces testování může být spojován s pouhým klikáním na počítači, ale opak je pravdou. Testeři často komunikují s vedením, mezi sebou a také s vývojovým týmem především ohledně nalezených chyb. Členové týmu by se měli vyvarovat vystavování konfliktů a raději problémy slovně vyřešit [39].

Utužování vztahů v týmu

Smyslem týmu není každodenní rutina v podobně příchodu do zaměstnání, komunikování ohledně pracovních záležitostí a včasný odchod. Mezi svými podřízenými je třeba vytvářet kolektiv. Pro vylepšování vztahů není na škodu organizovat tzv. team-buildingové akce, tedy akce, které jsou určeny pro ztužování kolektivu. Tyto akce jsou různého druhu, může se jednat o posezení v restauraci, pořádání večírků, ale i také o vícedenní události [39].

4.2 Dokumentace testování

Dokumentace je důležitou součástí testování. Je vytvářena před zahájením testování, během testování i po otestování. Slouží i pro reportování, tedy ohlašování stavu testování například vyššímu managementu, zákazníkovi a podobně. V této podkapitole jsou uvedeny některé základní typy dokumentace dle [13] a dále i náležitosti dokumentace.

4.2.1 Dokument Zásady testování

Dokument zásad testování [10] (Test Policy) je dokument, který se nachází na samotném vrchu struktury dokumentů. Základním cílem dokumentu je reprezentovat filozofii testování společnosti, popsat cíle testování a celkový přístup. Tento dokument by měl obsahovat následující náležitosti [40]:

- **Definici testování** - oznamuje proč se testuje, co se testuje a za použití jakých technik testování.
- **Popis procesu testování** - na kterou fázi je testování zaměřeno, jaké role jsou zahrnuty a podobně.
- **Vyhodnocení testování** - tento bod označuje jakým způsobem se bude testování vyhodnocovat a jaké metriky budou použity.
- **Stupeň kvality** - určuje jakou úroveň kvality bude výsledek testování mít.
- **Vylepšování testovacích procesů** - bod, který oznamuje jakými technikami budou jednotlivé prvky testování vylepšeny.

4.2.2 Dokument Strategie testování

Anglické označení je Test Strategy dokument [10]. Jedná se o dokument obvykle vytvořený projektovým manažerem. Značí se jako statický dokument, to znamená, že není často aktualizován, pravděpodobně z důvodu, aby ho bylo možné použít na více než jen jeden projekt. Určuje požadavky a způsoby testování. Obsahuje tyto složky [41]:

- *Možnosti a cíle*
- *Byznys záležitosti*
- *Role a jejich zodpovědnosti*
- *Komunikace a reporty stavu*
- *Předmět plnění*

- *Použité standardy*
- *Automatizované testy a nástroje*
- *Měření testování a použité metriky*
- *Rizika*
- *Identifikování a reportování defektů*
- *Uspořádání managementu*
- *Školící plán*

4.2.3 Dokument Plán testování

Plán testování (Test Plan) obvykle vytvořen vedoucím testování nebo manažerem testování. Oznamuje co se bude testovat, jakým způsobem, kdy se začne s testováním a kdo to bude provádět. Měl by obsahovat odpovědi na dotazované otázky. Měl by obsahovat následující náležitosti [10], [41]:

- *ID*
- *Úvod*
- *Cíle testování*
- *Jaké prvky se budou testovat*
- *Jaké prvky se testovat nebudou*
- *Techniky testování*
- *Úlohy testování*
- *Kritéria pro případné pozastavení*
- *Artefakty testování*
- *Zdroje testování*
- *Zodpovědnosti lidí*
- *Časový rozpis*

4.2.4 Dokument Reportování chyb

Dokument reportování chyb (Bug Report) je dokument, do kterého testeři zapisují veškeré informace o nalezeném bugu. Snaží se ho zde co nejlépe popsat, aby došlo k úplnému pochopení situace ve vývojovém týmu. Nejdůležitější body jsou [42]:

- **ID** - číslo bugu, které je obvykle generováno počítačem.

- **Shrnutí** (Summary) - je myšleno ve smyslu velmi stručného názvu. Testeři se zde snaží stručně popsat část, o kterou se jedná.
- **Popis** - jednoduše a přehledně popisuje jak došlo k bugu a co se stalo.
- **Závažnost** - tester určuje podle svého uvážení. Převážně se rozděluje na skupiny - cirical, high, medium, low (*kritická, vysoká, střední, nízká*).
- **Priorita** - třídí nalezené chyby dle priority k opravě. Podobné označení jako u závažnosti - high, medium, low. Pokud tester odešle chybu vývojovému týmu s vyšší prioritou, bude daleko rychleji opravena než s prioritou nižší.
- **Datum** - často generováno automaticky.
- **Verze** - určení v jaké verzi byl defekt nalezen.
- **Reportováno** - většinou se jedná o username testera. Vhodné například při reportování výsledků, kolik, jaký tester nalezl bugů k určitému datu.
- **Přílohy** - v případě, kdy není dostatečně vysvětleno nalezení bugu nebo je jednodušší znázornit chybu například screenshotem.

4.2.5 Dokument Shrnutí testování

Dokument shrnutí testování (Test Summary), kterým se reportuje například vyššímu managementu průběh dokončeného testování. Jsou zde vypsány všechny zahrnuté osoby, obvykle ve spodní části je prostor pro podpisy osob a data. Shrnutí by mělo mít tyto části [13]:

- **Popis** - obsahuje shrnutí předmětu testování, verzi produktu, opakování testů (do jakých kol došly), prostředí, ve kterém se testovalo a podobně.
- **Odchylky** - poukazuje na všechny nalezené rozdíly od specifikace, testovacího plánu, testovacích procedur a udává příčiny vzniku.
- **Posudek komplexnosti** - vyhodnocuje komplexnost proběhnutých testovacích procesů komplexními kritérii, které jsou stanoveny v testovacím plánu. Zaznamenává prvky či části produktu, které nebyly otestování a udává důvod proč tomu tak je.
- **Shrnutí výsledků** - zahrnuje všechny dříve nalezené a opravené chyby a způsob, jakým byly opraveny. Obsahuje seznam zbývajících neopravených chyb.
- **Vyhodnocení** - obsahuje všechny testy a jejich výsledky, zda proběhly úspěšně či neúspěšně (passed / failed).

- **Shrnutí technik a metod** - je celkové shrnutí všech technik a metod, které byly použity během testování, dále obsahuje například data, která udávají dobu trvání jednotlivých testů.

4.2.6 Denní / týdenní report testování

Ze strany vedení může být po testovacím týmu vyžadován pravidelný tzv. report testování a to buď na denní nebo týdenní bázi. Reporting testování je pojem, který by se dal definovat jako činnost, pomocí které je možné zjistit v jaké fázi a v jakém aktuálním stavu se testování softwaru nachází. Vytvářet reporty testování mohou různé osoby, často testovací tým poskytuje reporty vyššímu managementu (testovacímu manažerovi, projektovému manažerovi), který zjistí jaký je aktuální stav testování. Na základě obdržených reportů je možné provést analýzu, vyhodnotit chyby a pokusit se zvýšit efektivitu testování pro následující činnost. Pro vytvoření kvalitních reportů je třeba použít vhodné metriky, což jsou kritéria podle kterých je vyhodnocován například proces testování.

Tento dokument denního či týdenního reportu testování by měl zahrnovat informace o nalezených defektech a o testech, které byly v daném časovém úseku vykonány. Konkrétní obsah reportu se dělí dle internetového zdroje [43] na deset bodů:

- *Počet plánovaných testovacích případů pro daný den*
- *Počet vykonaných testovacích případů za daný den*
- *Celkový počet dokončených testovacích případů*
- *Počet zjištěných defektů pro daný den*
- *Celkový počet zjištěních defektů*
- *Počet stále otevřených kritických defektů*
- *Doba nečinnosti pracovního prostředí - v případě odstávky*
- *"Trháky", který vyvolají ohlasy - v případě, že nějaké nastaly*
- *Odkaz na umístění zhotovených testů*
- *Odkaz na umístění defektů*

5. Test Reporting

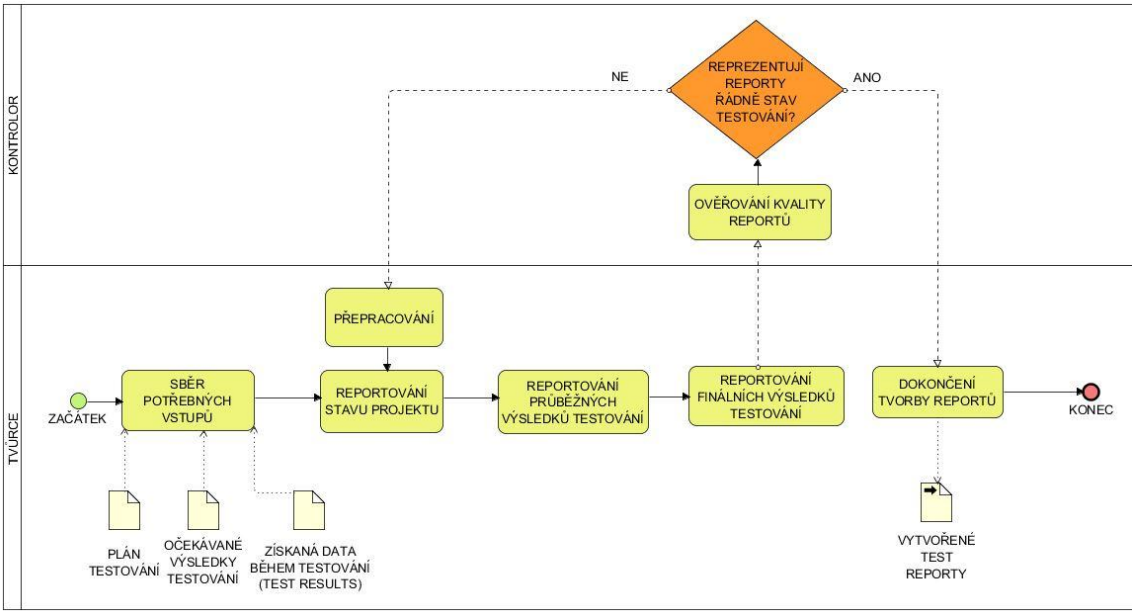
Pátá část bakalářské práce je zaměřena na teorii test reportingu, který je hlavním cílem práce a dále uvádí metriky pro měření kvality. Jsou popsány některé používané metriky v praxi, které jsou podloženy vzorci pro výpočet. Toto téma je úzce spojeno s testovací dokumentací, která se řadí mezi vstupy do procesu reportování. Náležitosti dokumentace jsou uvedeny v kapitole 4.2.

Chybně prováděné testování softwaru souvisí s časovou náročností a vysokými finančními náklady. Proto je vhodné neustále zjišťovat jaký je aktuální stav procesu a zda celá činnost testování běží pole plánu. Sdělení těchto informací nestačí pouze oznámit zainteresované osobě, zda je či není vše v pořádku, ale je třeba mít tyto informace podložené daty. Pojem Test Reporting je aktivita, během které se vytváří reporty z testování. Na základě analýzy reportů, lze stanovit jakým směrem se bude testování dále ubírat. Reporty nemusí obsahovat pouze technické informace ohledně vypsaných defektů a podobně, ale například reporty prováděné v rámci testovacího týmu mohou managementu sdělovat informace o výkonnosti testerů.

Obecným výsledkem reportování je dokument zvaný "Test Report" (report z testování), jehož anglická definice dle článku [44] je následující: *"Document that records data obtained from an experiment of evaluation in an organized manner, describes the environmental or operating conditions, and shows the comparsion of test results with test objectives."* Ve volném překladu do českého jazyka se jedná o dokument, k jehož sestavení jsou potřeba reálná data získaná během procesu testování. Tento dokument dále popisuje jaký je stav prostředí za určitých podmínek a dále dává možnost porovnaní reálných výsledků testování s původními stanovenými cíli.

5.1 Vstupy, úlohy a výstupy procesu reportování

Je třeba říci, že u mnoha projektů je testování nevhodně reportováno. To je zapříčiněno například tím, že se reportuje pouze v jedné fázi procesu testování a vytvářejí se pouze základní typy reportů, jako jsou: sumární stav testů a sumární stav defektů [45]. Nutností je si uvědomit, že reportování má důležitý význam v různých etapách testování. Je tedy vhodné jejich tvorbu správně načasovat. Úlohy tohoto procesu jsou velice přehledně zpracovány v jedné z kapitol knihy [14]. Proces reportování je zachycen na obrázku č. 5.



