**Vysoká škola ekonomická v Praze**
**Fakulta informatiky a statistiky**
**Katedra informačních technologií**

Studijní program : Aplikovaná informatika
Obor : Informační systémy a technologie

# Metodika testování podle mezinárodních praktik a standardů

**DIPLOMOVÁ PRÁCE**

Student : Bc. Iveta Králová
Vedoucí : doc. Ing. Alena Buchalcevová, Ph.D.
Oponent : Ing. et Ing. Michal Doležel

**2013**

## Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracovala samostatně a že jsem uvedla všechny použité prameny a literaturu, ze které jsem čerpala.

V Praze dne 22. 6. 2013 . . . . . .

Iveta Králová

## Poděkování

*Tímto bych ráda poděkovala vedoucí diplomové práce paní doc. Ing. Aleně Buchalcevové, Ph.D. za vstřícnost, trpělivost, odborné rady, cenné podněty a připomínky, které mi poskytla v průběhu tvorby této diplomové práce. Dále bych chtěla poděkovat i své rodině a blízkým, kteří mě při psaní práce i během celého studia velmi podporovali.*

# Abstrakt

V oboru softwarového inženýrství existuje mnoho metodik, které jsou zaměřeny na celý proces vývoje softwaru. Cílem procesu vývoje softwaru je dodat kvalitní produkt, který uspokojuje požadavky uživatelů, dodat jej včas a v rámci stanoveného rozpočtu. Testování je nástrojem ověřování kvality softwaru a funguje při vývoji softwaru jako zpětná vazba. Většina metodik vývoje softwaru je k dispozici pouze v angličtině, klade důraz spíše na samotný vývoj softwaru a oblasti testování se věnuje pouze okrajově. Správně nastavený proces testování ovšem může přispět k vyšší kvalitě softwaru a zabránit možným dopadům při selhání celého informačního systému. Současné české metodiky zaměřené výhradně na oblast testování vznikly buď interně ve firmách, kde nejsou přístupné veřejnosti, nebo v rámci diplomových prací, kde způsobem vytvoření bylo typicky přizpůsobení a rozšíření stávajících metodik vývoje softwaru pro testování v konkrétní společnosti nebo týmu.

Hlavním cílem této diplomové práce je vytvoření metodiky testování v češtině, která vychází z mezinárodních praktik a standardů, není přizpůsobena na míru konkrétní organizaci a podle které bude možné postupovat při definování procesů testování v různých organizacích bez ohledu na jejich specifické zaměření. Navržená metodika je zároveň implementována v nástroji EPF Composer, který umožňuje správu metodiky i její publikaci ve formě webové prezentace. Diplomová práce sleduje i cíle další. Za účelem ověření naplnění cíle vytvoření standardizované metodiky testování je cílem práce také následné zhodnocení metodiky pomocí standardizovaného procesního rámce pro zlepšování procesů testování a shrnutí, které oblasti mohou být předmětem dalšího rozšiřování metodiky. Cílem teoretické části práce je hodnocení metodik vývoje softwaru z pohledu testování, uvedení přehledu standardů se vztahem k testování, charakteristika vybraných standardů, objasnění účelu certifikací v oblasti testování a analýza možností využití materiálů pro přípravu na certifikace ISTQB. Cílů bylo dosaženo především analýzou dostupných materiálů pro certifikace ISTQB, analýzou metodik vývoje softwaru, standardů, procesního rámce TMMi a dalších zdrojů, které se zabývají oblastí testování, metodikami a standardy v oblasti testování. Hlavním přínosem práce je metodika testování, která je postavena na procesu testování definovaným mezinárodní organizací ISTQB, kterému dodává metodický rámec v podobě propojení a rozpracování jednotlivých metodických prvků. Jedním z prvků metodiky jsou pracovní produkty, které metodika specifikuje s ohledem na standard IEEE 829. Metodika není závislá na potřebách konkrétní společnosti a je možné ji jednoduše převzít, spravovat a dále přizpůsobovat. Může být využita jako výchozí bod při tvorbě testovací strategie v českých organizacích, které se zabývají vývojem softwaru a mají zájem k oblasti testování přistupovat podle mezinárodních praktik.