Obr. 5 - Proces Test Reporting (zdroj: [14], upraveno autorem)

Obrázek č. 5 je upraven autorem v notaci BPMN 2.0, která slouží pro modelování podnikových procesů pomocí diagramů. Proces je zahájen zeleným kolečkem, které značí startovní událost "Začátek". Následují aktivity, které jsou značeny obdélníky se zaoblenými rohy. Do první aktivity vstupuje několik objektů, které ovlivňují celý proces a jsou značeny symbolem papíru. Rozhodování je v modelu znázorněno rozhodovací bránou ve tvaru kosočtverce. Je třeba si všimnout rozdělení diagramu tzv. horizontálními bazénky, které určují účastníky procesu. Výstup celého procesu je zde na obrázku značen symbolem papíru s vyplněnou šipkou. Červené kolečko značí ukončovací událost celého procesu.

5.1.1 Vstupní objekty

Za základní vstupy do procesu reportování považujeme následující 3 typy:

- **Plán testování**

Jedná se o plán testován, který je součástí testovací dokumentace pro zajišťování kvality produktu. Náležitosti plánu testování jsou zmíněny v kapitole 4.2.3.

- **Očekávané výsledky testování**

Pro srovnání testeři reportují opravdové výsledky testování s očekávanými výsledky. Je tedy třeba, aby testeři věděli jaké jsou očekávané výsledky. V testování softwaru se převážně jedná o požadavky byznys osob.

- **Získaná data během testování**

Tato získaná data jsou výsledky (Test Results), se kterými se testeři setkávají během testování. Dále se jedná o vytvořené testovací sady a další důležité prvky k řádnému testování systému. V neposlední řadě velkou roli mají vzniklé defekty, které se nahlašují týmu vývojářů. Náležitosti správně zadaného defektu je možno vidět v kapitole 4.2.4.

5.1.2 Fáze tvorby

1) Reportování stavu projektu

V rámci této fáze dochází k vytváření reportů, které umožňují IT manažerům jednoduše zjistit stav projektu. Při tvorbě těchto dokumentů je doporučováno zpracovávat informace přehledně a dále je vhodné používat i grafické zabarvení zajímavých faktů, které umožní zaneprázdněnému manažerovi načerpat informace co nejrychleji a nejefektivněji. Mezi reporty stavu projektu se řadí:

- **Summary Status Report** (Report shrnutí stavu)

Tento report poskytuje obecné informace o projektech. Měl by obsahovat náležitosti, jako je datum vytvoření reportu, informace o projektu, např. název projektu, jméno manažera, fáze projektu a kdo projekt financuje. Obsahovat by měl technické informace a byznys požadavky, časový rozvrh prací projektu a informace o rozdělení rozpočtu. Pro jednodušší orientaci na dokumentu je vhodné použít legendu.

- **Project Status Report** (Report stavu projektu)

Oproti předchozímu reportu je tento zaměřen více do hloubky jednoho konkrétního projektu. Je tedy podrobnější, ale stále by měl být psán za dostatečné přehlednosti.

2) Reportování průběžných výsledků

Této části reportů je důležité se věnovat z toho důvodu, že zobrazují aktuální stavy testování a mohou být použity i pro závěrečné reporty. Během dobré analýzy z těchto reportů je možné ušetřit čas a zlepšit techniku testování. Pravidelnost vytváření je závislá na vlastním uvážení

členů testovacího týmu a na rozsáhlosti testovacího procesu. Konkrétní praktické příklady budou uvedeny v kapitole o metrikách.

3) Reportování finálních výsledků

Jedná se o finální dokument, který obsahuje důležité informace z jednotlivých testovacích aktivit. Během tohoto procesu by mělo dojít k shrnutí informací z reportů jako například jsou:

- **Report samotného projektu**
- **Report integračního testování**
- **Report systémového testování**
- **Report akceptačního testování**

Správně zpracovanému reportu finálních výsledků předchází kvalitně zpracovaný plán testování, jenž byl zmíněn v předchozích kapitolách.

5.1.2 Kontrola procesu

Při kontrole procesu správného reportování je vhodné vytvořit dotazník, který se dělí dle fáze tvorby reportů a obsahuje několik otázek, na které je třeba kladně odpovědět. Mezi tyto otázky například patří:

- *Získali tvůrci reportů očekávané výsledky testování?*
- *Je zde metoda reportování defektů zahrnuta?*
- *Konzultovali testeři s manažery, jaké reporty jsou vyžadovány?*
- *Obdržely zainteresované osoby požadované reporty?*
- *Zahrnují reporty fakta, co funguje správně a naopak?*

V případě kladných výsledků kontrolního seznamu či dotazníku jsou výsledkem procesu jednotlivé reporty. V opačné situaci je třeba celý proces od první fáze tvorby reportů opakovat a zlepšit.

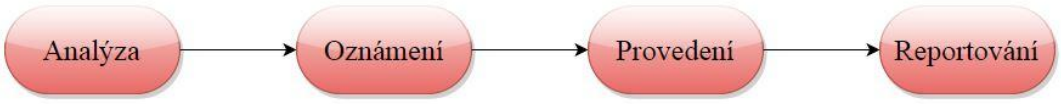
5.1.3 Výstupy

Za výstupy celého procesu považujeme tzv. Test Reporty, které jsou získané ze všech zmíněných fází reportování.

5.2 Využití metrik

Metriky jsou nedílnou součástí a základním prvkem pro vytváření reportů o průběžných výsledcích testování softwaru. Pojem metrika je definován následovně: *"Kritérium měření kvality produktu, podle kterého posuzujeme jeho efektivitu, výkonnost, vývoj, kvalitu plánu a průběh."* [46].

Jak je možné si všimnout, životní cyklus metrik (obr. 6) se dělí do čtyř základních fází. Prvním bodem cyklu je analýza, kde je třeba určit metriky, které budou použity a následně je definovat. Druhou fází je oznámení. Je nezbytné určit komu výsledky měření budou sděleny a vysvětlit osobám, proč dané metriky jsou potřebné pro výsledky měření. K této fázi se vztahuje ještě proces obeznámení testovacího týmu o měření kvality a zaúkolování týmu, aby proběhl sběr potřebných dat pro vykonání tohoto procesu. Dalším bodem je zachycení a ověření sebraných dat, zda to jsou opravdu ta, která hledáme. V rámci fáze provedení je zahrnuta i samotná práce s daty, kde dochází k výpočtů měření. Poslední fází je reportování. Jedná se o vytvoření konečného reportu, který bude srozumitelný pro všechny zainteresované strany, bude distribuován mezi tyto strany s určitým závěrem a obdržením zpětné vazby.



Obr. 6 - Životní cyklus metrik (zdroj: [47], upraveno autorem)

Velké množství vstupů pro zpracování reportů tvoří metriky s nalezenými defekty a testy. S těmito daty je možné vytvářet různé reporty dle vlastního uvážení. Mnoho reportů lze vytvořit v nástrojích pro podporu testování, avšak je dobré vědět, jaké výpočty vytvoření reportu předcházejí. V této kapitole jsou zobrazeny příklady některých z nich.

5.2.1 Hustota defektů

Metrikou tzv. hustotou defektů [13] zjistíme procentuální počet výskytu defektů ve zdrojovém kódu. K výpočtu je potřeba vědět celkový počet defektů a hodnotu KLOC (počet tisíců řádků nezakomentovaného zdrojového kódu [48]).

$$hustota\ defektů\ = \frac{celkový\ počet\ nalezených\ defektů}{KLOC}$$

5.2.2 Závažnost nalezených defektů

Tato metrika nám vypočítá procentuální počet nalezených defektů dle závažnosti. Díky této metodě je možné určit, kolik procent defektů je dobré vyřešit co nejrychleji kvůli větší závažnosti a podobně. Výstup z této metriky vypadá například jako obrázek č. 7. Tato metrika je možná i pro použití měření defektů dle priorit.

$$\%\ počet\ kritických\ defektů\ = \frac{celkový\ počet\ kritických\ defektů}{celkový\ počet\ defektů} \times 100$$

$$\%\ počet\ defektů\ vyšší\ závažnosti\ = \frac{celkový\ počet\ defektů\ vyšší\ závažnosti}{celkový\ počet\ defektů} \times 100$$

$$\%\ počet\ defektů\ střední\ závažnosti\ = \frac{celkový\ počet\ defektů\ střední\ záva\check{z}.}{celkový\ počet\ defektů} \times 100$$

$$\%\ počet\ defektů\ nízké\ závažnosti\ = \frac{celkový\ počet\ defektů\ nízké\ závažnosti}{celkový\ počet\ defektů} \times 100$$

Obr. 7 - Graf závažnosti nalezených defektů (zdroj: autor)

5.2.3 Efektivita nalézání defektů

Metrika efektivity nalézání defektů (angl. DRE - Defect Removal Efficiency) je velmi zajímavou metrikou pomocí které se zjistí procentuální efektivita nalézání defektů během procesu testování. Je třeba do vzorce pro procentuální výpočet dosadit za čitatel celkový počet nalezených defektů a za jmenovatel součet celkového počtu nalezených defektů a celkového

počtu nalezených defektů koncovým uživatelem. Tito uživatelé mohou nalézt defekty například když je software v provozu jako alfa či beta [47].

$$[\%]\ DRE = \frac{počet\ nalezených\ defektů\ v\ rámci\ testování}{počet\ nalezených\ defektů\ v\ rámci\ testování + počet\ nalez.defektů\ koncovým\ uživatelem}$$

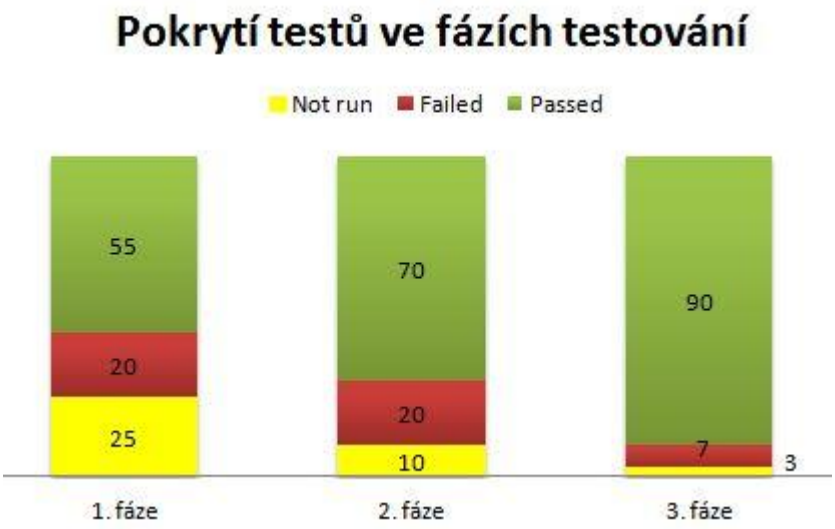
5.2.4 Pokrytí testů ve fázích testování

Dalším bodem je pokrytí testů kde metrika zobrazuje jaké jsou procentuální výsledky proběhnutých testů v jednotlivých fázích testování. Jedná se o hodnoty - úspěšně, neúspěšně, test neproběhl (passed, failed, not run). Umožňuje snadno a rychle vystihnout stav testování, avšak je vhodné tuto metriku doplnit údajem o počtu testů. Provedení reportu by mohlo vypadat následovně (obr. 8). K výpočtu je třeba vydělit výsledky testů celkovým počtem testů.

$$\%\ počet\ úspešných\ testů\ = \frac{celkový\ počet\ úspěšných\ testů}{celkový\ počet\ testů} \times 100$$

$$\%\ počet\ neúspěšných\ testů\ = \frac{celkový\ počet\ neúspěšných\ testů}{celkový\ počet\ testů} \times 100$$

$$\%\ počet\ neproběhnutých\ testů\ = \frac{celkový\ počet\ neproběhnutých\ testů}{celkový\ počet\ testů} \times 100$$



Obr. 8 - Graf pokrytí testů ve fázích testování (zdroj: autor)

5.2.5 Rychlost nalezení defektu

Následující metrika měří rychlost nalezení defektu. Pro příklad (obr. 9) je nutné nejprve vypočítat počet nalezených defektů za hodinu a poté porovnat v jednotlivých dnech testování. Lze zjistit spoustu informací jako například výkonnost testovacího týmu [10].

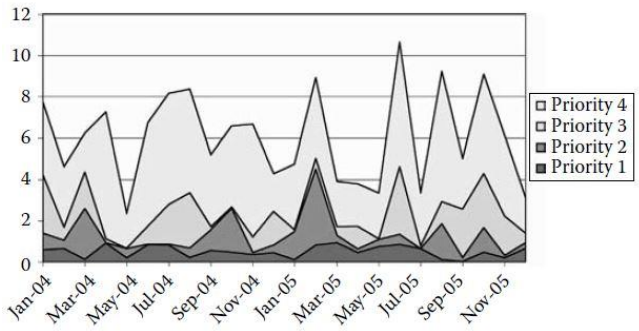
$$počet\ defektů\ za\ hodinu = \frac{počet\ nalezených\ defektů}{počet\ hodin\ strávených\ nalézáním}$$

Obr. 9 - Graf počtu defektů ve dnech testování (zdroj: [10], upraveno autorem)

5.2.6 Průměrná doba otevřeného bugu

Zajímavou metrikou měření kvality je i výpočet průměrné doby otevření bugu. Většinou je tato metrika počítána na konci každého měsíce a je často doplněna rozdělením podle závažnosti či priority bugu. Pro výpočet potřebujeme odečíst od současného času čas otevření bugu, a to pro každý bug a následně vydělit počet všech bugů, které byly otevřeny. Praktickou ukázku je možné vidět na obrázku č. 10, kde si lze povšimnout, že například bugy priority 1 byly vyřešeny během cca 3 dní. Svislá osa je udávána v týdnech [13].

$$průměrná\ doba\ otevření\ bugu = \frac{\Sigma\ (\text{čas současný} - \text{doba otevření bugu})}{celkový\ počet\ otevřených\ bugů}$$



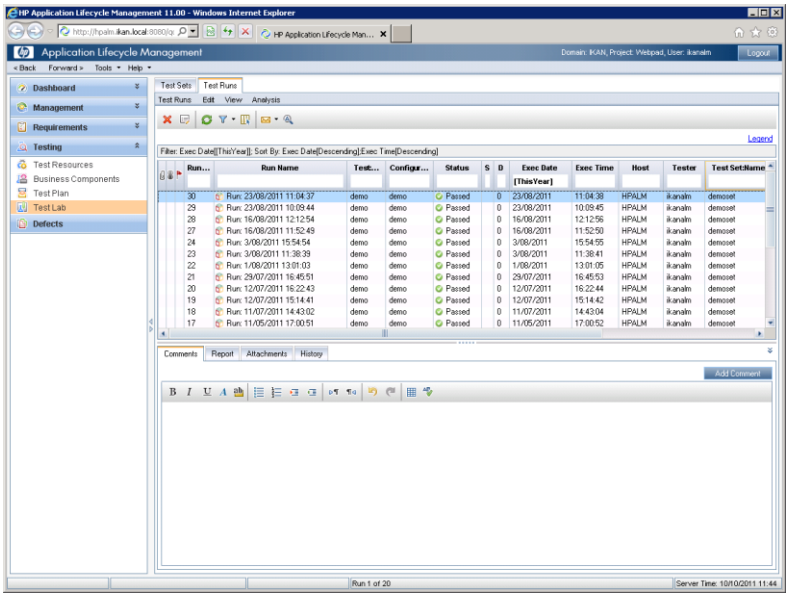
Obr. 10 - Příklad grafu průměrné doby otevření bugů (zdroj: [13])

6. Tvorba reportů v nástroji HP ALM

Tato kapitola představuje praktickou část bakalářské práce a má za úkol poskytnout cenné informace uživatelům, kteří mají alespoň elementární znalosti s používáním softwaru HP Application Lifecycle Management, o tom, jak vytvářet účelné reporty testování. Dále kapitola obsahuje tutoriál vytvoření nástěnky z reportů. Nástroj HP ALM byl vybrán z důvodu praktických zkušeností autora s tímto softwarem.

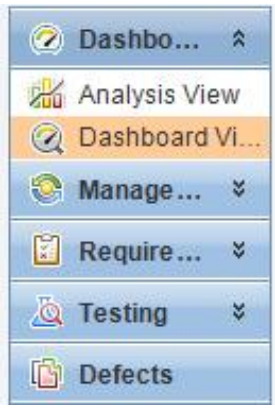
6.1 Charakteristika nástroje HP ALM

HP Application Lifecycle Management, zkráceně HP ALM, je rozšířenou verzí jedné z jeho složek - HP Quality Center. Tento produkt byl vytvořen softwarovou divizí americké společnosti Hewlett-Packard. Nástroj podporuje všechny fáze testovacího procesu, mezi které se například řadí: definování releasů, hlubší specifikování požadavků, vytváření testovacích případů a jejich následné spouštění, zaznamenávání chyb, reportování stavu testování a podobně. Jedná se o webovou aplikaci, protože pro připojení je třeba použití webového prohlížeče, kterým je nejčastěji Internet Explorer. Umožňuje komunikaci mezi mnoha skupinami uživatelů, jakými jsou například: vývojáři, byznys analytici, testovací tým [49], [50]. Jak vypadá pracovní prostředí produktu HP ALM je možno vidět na obrázku č. 11 [51].



Obr. 11 - Pracovní prostředí HP ALM (zdroj: [51])

V pracovním prostředí je možno vlevo vidět panel modulů HP ALM (obr. 12), mezi které se řadí: Dashboard, Management, Requirements, Testing a Defects. Tyto moduly jsou níže stručně popsány a rozděleny [49].



Obr. 12 - Moduly HP ALM (zdroj: autor)

6.1.1 Dashboard

- **Analysis View** - umožňuje vytvářet různé typy grafů a MS Excel reporty.
- **Dashboard View** - díky této sekci je možné vytvořit si přehlednou nástěnku s jednotlivými grafy.

6.1.2 Management

- **Releases** - definování a nastavení release a časových cyklů.
- **Libraries** - knihovny zaznamenávající změny v projektu, používání entit mezi projekty.

6.1.3 Requirements

- **Requirements** - sekce pro správu požadavků. Požadavky mohou být spojeny s ostatními požadavky, testy nebo defekty.
- **Business Models** - umožňuje naimportovat procesní byznys modely, testovat kvalitu modelů a komponent. Tato sekce je omezená dle ALM licence.

6.1.4 Testing

- **Test Resources** - sekce pro správu zdrojů pro testování.
- **Business Components** - testování byznys procesů (omezení licencí).

- **Test Plan** - umožňuje vytvářet testovací případy. Tyto případy mohou být dále propojeny s jednotlivými požadavky a defekty.
- **Test Lab** - řízení a spouštění testů. Po otestování je možné analyzovat výsledky.

6.1.5 Defects

Správa defektů od jejich vystavení, udávání priorit, opravu defektů až po jejich uzavření.

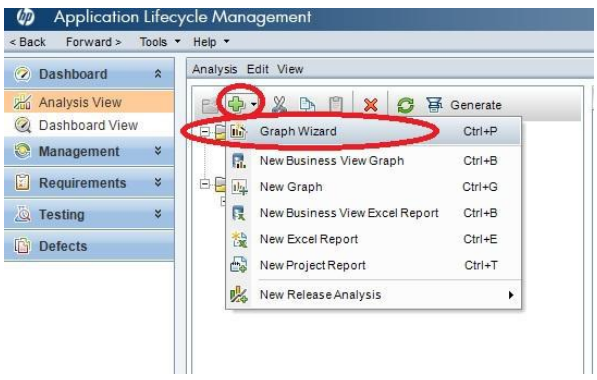
6.2 Tutoriál vytváření reportů a nástěnky

Tato podkapitola se věnuje samotnému vytváření reportů a nástěnky v aplikaci HP ALM. Tutoriály jak vytvářet reporty a nástěnky mohou posloužit nejen manažerovi testování, ale například i jednotlivým testerům.

6.2.1 Tutoriál vytváření reportů

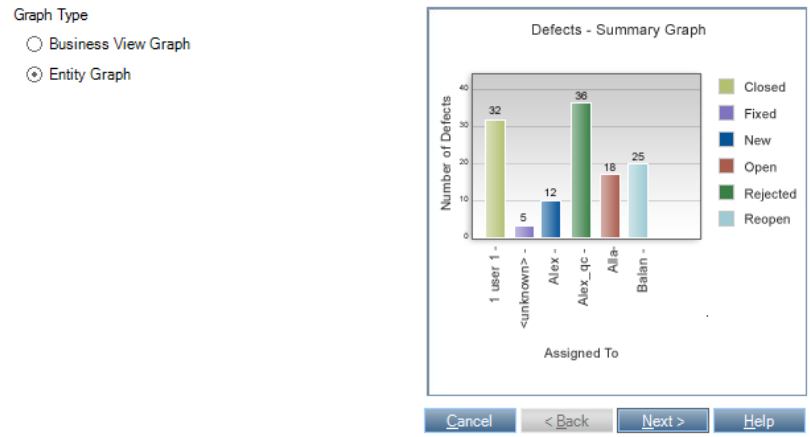
Tento tutoriál popisuje jednotlivé kroky jak vytvářet reporty v HP ALM. Seznam konkrétních reportů a reportu vygenerovaného v tomto tutoriálu je obsahem podkapitoly 6.3 i s konfiguracemi, které jsou třeba definovat do aplikace HP ALM.

Nejprve je třeba v levém panelu modulů kliknout na sekci **Dashboard**, následně se rozbalí dvě možnosti (**Analysis View** a **Dashboard View**). Pro vytváření reportů je třeba kliknout na pole **Analysis View**. Po kliknutí na tuto sekci dochází k načtení složek **Private** a **Public**. Pokud manažer testování chce umožnit testerům veřejně sledovat reporty, je třeba tyto reporty vytvářet ve složce **Public**. V opačném případě vytváření reportů ve složce **Private** umožňuje pouze osobní přístup k nahlížení obsahu. Vytvoření nového reportu se zahájí kliknutím na zelené tlačítko znaménka plus (obr. 13), následně se zobrazí seznam možností. Velkou výhodou je vytváření reportů pomocí průvodce **Graph Wizard**.



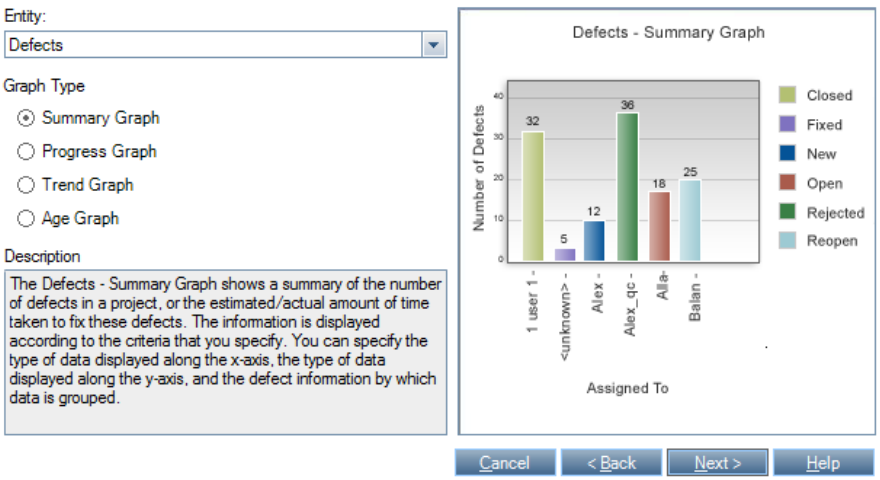
Obr. 13 - Nový report (zdroj: autor)

Prvním krokem průvodce vytváření reportů je výběr, zda se jedná o byznys graf **Business View Graph** či o graf entit (obr. 14). Pro následující kroky bude vybrán graf s entitami **Entity Graph**.



Obr. 14 - Průvodce vytvářením reportů 1 (zdroj: autor)

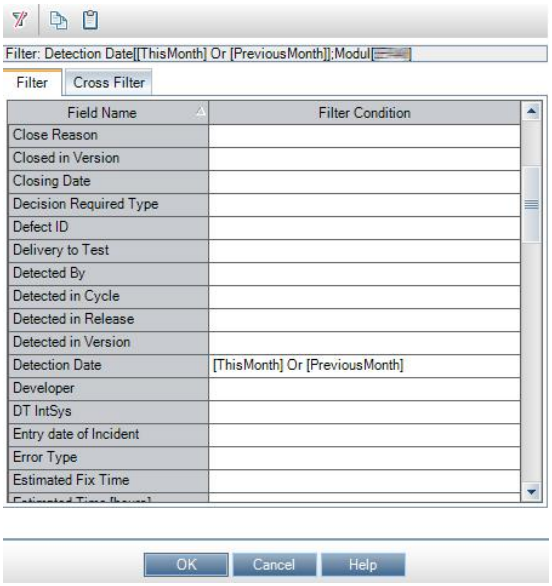
Druhá obrazovka uživateli nabízí možnost volby entity (obr. 15) - uvádí co bude podstatou výsledného grafu či na co se uživatel zaměřuje. Následně je třeba vybrat typ grafu (Summary, Progress, Trend a Age). Je možné si všimnout, že při zaškrtávání různých typů se vpravo mění náhled pro představu, jak bude report vypadat.