## Klíčová slova:

Metodika testování, ISTQB, standard, úloha, role, technika, pracovní produkt, defekt

# Abstract

There are many methodologies in the field of software engineering that focus on the entire software development process. The software development process aims to provide a quality product that meets user requirements, and to deliver it on time as well as within budget. Testing is a tool for verifying the quality of software and serves as feedback in software development. Most software development methodologies are available only in English, they emphasize the actual software development and address testing only marginally. An accurately designed testing process however, may contribute to higher software quality and avoid potential problems resulting from the entire information system's failure. Current Czech methodology that focuses exclusively on testing has been developed either inside individual companies and thus is not accessible to the public, or is contained in graduate theses, where it was typically created by adapting and expanding the existing software development methodologies for testing within a particular company or team.

The main objective of this thesis is to establish a methodology for testing in Czech, which is based on international practices and standards, which is not tailored to a specific organization and which can be used as guidance in defining test processes in various organizations, regardless of their specific focus. The proposed methodology is also implemented in the tool EPF Composer that enables administration of methodology and its publication in the form of the website. The thesis also pursues additional objectives. In order to verify the objective of developing a standardized testing methodology is met, the thesis strives to produce a subsequent evaluation methodology using a standardized process framework in order to improve the testing process as well as a summary detailing which areas may be the subject of further expansion methodology. The theoretical part of the thesis aims to evaluate software development methodologies in terms of testing, to provide an overview of standards related to testing and to describe selected standards, to clarify the purpose of certification in testing and to analyze the possibilities of utilizing materials to prepare for the ISTQB certification. The objectives were achieved by analyzing the available materials for ISTQB certification, analyzing software development methodologies, standards, the TMMi procedural framework and other resources that deal with testing as well as its methodologies and standards. The main contribution of this work is a testing methodology, which is based on the testing process as defined by the international organization ISTQB. It provides a methodological framework by interconnecting and expanding individual methodological elements. Elements of the methodology include working products, as specified by the methodology with respect to the IEEE 829 standard. The methodology does not depend on the needs of a particular company and may be easily adopted, managed and customized. It may be used as a starting point in creating a testing strategy in Czech companies focused on software development which are interested in approaching testing in compliance with international practices.



## Keywords:

Testing methodology, ISQTB, standard, task, role, technique, work product, defect

# Obsah

# 1 Úvod

Pro oblast informačních systémů a technologií (IS/ICT) je v dnešní době charakteristický dynamický vývoj, který s sebou přináší rapidní nárůst jejich komplexity. Informační systémy už neslouží pouze jako podpora firemních procesů, jako tomu bylo kdysi. Mnohým podnikům může propojení IS/ICT s podnikovými procesy a podnikovou kulturou přinést dokonce konkurenční výhodu a pomoci čelit potencionálním hrozbám. Požadavky na vyvíjené informační systémy prudce stoupají, a proto již nestačí přistupovat k vývoji systémů ad hoc, ale je zapotřebí proces vývoje vhodně organizovat. Jádrem informačních systémů je software.

Za účelem stanovení postupů, které pomáhají řešit složitost vývoje softwaru, vznikla řada metodik vývoje softwaru. Metodiky vývoje softwaru představují sadu postupů, pravidel, principů, procesů, praktik, technik, nástrojů, návodů, rolí, pracovních produktů a dalších prvků, jejichž cílem je proces vývoje softwaru do jisté míry formalizovat a vymezit zodpovědnosti za jednotlivé činnosti a výstupy.

Vzhledem k vysoké komplexitě softwaru, je velmi významnou součástí procesu vývoje softwaru i testování. Čím je software složitější, tím náchylnější je také k defektům, které mohou vést k následným selháním systému. Pokud defekty, které selhání systému mohou způsobit, nejsou nalezeny a odstraněny, mohou mít za následek značné škody. Cílem testování je ověřit, zda se systém chová v souladu s očekáváním uživatelů, odhalit případné defekty a upozornit na ně projektový tým, aby mohly být včas opraveny.

Cílem procesu vývoje softwaru je dodat kvalitní software, který uspokojuje požadavky uživatelů, dodat jej včas a v rámci stanoveného rozpočtu. Celková kvalita informačního systému je ovlivněna kvalitou softwaru. Testování je nástrojem ověřování kvality softwaru a funguje při vývoji softwaru jako zpětná vazba. Aby testování plnilo své poslání a poskytovalo relevantní informace o kvalitě vyvíjeného softwaru, je zapotřebí, aby i samotný proces testování byl po metodické stránce zvládnutý. Metodiky vývoje softwaru typicky kladou důraz spíše na samotný vývoj softwaru a oblasti testování se věnují pouze okrajově. Nicméně je třeba si uvědomit, že správně nastavený proces testování může přispět k vyšší kvalitě softwaru a zabránit možným dopadům při selhání celého informačního systému.