Obr. 15 - Průvodce vytvářením reportů 2 (zdroj: autor)

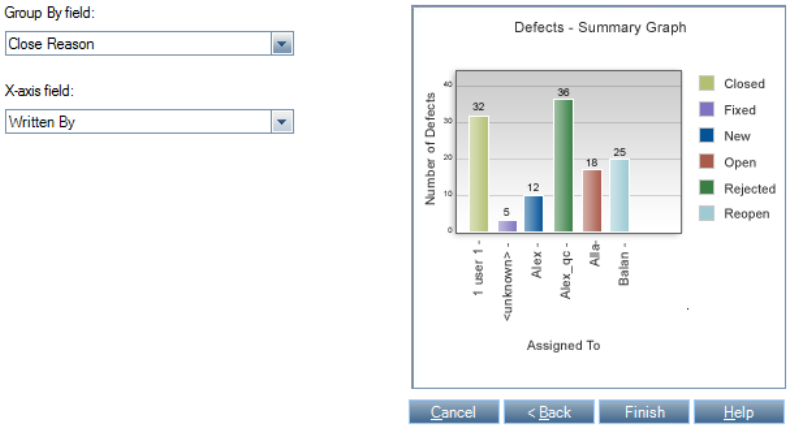
Na třetí obrazovce uživatel vybírá ALM projekt, ze kterého se budou čerpat data pro vytvoření reportu. Při volbě zanechání současného projektu je třeba zvolit **Use Current Project**. V opačném případě **Use Selected Projects** a vybrat ze seznamu projekt.

Jednou z nejpodstatnějších částí při vytváření reportů je následná čtvrtá obrazovka, kde uživatel definuje filtry reportu **Define a new filter**. Po kliknutí na definování filtrů se zobrazí tabulka s mnoha možnostmi filtrování, jako např. filtry dle času uzavření defektu, dle uživatelů, kteří defekt vystavili, dle data nalezení / provedení apod. (obr. 16).



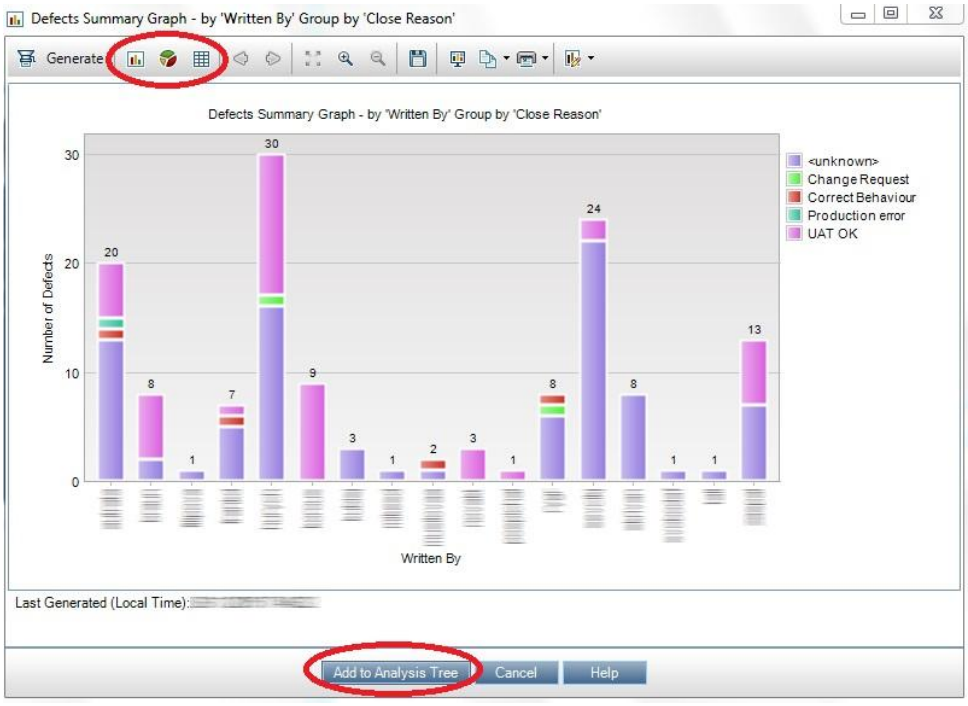
Obr. 16 - Průvodce vytvářením reportů 3 (zdroj: autor)

Posledním krokem průvodce je stav, kdy uživatel nastaví co se bude na osách x či y zobrazovat. Dále je třeba rozhodnout dle čeho se budou data seskupovat - pole **Group By field**. Závěrečný krok nabízí uživateli opět spoustu možností. Obr. 17 zobrazuje jak vypadá poslední krok průvodce.

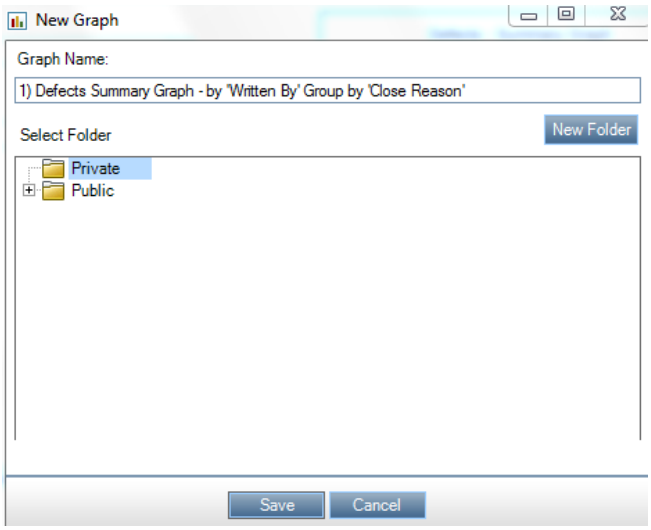


Obr. 17 - Průvodce vytvářením reportů 4 (zdroj: autor)

Průvodce vytvářením reportů je třeba dokončit stisknutím tlačítka **Finish**. Po stisknutí dochází k načtení obrazovky, která vygenerovala uživateli nadefinovaný report (obr. 18). V horní liště na obrázku jsou znázorněny tři ikony, které určují jaké bude mít vygenerovaný report zobrazení. Uživatel může zvolit sloupcový graf, výsečový graf či tabulku (Bar Chart, Pie Chart, Data Grid). Po vybrání toho nejsmysluplnějšího zobrazení je možno vygenerovaný report přidat do složek tlačítkem **Add to Analysis Tree**. Následuje načtení obrazovky s dotazem kam daný report uložit a také jak ho pojmenovat (obr. 19). Reporty se po uložení nacházejí pod modulem **Dashboard** v sekci **Analysis View**.



Obr. 18 - Vygenerovaný report (zdroj: autor)

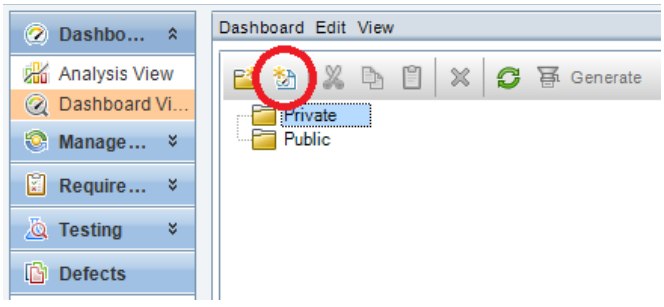


Obr. 19 - Uložení reportu (zdroj: autor)

6.2.2 Tutoriál vytváření nástěnky

V modulu **Dashboard** se dále nachází sekce zvaná **Dashboard View**, která slouží pro vytváření přehledných nástěnek. Na tyto nástěnky si uživatel může umístit předem definované reporty. Nástěnku je možno pojmenovat např. týdenní nástěnka, kde budou jednotlivé reporty zachycovat data pouze ke konkrétnímu týdnu. Ukládání nástěnek je stejné jako pro sekci **Analysis View**. Nachází se zde dvě složky **Private** a **Public**.

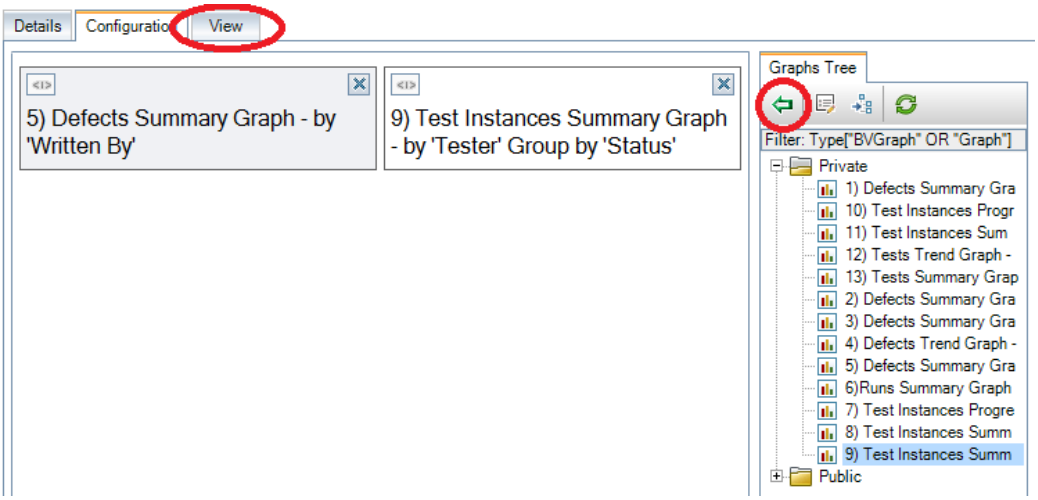
Vytvoření takové nástěnky se provede kliknutím na ikonu (obr. 20) v příslušné složce. Následuje zobrazení okna **New Dashboard Page**, ve kterém je třeba pojmenovat nástěnku a potvrdit tlačítkem **OK**.



Obr. 20 - Vytvoření nástěnky (zdroj: autor)

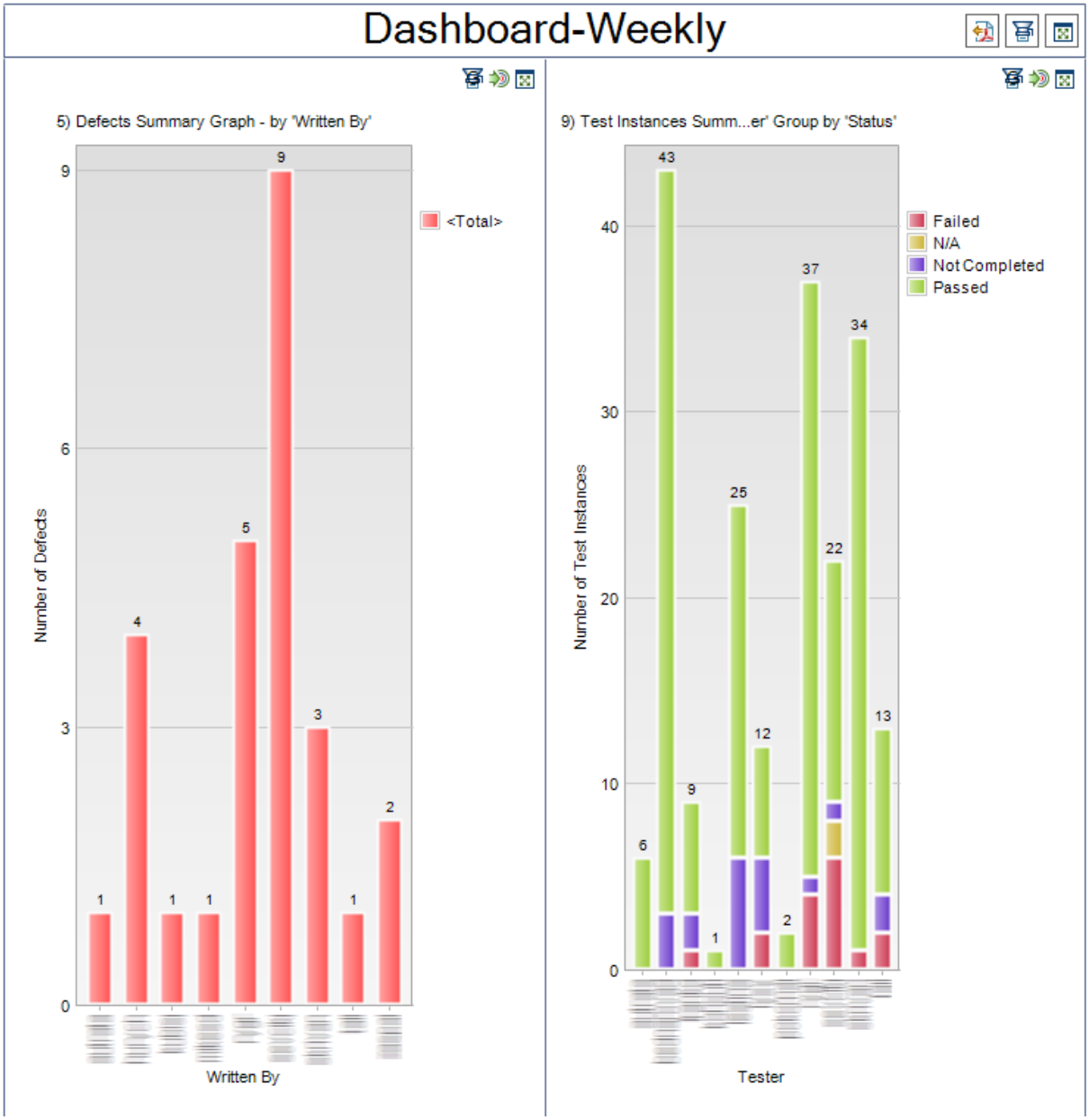
V dalším kroku je třeba rozhodnout, které reporty budou v nástěnce obsaženy. Například pro vytváření nástěnky s týdenními informacemi je nutno, aby použité reporty měly správně nastavenou konfiguraci. Pro vložení více reportů na nástěnku se reporty vybírají ze sekce

Analysis View a potvrzují ikonou zelené šipky (obr. 21). Nyní je nástěnka nakonfigurovaná a uživatel si ji může zobrazit kliknutím na tlačítko **View**. Příklad vygenerované nástěnky je možno vidět na obrázku č. 22.



Obr. 21 - Konfigurace nástěnky (zdroj: autor)

Na ukázkové nástěnce se nachází dva reporty. Report vlevo znázorňuje počet vystavených defektů jednotlivými testery v rámci jednoho týdne. Druhý report zobrazuje počet otestovaných kroků testery v daném týdnu.



Obr. 22 - Příklad vytvořené nástěnky (zdroj: autor)

6.3 Příklady reportů s konfiguracemi

V této části se nachází účelné reporty známé z praxe, ze kterých mohou manažeři čerpat mnoho informací. U každého reportu je uvedena konfigurace daného reportu, kterou uživatel zadává do průvodce vytvářením reportu. Je tedy možné dle níže uvedených konfigurací vytvořit reporty pro různé projekty. Všechny zmíněné reporty jsou sestaveny na základě reálných dat, proto v rámci zachování důvěrnosti informací byly klíčové údaje (např. jméno testera) upraveny tak, aby byly nečitelné.

Jednotlivé reporty je vhodné rozřadit do sekcí např. dle HP ALM modulů, aby bylo zřejmé čeho se daný report týká. Mezi sekce uvedené v této kapitole patří:

- **Defects** - reporty týkající se defektů
- **Test Lab** - reporty spuštěných testovacích sad, jednotlivé kroky testů apod.
- **Test Plan** - reporty spojené s vytvářením testů

Seznam uvedených příkladů dle sekcí:

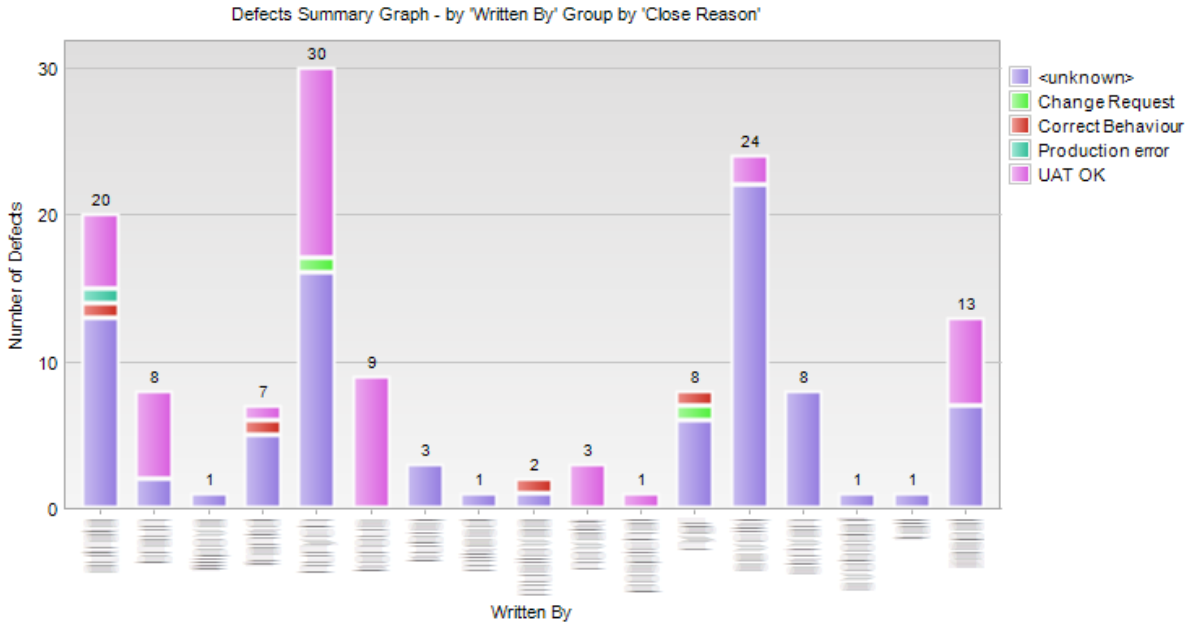
1) **Defects**
    - *Report důvodu uzavření vypsaných defektů dle testerů*
    - *Report stavu defektů dle závažnosti*
    - *Report závažnosti defektů dle testovaných modulů*
    - *Report stavu defektů dle časového úseku*
2) **Test Lab**
    - *Report stavu otestovaných sad dle testerů*
    - *Report stavu všech testovaných kroků*
    - *Report vývoje otestovaných kroků dle testerů v časovém úseku*
3) **Test Plan**
    - *Report psaní testů testery v časovém úseku*
    - *Report stavu psaných testů testery*

6.3.1 Report důvodu uzavření založených defektů dle testerů

Report (obr. 23) udává jaké jsou důvody uzavírání defektů založených jednotlivými testery v definovaném časovém úseku. Je možné vidět kolik defektů tester, který je uvedený na ose x, založil. Tento typ reportu dále ukazuje, zda testeři zakládají kvalitní defekty. Defekty, které jsou založeny špatně a chování je správné, mají označení **Correct Behaviour**. Konfigurace (nastavení) reportu je uvedena v tabulce č. 2.

Tabulka 2 - Konfigurace reportu 1 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Defects
Graph Type	Summary Graph
Filter	Detection Date; Modul
Group By field	Close Reason
X-axis field	Written By



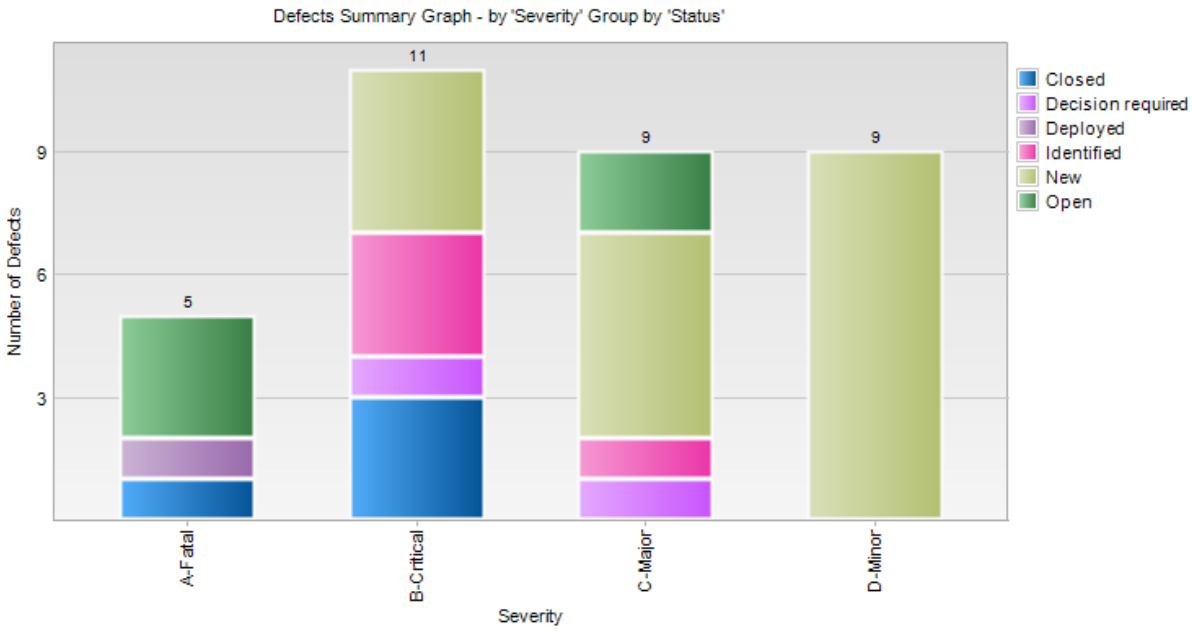
Obr. 20 - Report důvodu uzavření založených defektů dle testerů (zdroj: autor)

6.3.2 Report stavu defektů dle závažnosti

Tento report (obr. 24) na ose x uvádí závažnost založených defektů a na ose y počet defektů. Díky tomuto reportu je možno vidět jaký je stav defektů dle závažnosti. Je vhodné mít defekty vyšší závažnosti dříve opraveny a uzavřeny než defekty závažnosti nižší. Konfigurace reportu je uvedena v tabulce č. 3.

Tabulka 3 - Konfigurace reportu 2 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Defects
Graph Type	Summary Graph
Filter	Detection Date
Group By field	Status
X-axis field	Severity



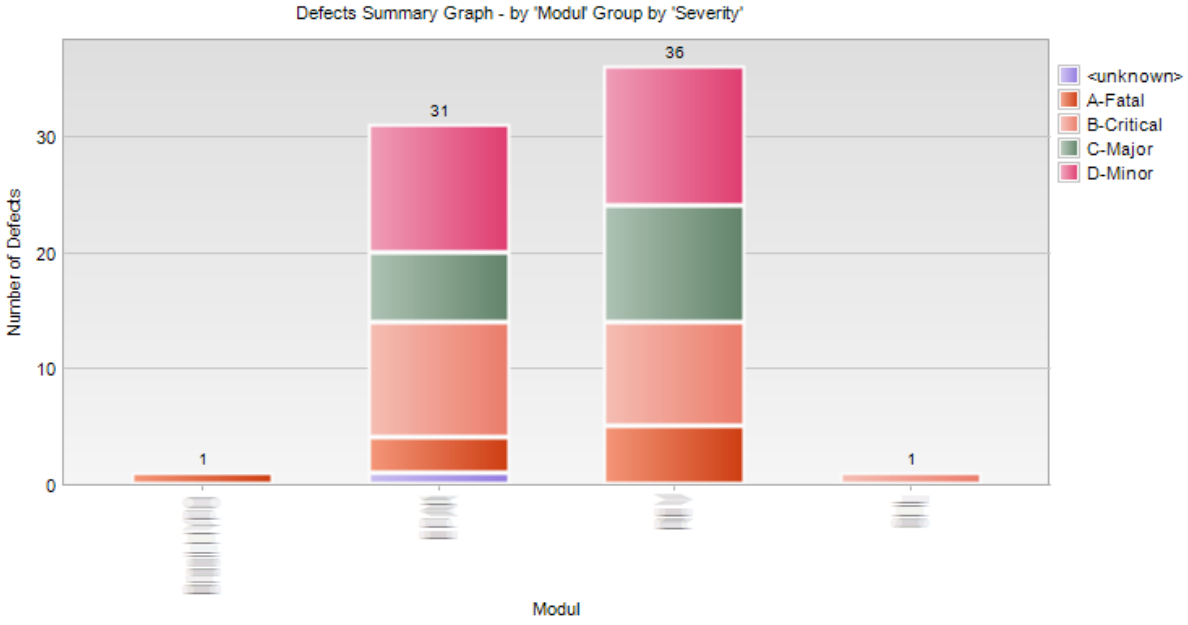
Obr. 21 - Report stavu defektů dle závažnosti (zdroj: autor)

6.3.3 Report závažnosti defektů dle testovaných modulů

Na následujícím grafu (obr. 25) má manažer testování možnost vidět moduly, které jsou řízeny pod manažerovým vedením. Osa x uvádí jednotlivé moduly a osa y počet defektů v těchto modulech. Graf ukazuje na který modul je vystaveno více chyb. Tyto chyby jsou následně rozděleny dle závažnosti. Konfigurace reportu je uvedena tabulce č. 4.