## 1.1 Vymezení tématu práce a důvod výběru tématu

Tématem této diplomové práce je především vytvoření metodiky testování podle mezinárodních praktik a standardů. Metodika je zaměřená na proces testování a prvky, které s procesem testování souvisí, nebo jej podporují. Účelem metodiky je vytvořit standardizovaný rámec procesu testování, podle kterého bude možné postupovat při definování procesu testování v organizacích zabývajících se vývojem softwaru.

Teoretická část práce se v kontextu k praktické části zaměřuje na metodiky vývoje z pohledu testování, standardy se vztahem k testování softwaru a certifikace v oboru testování.

Důvodem výběru tohoto tématu byl především můj zájem o oblast testování softwaru. Již dva roky se testování softwaru věnuji v praxi a vypozorovala jsem řadu problémů spojených s testováním. V České republice mnoho organizací stále postupuje při testování ad hoc a nejsou stanovené jednotné postupy, podle kterých by se při řešení projektů postupovalo. O testování existuje množství materiálů v angličtině, ale v českém jazyce je zatím literatury věnující se testování málo. Mým záměrem bylo přispět k dosavadním českým materiálům prací, která přibližuje mezinárodní praktiky a standardy. V oblasti testování softwaru chci působit i po ukončení studií a zkušenosti získané při vytváření metodiky hodlám ve své budoucí praxi uplatnit.

## 1.2 Cíle práce

Hlavním cílem diplomové práce je vytvořit metodiku pro organizaci procesu testování z hlediska řízení, přípravy i provádění testů, která vychází z mezinárodních praktik a standardů a není vytvořena na míru konkrétní organizaci. Podle navržené metodiky bude možné postupovat při definování procesu testování v různých organizacích bez ohledu na jejich specifické zaměření, a to zejména na středních a větších projektech, kde za testování zodpovídá testovací tým tvořený specializovanými testery. Metodika ovšem není určena pro organizace, které vyvíjí vysoce kritické aplikace. Navržená metodika je zároveň implementována v nástroji EPF Composer, který umožňuje správu metodiky i její publikaci ve formě webové prezentace.

Za účelem ověření naplnění cíle vytvoření standardizované metodiky testování je dalším cílem následně metodiku zhodnotit pomocí standardizovaného procesního rámce pro zlepšování procesů testování a shrnout, které oblasti jsou metodikou pokryty a které oblasti pokryty dostatečně nejsou a mohou být předmětem dalšího rozšiřování metodiky.

Dílčím cílem této práce je rozbor problematiky, která má vztah k vytváření metodiky testování. Cílem je objasnit, co to metodika je, uvést základní rozdělení metodik a zhodnotit tři vybrané metodiky vývoje softwaru z pohledu testování. Dalším cílem je uvést přehled standardů, které mají vztah k testování a charakterizovat některé z nich. Dále je cílem objasnit, k čemu jsou užitečné certifikace v oblasti testování a speciálně certifikace ISTQB. Jelikož navržená metodika testování vychází z materiálů pro certifikace ISTQB, cílem je analyzovat i jiné možnosti využití znalostí v materiálech poskytovaných organizací ISTQB, než je samotné dosažení certifikátu.

Předpokladem pro porozumění obsahu práce je vymezení základní terminologie, kterou práce používá.

Cílů práce jsem dosáhla analýzou dostupných materiálů, které poskytuje organizace ISTQB, analýzou metodik vývoje softwaru, standardů, procesního rámce TMMi a dalších zdrojů, které se zabývají oblastí testování, metodikami a standardy v oblasti testování. Především se jedná o odborné knihy a články, diplomové práce i webové stránky. Navíc jsem při vytváření metodiky a zpracování práce využila i znalosti z předmětu Řízení kvality softwaru a své dosavadní znalosti z praxe v oblasti testování.

## 1.3 Struktura práce

Diplomová práce je rozdělena na teoretickou a praktickou část.