Tabulka 4 - Konfigurace reportu 3 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Defects
Graph Type	Summary Graph
Filter	Detection Date
Group By field	Severity
X-axis field	Modul



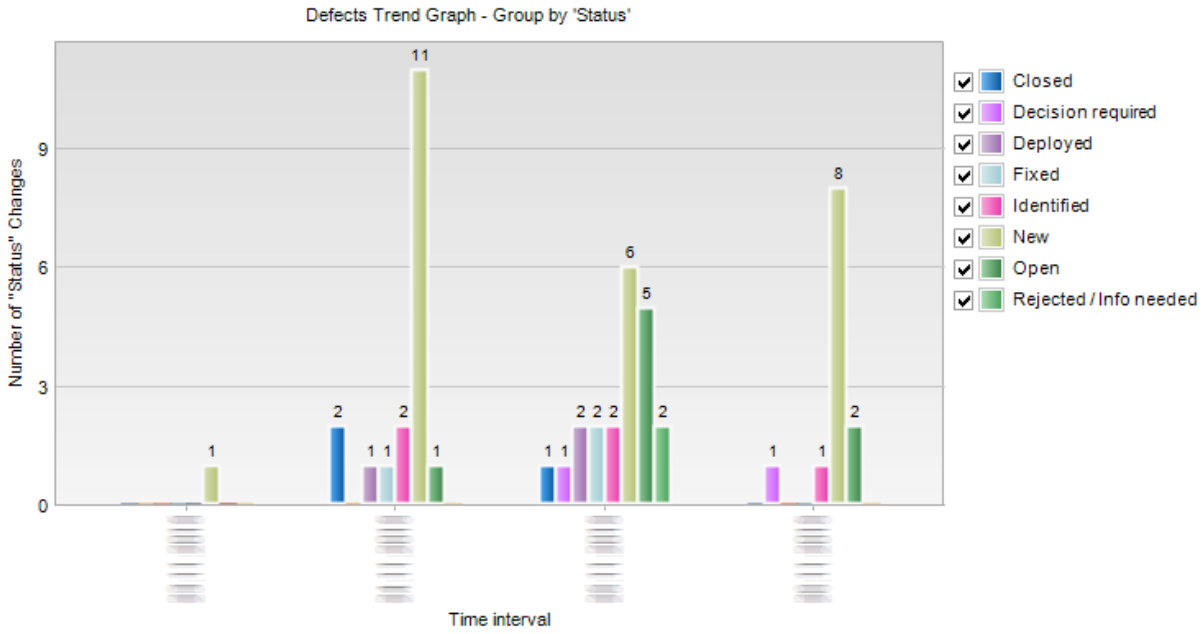
Obr. 22 - Report závažnosti defektů dle testovaných modulů (zdroj: autor)

6.3.4 Report stavu defektů dle časového úseku

Report (obr. 26) stavu defektů dle časového úseku zobrazuje na ose x definovaný časový úsek. Pro tento report je osa x rozdělena na dny. Osa y znázorňuje počet změn stavu defektů pro jednotlivé dny. Stavy defektů jsou: uzavřen, žádáno rozhodnutí, nasazen, opraven, indentifikován, nový, otevřený, vrácený. Konfigurace reportu je uvedena tabulce č. 5.

Tabulka 5 - Konfigurace reportu 4 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Defects
Graph Type	Trend Graph
Filter	Detection Date; Modul
Group By field	Status



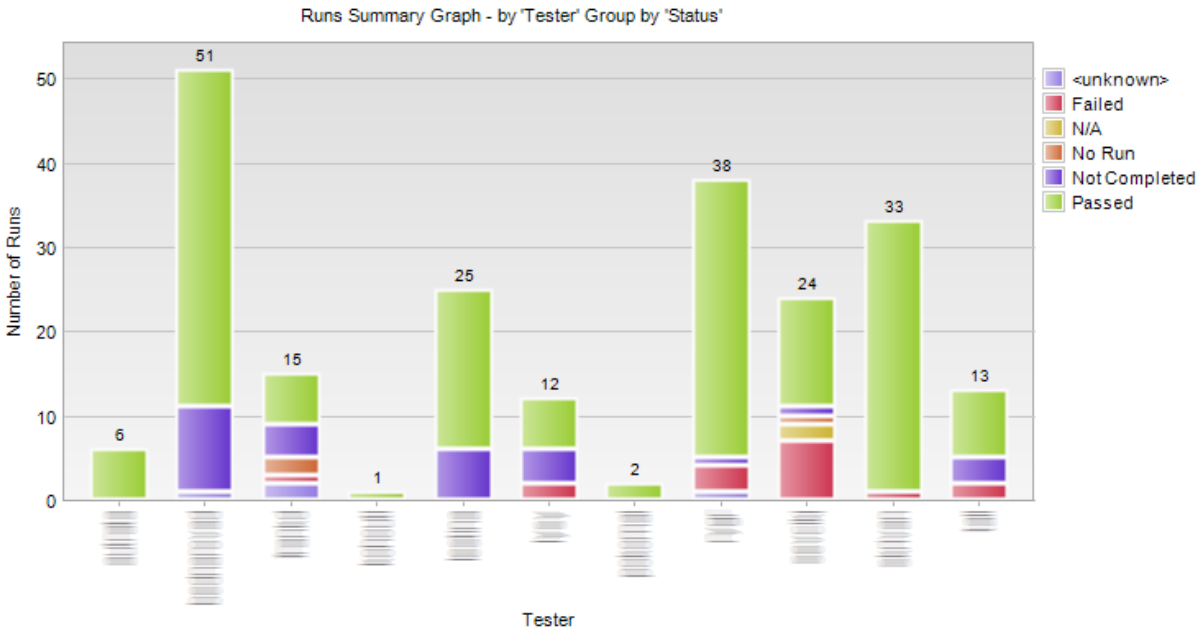
Obr. 23 - Report stavu defektů dle časového úseku (zdroj: autor)

6.3.5 Report stavu otestovaných sad dle testerů

Tento report (obr. 27) se již neřadí mezi reporty defektů, ale jedná se o report stavů spuštěných testovacích sad. Dle filtru je možno nastavit konkrétní období a lze vidět jakých výsledků dosáhli testeři při testování vytvořených testovacích sad. Konfigurace reportu je uvedena v tabulce č. 6.

Tabulka 6 - Konfigurace reportu 5 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Runs
Graph Type	Summary Graph
Filter	Exec Date
Group By field	Status
X-axis field	Tester



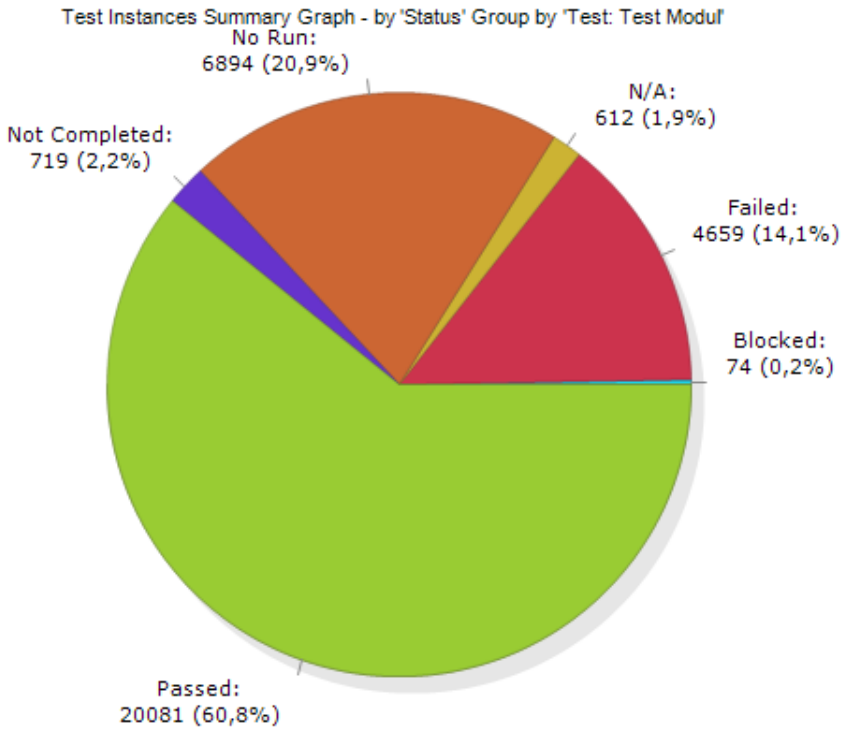
Obr. 24 - Report stavu otestovaných sad dle testerů (zdroj: autor)

6.3.6 Report stavu všech testovaných kroků

Z uvedeného reportu (obr. 28) je možno zjistit jaký je aktuální stav všech existujících testovacích kroků, které jsou umístěny v jednotlivých testovacích sadách. Po vygenerování reportu bylo zobrazení grafu přepnuto na výsečové - v HP ALM "Pie Chart". V tomto grafu jsou zahrnuty veškeré testované moduly. Konfigurace reportu je uvedena v tabulce č. 7.

Tabulka 7 - Konfigurace reportu 6 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Test Instances
Graph Type	Summary Graph
Group By field	Test: Test Modul
X-axis field	Status



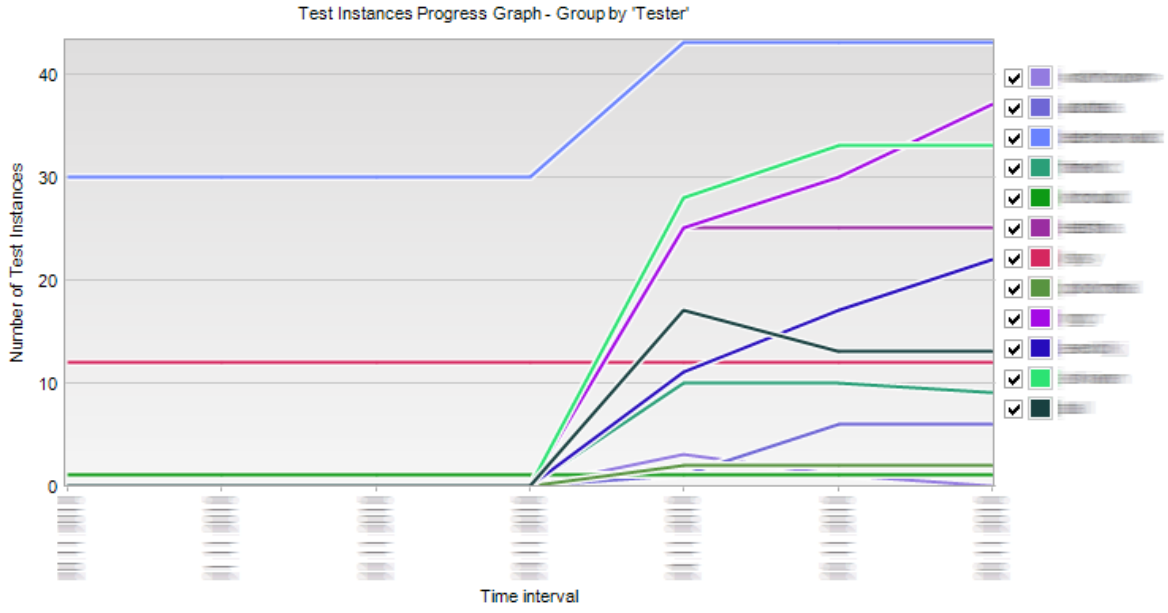
Obr. 25 - Report stavu všech testovaných kroků (zdroj: autor)

6.3.7 Report vývoje otestovaných kroků dle testerů v časovém úseku

Na následujícím spojnicovém grafu (obr. 29) je osa x nadefinována dle denního časového úseku. Osa y znázorňuje počty jednotlivých testovacích kroků. Manažerovi je umožněno dosáhnout informací jak se testovací postup podřízených testerů vyvíjí oproti předchozím dnům. Konfigurace reportu je uvedena v tabulce č. 8.