Teoretické zaměření má druhá, třetí, čtvrtá a pátá kapitola práce. Ve druhé kapitole práce jsou vymezeny základní pojmy, které jsou v práci používány. Třetí kapitola práce uvádí definici metodiky vývoje softwaru, základní rozdělení metodik, přičemž hlavní pozornost je věnována hodnocení třech vybraných metodik vývoje softwaru z pohledu testování. Ve čtvrté kapitole práce je uveden přehled standardů, které mají vztah k testování a charakteristika některých z nich. Hlavní pozornost je věnována především standardu IEEE Standard for Software and System Test Documentation (IEEE 829) [IEEE, 2008], podle kterého jsou navrhovány pracovní produkty v praktické části práce. V páté kapitole práce je objasněno, co to je certifikát a k čemu mohou být užitečné certifikace v oblasti testování. Hlavní pozornost je věnována certifikacím ISTQB a možnostem využití znalostí v materiálech poskytovaných organizací ISTQB.

Praktické zaměřená má šestá a sedmá kapitola práce. V šesté kapitole práce je nejdříve charakterizována navrhovaná metodika testování a následně jsou specifikovány její prvky, rozdělené podle osmi hlavních aktivit procesu testování: Plánování testů, analýza testů, návrh testů, implementace testů, provádění testů, monitorování, reportování a řízení testování, vyhodnocení testování a uzavření testování. V příloze E je potom možné nalézt náhledy na vytvořenou metodiku v publikované verzi. V sedmé kapitole práce je navržená metodika podrobena hodnocení pomocí standardizovaného procesní rámce pro zlepšování procesů testování Test Maturity Model integration (TMMi) a na základě výsledků hodnocení je shrnuto, které oblasti metodika pokrývá dostatečně a které naopak dostatečně pokryty nejsou a mohou být předmětem dalšího rozšiřování metodiky.

## 1.4 Předpoklady a omezení práce

Navržená metodika se soustředí výhradně na oblast testování softwaru a nepokrývá ostatní procesy životního cyklu vývoje softwaru. Metodika je určena pro úroveň systémových testů, za kterou obvykle zodpovídá testovací tým a která ve srovnání s ostatními úrovněmi testování vyžaduje vyšší stupeň organizovanosti. Pro účely ostatních úrovní testů je možné metodiku přizpůsobit. V rámci životního cyklu metodika pokrývá pouze úsek vývoje do okamžiku nasazení softwaru do produkčního prostředí, nebo v případě, kdy

po úrovni systémových testů následuje úroveň akceptačních testů, tak do okamžiku předání softwaru k akceptačnímu testování. Metodikou není pokryta fáze provozu a údržby systému. Metodika se soustředí na testování softwaru v dynamickém pojetí a nepokrývá statické testování, jehož předmětem jsou revize zdrojového kódu a pracovních produktů. Navržená metodika není určena pro konkrétní společnost, a tak se nepředpokládá, že by mohla být ve své obecné podobě okamžitě v určité společnosti implementována. Při zavádění metodiky testování v konkrétní organizaci je možné z této metodiky vycházet, jelikož zahrnuje standardizované postupy a praktiky, které je možné pro konkrétní potřeby organizace přizpůsobit.

Proces testování pokrytý touto metodikou se blíží druhé úrovni zralosti dle modelu TMMi, nicméně vzhledem k rozsahu práce nebyla věnována pozornost kompletně všem 19 cílům, jejichž splnění druhá úroveň zralosti vyžaduje. Rámcově jsou pokryty všechny základní charakteristiky dle modelu TMMi, kromě politiky a strategie testování, které stojí mimo rámec této metodiky.

Uživateli této metodiky mohou být všichni, kdo jsou zainteresováni na procesu testování od vedoucích projektů, manažerů vývoje, vývojářů a zejména test manažeři, test analytici, testeři i správci testovacího prostředí a případní další zájemci.

## 1.5 Očekávané přínosy

Hlavním přínosem práce je vytvoření metodiky pro testování v českém jazyce, která vychází z mezinárodních praktik a standardů a není závislá na potřebách konkrétní společnosti. Metodika je postavena na procesu testování, který definuje světově uznávaná organizace International Software Testing Qualifications Board (ISTQB) [ISTQB, 2012a] ve svých materiálech pro přípravu na certifikaci. Tomuto procesu je oproti materiálům ISTQB dodán metodický rámec. Z hlavních aktivit, které ISTQB popisuje, jsou vyčleněny a rozpracovány jednotlivé úlohy. Na úlohy jsou potom navázány role, techniky a pracovní produkty, kde většina z nich byla vytvořena podle standardu IEEE Standard for Software and System Test Documentation (IEEE 829) [IEEE, 2008], nebo jsou zdůvodněny případné odchylky. Metodiku je možné využít jako výchozí bod při tvorbě testovací strategie v českých organizacích, které se zabývají vývojem softwaru a mají zájem k oblasti testování přistupovat podle mezinárodních praktik.

Metodiku je možné využít i v rámci kompetenčního centra Software Quality Assurance. Kompetenční centrum sdružuje studenty VŠE, kteří mají zájem si vedle studia osvojit i praktické zkušenosti a VŠE jim v tomto směru nabízí možnost zapojení do reálných projektů. Studenti mají přístup k nástroji pro řízení testování IBM Rational Quality Manager, na kterém navržená metodika demonstruje možnosti využití nástrojové podpory při řízení procesů testování.

Pro možnosti využití je významný i fakt, že navržená metodika je volně dostupná v podobě webové prezentace, a tak lze jednoduše procházet mezi jednotlivými prvky metodiky. Byla vytvořena a publikována pomocí nástroje pro správu metodik EPF Composer, kde je možné ji dále přizpůsobovat a rozšiřovat.

V neposlední řadě může být navržená metodika přínosem i pro studenty kurzů testování a řízení kvality softwaru, kteří se chtějí seznámit s tím, jak proces testování může probíhat a jaké pracovní produkty, role a techniky může obsahovat. Zároveň je možné metodiku využít i jako doplňkový materiál při přípravě na certifikaci ISTQB.

Kromě praktické části je přínosem práce i část teoretická, kde jsou analyzovány metodiky vývoje softwaru z pohledu testování. Zároveň jsou v rámci teoretické části v přehledné formě zpracovány standardy se vztahem k testování, které dosud nebyly předmětem zpracování žádné z dostupných českých diplomových prací ani knih.[1]

## 1.6 Rešerše literatury

Oblastí testování softwaru a metodikami testování se zabývá množství odborných knih, článků, diplomových a bakalářských prací i webových stránek. V této podkapitole shrnuji pouze ty zdroje, které mi zpracování této práce byly inspirací, jsou zaměřeny na příbuzné téma jako tato práce, nebo je v souvislosti s touto prací považuji za důležité zmínit. Pro větší přehlednost je tato kapitola rozčleněna do podkapitol podle druhu zdroje. V této rešerši se nevěnuji metodikám, standardům a materiálům pro certifikaci ISTQB, jelikož jsou předmětem samostatných kapitol teoretické části této práce. Zvlášť se nevěnuji ani webovým stránkám a článkům, jelikož jsem je v práci využila pouze jako doplňující zdroj.

### 1.6.1 Akademické práce

Na testování softwaru v obecné rovině se zaměřují diplomové práce ***„Testování webových aplikací“*** **[Borovcová, 2008]** a ***„Způsoby ověření aplikací a systémů (metodika a nástroje)“*** **[Borůvka, 2009]**.

Diplomová práce *„Testování webových aplikací“* [Borovcová, 2008] se zaměřuje na komplexní vysvětlení problematiky testování webových aplikací a svým zaměřením pokrývá základy prakticky celé oblasti testování jako je definice testování, chyby, testovací tým, kategorie testů, testovací dokumentace, typy testů a testovací techniky i automatizované a bezpečnostní testy.

---

[1] Standardy pro procesy životního cyklu softwaru jsou zpracovány v [Buchalcevová, 2009, s. 27-45]. Některé z nich mají vztah i k testování a zajišťování kvality softwaru, ale zpracování standardů v této knize není přímo zaměřeno na standardy ve vztahu k testování.

Diplomová práce *„Způsoby ověření aplikací a systémů (metodika a nástroje)“* [Borůvka, 2009] je zaměřená na testování spolu se systematickým ověřováním kvality všech projektových činností. Zasazuje testování do kontextu celého projektu, vymezuje typy testů, rozlišuje testování různých druhů aplikací, shrnuje testování dle metodiky RUP, zaměřuje se na různé druhy nástrojů pro podporu testování a poukazuje i na trendy a rizika testování. Obě tyto práce jsou vhodné pro rychlé získání povědomí o základech testování.