Tabulka 8 - Konfigurace reportu 7 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Test Instances
Graph Type	Progress Graph
Filter	Exec Date
Group By field	Tester



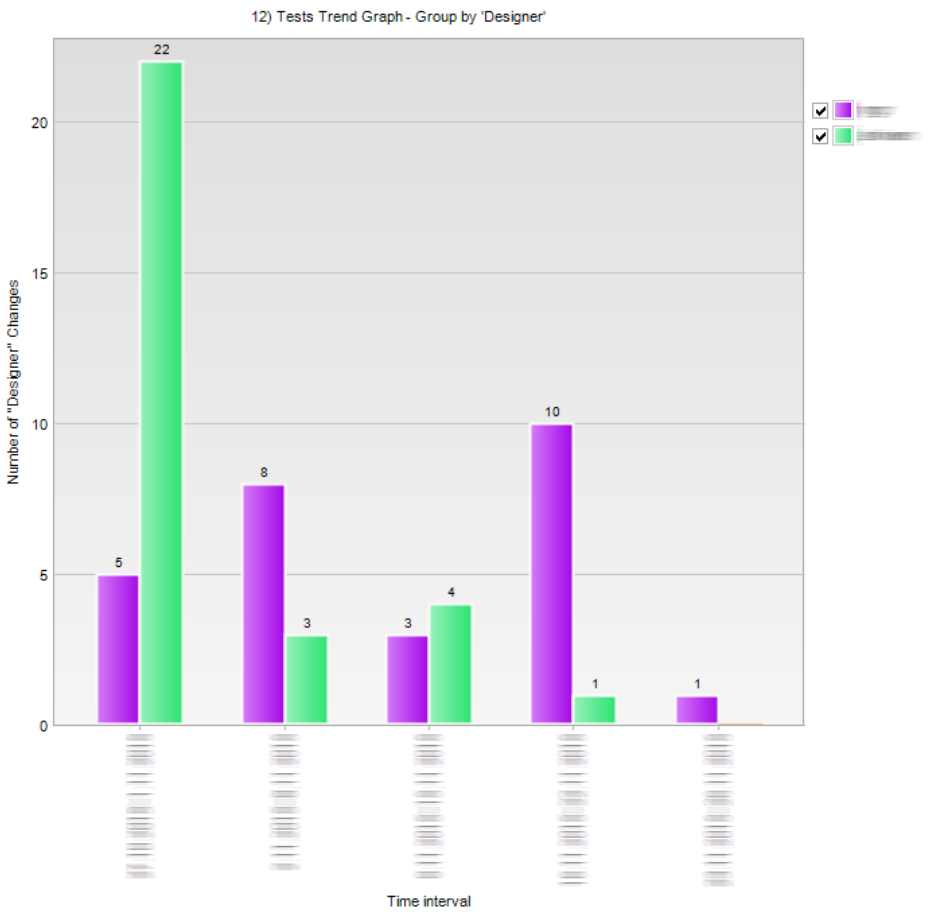
Obr. 26 - Report vývoje otestovaných kroků dle testerů v časovém úseku (zdroj: autor)

6.3.8 Report psaní testů testery v časovém úseku

Report (obr. 30) týkající se psaní testů v modulu aplikace HP ALM Test Plan. Osa x znázorňuje nastavený časový interval a osa y počty napsaných testů. Pro tento report si může uživatel nastavit testery, které chce na grafu zobrazit. Dále je možno rozdělit osu x na období měsíce dle týdnů. Konfigurace reportu je uvedena v tabulce č. 9.

Tabulka 9 - Konfigurace reportu 8 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Tests
Graph Type	Trend Graph
Filter	Designer
Group By field	Designer



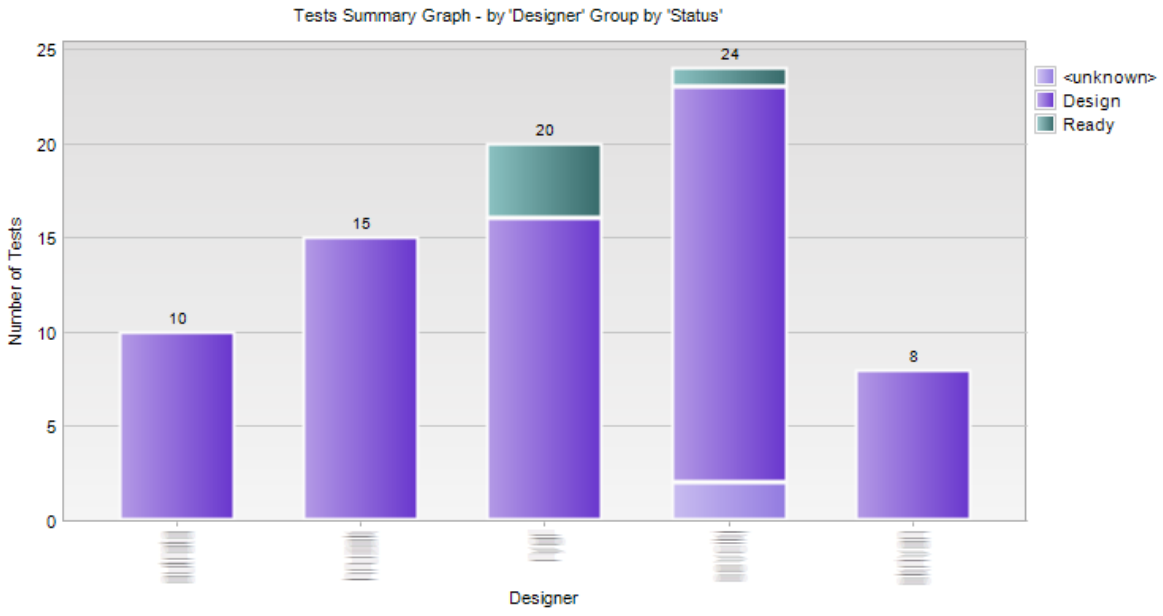
Obr. 30 - Report psaní testů testery v časovém úseku (zdroj: autor)

6.3.9 Report stavu psaných testů testery

Report (obr. 31) znázorňuje na ose x uživatele, kteří vytvářejí testy v sekci Test Plan. Na ose y lze vidět počet testů. Dle daného grafu lze říci, zda jsou napsané testy připraveny ke schválení či zda jsou stále ve vývoji apod. Konfigurace reportu je uvedena v tabulce č. 10.

Tabulka 10 - Konfigurace reportu 9 (zdroj: autor)

Pole v průvodci tvorbou reportu	Zadávaná hodnota
Entity	Tests
Graph Type	Summary Graph
Filter	Creation Date; Modul
Group By field	Status
X-axis field	Designer



Obr. 31 - Report stavu psaných testů testery (zdroj: autor)

7. Závěr

Informační systémy a technologie jsou v dnešní době nepostradatelné pro všechny sféry lidské činnosti. Vyvíjely se postupně a s jejich důležitostí vzrůstají i nároky na jejich kvalitu. Kvalitní a spolehlivý software je jak konkurenční výhodou firmy, tak i zárukou její úspěšnosti. Klíčovým faktorem kvality softwaru je proces jeho testování. Tento proces zajišťuje ověřování softwarového produktu, zda odpovídá daným požadavkům.

Testování je prováděno prostřednictvím různých standardů a nástrojů s cílem ověření kvality softwaru a systémů navrhovaných vývojovými týmy. Všechny metody testování jsou tedy založeny na procesu prověřování kvality při vývoji softwaru. Podoba testování softwaru se mění (nové technologie, programy), ale stále zůstává nutnou součástí vývoje softwaru a vyvíjených systémů.

Bakalářská práce byla zaměřena na test reporting s cílem vypracování tutoriálu pro vytváření test reportů s příklady v nástroji HP Application Lifecycle Management. Dílčím cílem bylo přiblížit základní aspekty testování a zajišťování kvality, rozebrat nejdůležitější pojmy, cíle a způsoby testování.

Splnění vymezených cílů práce bylo dosaženo těmito uvedenými kapitolami:

Ve druhé kapitole je vyzdvižen význam pojmu kvalita softwaru, model FURPS, který je určen pro kategorizaci požadavků na produkt. Dále se tato kapitola zabývá rozlišením často zaměňovaných pojmů zajišťování a řízení kvality, verifikace a validace.

Pro dosahování vysoké kvality je třeba vykonávat proces testování softwaru, který je popsán v kapitole č. 3. Dále je zde definován testovací tým s rolemi jeho členů, popsání důležitých pojmů testování, způsobů testování a určení zahájení a ukončení testování.

Čtvrtá kapitola pojednává o řízení testování softwaru. Definuje klíčové faktory pro vytvoření dostatečně kvalitního testovacího týmu. Uvádí příklady testovací dokumentace, která je součástí reportování výsledků.

Důležitá je pátá část bakalářské práce, která je zaměřena na teorii procesu test reportingu s uvedením metrik pro měření kvality. Zde jsou popsány některé používané metriky v praxi, které jsou podloženy vzorci pro výpočet. Metriky jsou základními prvky pro vytváření reportů.

Poslední část tvoří práce s nástrojem HP Application Lifecycle Management. Zde se nachází tutoriály jak vytvářet reporty a nástěnky z nadefinovaných reportů v tomto nástroji. Dále jsou uvedeny některé příklady vytvořených reportů s jejich konfiguracemi.

Terminologický slovník

Termín	Zkratka	Význam [zdroj]
brainstorming	-	Skupinová technika komunikace zaměřená na generování co nejvíce nápadů na dané téma [52].
implementace	-	Proces uskutečňování teoreticky stanovené myšlenky nebo projektu za účelem jejího dalšího použití. Implementaci předchází analýza zadání, plánování postupu a očekávaných výsledků [58].
Mean Time Between Failures (střední doba mezi poruchami)	MTBF	Střední doba mezi poruchami je statistická veličina, která slouží k ohodnocení spolehlivosti výrobku, nebo výrobního zařízení [59].
metrika	-	Kritérium měření kvality produktu, podle kterého posuzujeme jeho efektivitu, výkonnost, vývoj, kvalitu plánu a průběh [46].
operační systém	OS	Základní programové vybavení počítače, které je zavedeno do paměti počítače při jeho startu a zůstává v činnosti až do jeho vypnutí [53].
release	-	Verze softwaru uvolněná pro běžné použití [autor].
report	-	Slovem report se rozumí zpráva nebo hlášení o určitém stavu [autor].
reportování / reporting	-	Reporting testování je pojem, který by se dal definovat jako činnost, pomocí které je možné zjistit v jaké fázi a v jakém aktuálním stavu se testování softwaru nachází [autor].
řízení kvality (Quality Control)	QC	Soubor aktivit zaměřených na identifikování defektů v již vytvořených produktech k zajištění kvality [23].
screenshot	-	Jedná se o snímek obrazovky, který je možné uložit na pevný disk počítače [autor].
software	SW	Software je v informatice sada všech počítačových programů používaných v počítači, které provádějí nějakou činnost [54].

specifikace	-	Specifikace někdy též zaměření je detailní popis; jasné a přesné vymezení kvality, rozměrů a vlastností surovin, zboží, výrobků, materiálů, součástek a podobně [55].
Structured Query Language	SQL	SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích [56].
Unified Modeling Language	UML	Je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů [57].
validace	-	Proces, který slouží pro vyhodnocení produktu na konci fáze vývoje, kde se střetne konečný produkt s očekáváními zákazníka a jeho požadavky [10].
verifikace	-	Proces, kterým je třeba se ujistit, zda daný produkt splňuje předem specifikované požadavky již ve fázi vývoje [10].
zajišťování kvality (Quality Assurance)	QA	Soubor aktivit zajišťující kvalitu v procesech cyklu vývoje produktu [23].