Další skupinou prací jsou bakalářské práce zaměřené na testování v metodikách nebo přímo na určitou metodiku vývoje softwaru. Na testování v metodice RUP se zaměřují bakalářské práce ***„Testování softwaru a metodika RUP“*** **[Kučera, 2006]** a ***„Metodika RUP a testování“*** **[Randová, 2007]**. Testování v metodice OpenUP je předmětem bakalářské práce ***„Testování softwaru v metodice OpenUP“*** **[Chodura, 2010].** Na metodiku RUP komplexně je potom zaměřena diplomová práce ***„Použití RUP pro malé SW projekty“*** **[Julinek, 2008]**. Přehled o celé metodice OpenUP je předmětem bakalářské ***„Metodika OpenUp“*** **[Kocura, 2010].** Zejména práce zaměřené na testování v metodikách RUP a OpenUP jsem využila při zpracování teoretické části, ve které jsou jednou z kapitol právě metodiky vývoje softwaru z pohledu testování.

Vytvoření nové metodiky vývoje softwaru MMSP je předmětem diplomové práce ***„Lokalizace a přizpůsobení metodiky OpenUP“*** **[Rejnková, 2011]**. Metodika MMSP byla vytvořena lokalizací a přizpůsobením metodiky OpenUP pro projekty probíhající v rámci výuky softwarového inženýrství na VŠE a je jí možné využít i pro menší projekty v praxi. Stejně jako metodika OpenUP pokrývá celý životní cyklus vývoje softwaru. Součástí metodiky jsou role, pracovní produkty, disciplíny a úlohy a životní cyklus. Navržená metodika čerpá primárně z metodiky OpenUP, ale i z jiných agilních metodik, odborných knih a článků, ostatních diplomových prací a webových stránek. Při vytváření metodiky v této práci jsem se inspirovala strukturou metodiky MMSP a využila i postupu vytváření metodiky v nástroji EPF Composer, který autorka v diplomové práci uvádí.

Ve vztahu k praktické části této diplomové práce jsou klíčové zejména diplomové práce, jejichž cílem je vytvoření metodiky testování. Přímo na vytváření metodiky testování se soustředí diplomové práce ***„Metodika testování webových aplikací“*** **[Šplíchalová, 2008]**, ***„Návrh metodiky testování webových aplikací“*** **[Fiurášek, 2010]** a *„Návrh metodiky zátěžového testování s nástrojem IBM Rational Performance tester“* **[Vomáčko, 2012]**. Metodiky navržené v těchto pracích jsou uzpůsobeny potřebám konkrétní společnosti nebo testovacího týmu a vychází z již existujících metodik vývoje softwaru, přičemž rozpracovávají oblast testování.

Cílem práce *„Metodika testování webových aplikací“* [Šplíchalová, 2008] je vytvoření metodiky pro menší testovací oddělení. Metodika je zaměřená na proces testování, role, aktivity a artefakty testování a vychází z přístupů metodik RUP a OpenUP. Zvláštní pozornost je věnována také chybám v softwaru a jejich reportování. Zdrojem pro vytváření metodiky byly zejména metodiky RUP a OpenUP, odborné knihy, interní materiály testovacího oddělení konkrétní firmy, webové stránky a praktické zkušenosti autorky. Metodika

je přizpůsobena pro potřeby testovacího oddělení konkrétní firmy a používá terminologii metodik RUP a OpenUP. Tato práce mi byla inspirací zejména při navrhování struktury pracovního produktu defekt report a dalších dílčích záležitostech.

Cílem práce *„Návrh testování webových aplikací“* [Fiurášek, 2010] je navržení metodiky testování webových aplikací pro menší softwarovou společnost. Metodika se svým zaměřením podobá metodice dle [Šplíchalová, 2008]. Je zaměřena na proces testování, role, artefakty a zvláštní pozornost je věnována i chybám, doporučením pro testování vstupních prvků a doporučením pro přípravu testovacích dat. Za přínosné považuji, že metodika pokrývá i role, jejichž předmětem zájmu není primárně testování, ale k oblasti testování se vztahují, jako je vedoucí projektu, vývojář a klient. Metodika vychází z principů metodiky OpenUP a zdrojem jsou zejména akademické práce, metodika OpenUP, odborné knihy, články a webové stránky. Při vyváření metodiky v této práci jsem