Zdroje

[1] BOROVCOVÁ, Anna. *Testování webových aplikací* [online]. Praha, 2008 [cit. 2015-11-25]. Dostupné z: https://is.cuni.cz/webapps/zzp/detail/46536/

[2] BORŮVKA, Zdeněk. *Testování webových aplikací* [online]. Praha, 2009 [cit. 2015-11-25]. Dostupné z: https://www.vse.cz/vskp/16453_zpusoby_overeni_kvality_aplikaci_a_systemu_(metodika_nastroje)

[3] FAUSTOVÁ, Tereza. *Nástroje na podporu testování* [online]. Praha, 2009 [cit. 2015-11-25]. Dostupné z: https://www.vse.cz/vskp/15791_nastroje_na_podporu_testovani

[4] PAVELKA, Tomáš. *Nástroje pro podporu testování a jejich praktické využití* [online]. Praha, 2014 [cit. 2015-11-25]. Dostupné z: https://www.vse.cz/vskp/40782_nastroje_pro_podporu_testovani_a%C2%A0jejich_prakticke_vyuziti

[5] KRÁLOVÁ, Iveta. *Metodika testování podle mezinárodních praktik a standardů* [online]. Praha, 2013 [cit. 2015-11-25]. Dostupné z: http://www.vse.cz/vskp/eid/38510

[6] LÍZNER, Tomáš. *Využití automatizace testování z hlediska nákladů a přínosů* [online]. Praha, 2014 [cit. 2015-11-25]. Dostupné z: https://www.vse.cz/vskp/44196_vyuziti_automatizace_testovani_z_hlediska_nakladu_a_prinosu

[7] RYBAK, Yauhen. *Testování softwaru* [online]. Praha, 2012 [cit. 2015-11-25]. Dostupné z: https://www.vse.cz/vskp/33208_testovani_softwaru

[8] ŠPLÍCHALOVÁ, Marcela. *Metodika testování webových aplikací* [online]. Praha, 2008 [cit. 2015-11-25]. Dostupné z: http://www.vse.cz/vskp/eid/7634

[9] VACHALEC, Vladan. *Testování a kvalita softwaru v metodikách vývoje softwaru* [online]. Praha, 2014 [cit. 2015-11-25]. Dostupné z: https://www.vse.cz/vskp/40451_testovani_a%C2%A0kvalita_softwaru_v%C2%A0metodikach_vyvoje_softwaru

[10] ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. *Řízení kvality softwaru: průvodce testováním*. 1. vyd. Brno: Computer Press, 2013, 208 s. ISBN 978-80-251-3816-8.

[11] PATTON, Ron. *Testování softwaru*. Vyd. 1. Praha: Computer Press, 2002, xiv, 313 s. Programování. ISBN 80-722-6636-5.

[12] KANER, Cem, Jack L FALK a Hung Quoc NGUYEN. *Testing computer software*. 2nd ed. New York: John Wiley, 1999, xv, 480 s. ISBN 04-713-5846-0.

[13] FARRELL-VINAY, Peter. *Manage software testing*. Boca Raton: Auerbach Publications, c2008, xxiii, 573 s. ISBN 978-0-8493-9383-9.

[14] PERRY, William E. *Effective methods for software testing*. 3rd ed. Indianapolis, IN: Wiley, c2006, xiv, 313 s. Programování. ISBN 978-076-4598-371.

[15] Testing Excellence. *Testing Excellence | Software Testing for Beginners and Experts* [online]. [cit. 2015-11-28]. Dostupné z: http://www.testingexcellence.com/

[16] Software Testing Help. *Software Testing Help - A Must Visit Software Testing Portal* [online]. [cit. 2015-11-28]. Dostupné z: http://www.softwaretestinghelp.com/

[17] Kvalita (jakost). *Sociální síť pro business - ManagementMania.com* [online]. [cit. 2015-04-02]. Dostupné z: http://managementmania.com/cs/kvalita-jakost

[18] KRÁTKÝ, Tomáš. Software Quality Assurance. *PROFINIT* [online]. [cit. 2015-04-02]. Dostupné z: http://www.profinit.eu/fileadmin/Content/profinit.eu.maas.at/Academy/NSWI129/prednasky/06_QA.pdf

[19] FURPS. *Wikipedia, the free encyclopedia* [online]. [cit. 2015-04-02]. Dostupné z: http://en.wikipedia.org/wiki/FURPS

[20] UML - Analýza požadavků na systém. *Martin Řimnáč* [online]. [cit. 2015-04-03]. Dostupné z: http://www.cs.cas.cz/~rimnacm/posts/UML-funkcionalni-a-nefunkcionalni-pozadavky.html

[21] FURPS. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-04-03]. Dostupné z: http://cs.wikipedia.org/wiki/FURPS

[22] Agile in a Flash: FURPS+. *Agile in a Flash* [online]. [cit. 2015-04-03]. Dostupné z: http://agileinaflash.blogspot.cz/2009/04/furps.html

[23] Quality Assurance vs. Quality Control. *Diffen - Compare Anything. Diffen. Discern. Decide.* [online]. [cit. 2015-04-13]. Dostupné z: http://www.diffen.com/difference/Quality_Assurance_vs_Quality_Control

[24] Difference between Verification and Validation. *Software Testing Classes* [online]. [cit. 2015-04-13]. Dostupné z: http://www.softwaretestingclass.com/difference-between-verification-and-validation/

[25] BELKHATOUI, Nizar. Verification Vs Validation software testing: what is the difference ?. *Focus Technological Blog - Software Development and IT Outsourcing services* [online]. [cit. 2015-04-13]. Dostupné z: http://www.focus-itoutsourcing.com/verification-vs-validation-what-is-the-difference/

[26] What is Software Testing?. *ISTQB Exam Certification – Study material for Foundation level, through ISTQB and ASTQB Exam, Certification questions, answers, tutorials and more* [online]. [cit. 2015-04-14]. Dostupné z: http://istqbexamcertification.com/what-is-a-software-testing/

[27] Improving Your Software Testing: Focus on the Testing Team. *InformIT: The Trusted Technology Source for IT Pros and Developers* [online]. [cit. 2015-04-14]. Dostupné z: http://www.informit.com/articles/article.aspx?p=31196

[28] Test Case. *Tutorials for Anger Management, Social Media Marketing, AIML, Artificial Intelligence, RESTful, Swift, Node.js, LinQ, Drools, Content Marketing, SIP, Pay per Click, Accounting, Sqoop, ITIL, Jackson, Security Testing, Awk, JDB, Hadoop, Hive, HBase, XStream, Java8, Guava, Memcached, EasyMock, CICS, Object Oriented Analysis & Design, Tika, DOM, AngularJS, IMS-DB, NGN, Sed, WCF, JPA, Apache POI, Signals and Systems, JOGL, DB2, PhoneGap, SVG, VSAM, COBOL, XSD, XPath, Redis, XSLT, Java XML, VBA, JFree* [online]. [cit. 2015-04-14]. Dostupné z: http://www.tutorialspoint.com/software_testing_dictionary/test_case.htm

[29] Test data. *Collins Dictionaries | Always Free Online.* [online]. [cit. 2015-04-14]. Dostupné z: http://www.collinsdictionary.com/dictionary/english/test-data

[30] Why is software testing necessary?. *ISTQB Exam Certification – Study material for Foundation level, through ISTQB and ASTQB Exam, Certification questions, answers, tutorials and more* [online]. [cit. 2015-04-14]. Dostupné z: http://istqbexamcertification.com/why-is-testing-necessary/

[31] What is black-box, Specification-based, also known as behavioral testing techniques?. *ISTQB Exam Certification – Study material for Foundation level, through ISTQB and ASTQB Exam, Certification questions, answers, tutorials and more* [online]. [cit. 2015-04-14]. Dostupné z: http://istqbexamcertification.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques/

[32] KANER, Cem, Jack L FALK a Hung Quoc NGUYEN. *Testing computer software*. 2nd ed. New York: John Wiley, 1999, xv, 480 s. ISBN 04-713-5846-0.

[33] What is white-box or Structure-based or structural testing techniques?. *ISTQB Exam Certification – Study material for Foundation level, through ISTQB and ASTQB Exam, Certification questions, answers, tutorials and more* [online]. [cit. 2015-04-16]. Dostupné z: http://istqbexamcertification.com/what-is-white-box-or-structure-based-or-structural-testing-techniques/

[34] What is static Testing?. *ISTQB Exam Certification – Study material for Foundation level, through ISTQB and ASTQB Exam, Certification questions, answers, tutorials and more* [online]. [cit. 2015-04-16]. Dostupné z: http://istqbexamcertification.com/what-is-static-testing/

[35] Static Vs Dynamic Testing. *Meet Guru99* [online]. [cit. 2015-04-18]. Dostupné z: http://www.guru99.com/static-dynamic-testing.html

[36] What is Dynamic testing technique?. *ISTQB Exam Certification – Study material for Foundation level, through ISTQB and ASTQB Exam, Certification questions, answers, tutorials and more* [online]. [cit. 2015-04-18]. Dostupné z: http://istqbexamcertification.com/what-is-dynamic-testing-technique/

[37] Management. *BusinessDictionary.com - Online Business Dictionary* [online]. [cit. 2015-04-20]. Dostupné z: http://www.businessdictionary.com/definition/management.html

[38] Project management triangle. *Wikipedia, the free encyclopedia* [online]. [cit. 2015-04-20]. Dostupné z: http://en.wikipedia.org/wiki/Project_management_triangle

[39] How to Form an Effective Test Team. *Software Testing Complete Guide* [online]. [cit. 2015-04-20]. Dostupné z: http://www.softwaretestinghelp.com/how-to-form-an-effective-test-team/

[40] Test Policy Document. *Testing Excellence | Software Testing for Beginners and Experts* [online]. [cit. 2015-04-20]. Dostupné z: http://www.testingexcellence.com/test-policy-document/

[41] Test Strategy and Test Plan. *Testing Excellence | Software Testing for Beginners and Experts* [online]. [cit. 2015-04-20]. Dostupné z: http://www.testingexcellence.com/test-strategy-and-test-plan/

[42] Elements of a Bug Report. *Testing Excellence | Software Testing for Beginners and Experts* [online]. [cit. 2015-04-20]. Dostupné z: http://www.testingexcellence.com/elements-of-a-bug-report/

[43] How to Report Test Execution Smartly. *Software Testing Complete Guide* [online]. [cit. 2015-04-20]. Dostupné z: http://www.softwaretestinghelp.com/test-execution-report/

[44] Business Dictionary. *What is test report? definition and meaning* [online]. [cit. 2015-11-15]. Dostupné z: http://www.businessdictionary.com/definition/test-report.html

[45] Computerworld.cz. *Správné použití a načasování test-reportů* [online]. [cit. 2015-11-15]. Dostupné z: http://computerworld.cz/ness-up-ideas/spravne-pouziti-a-nacasovani-test-reportu-50477

[46] Metrics. *BusinessDictionary.com - Online Business Dictionary* [online]. [cit. 2015-04-22]. Dostupné z: http://www.businessdictionary.com/definition/metrics.html

[47] Important Software Test Metrics and Measurements – Explained with Examples and Graphs. *Software Testing Complete Guide* [online]. [cit. 2015-04-22]. Dostupné z: http://www.softwaretestinghelp.com/software-test-metrics-and-measurements/

[48] KLOC – What does it mean to Software Testing. *Software Testing Blog* [online]. [cit. 2015-04-22]. Dostupné z: http://venkatreddyc.wordpress.com/2007/02/28/kloc-what-does-it-mean-to-software-testing/

[49] HP Application Lifecycle Management. *Norwood Technologies - Software Testing Service Training* [online]. [cit. 2015-04-25]. Dostupné z: http://norwood-technologies.com/documents/QC-11-Tutorial.pdf

[50] HP Quality Center. *Wikipedia, the free encyclopedia* [online]. [cit. 2015-04-25]. Dostupné z: http://en.wikipedia.org/wiki/HP_Quality_Center

[51] Image. *Application Lifecycle Management - IKAN ALM* [online]. [cit. 2015-04-25]. Dostupné z: http://www.ikanalm.com/images/stories/solutions/hp/hp-alm-quality-center-automated-test-log.png

[52] Brainstorming. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/Brainstorming

[53] Operační systém. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/Opera%C4%8Dn%C3%AD_syst%C3%A9m

[54] Software. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/Software

[55] Specifikace. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/Specifikace

[56] SQL. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/SQL

[57] Unified Modeling Language. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-04-26]. Dostupné z: http://cs.wikipedia.org/wiki/Unified_Modeling_Language

[58] Implementace. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-29-04]. Dostupné z: http://cs.wikipedia.org/wiki/Implementace

[59] Střední doba mezi poruchami. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2015-29-04]. Dostupné z: http://cs.wikipedia.org/wiki/St%C5%99edn%C3%AD_doba_mezi_poruchami

Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek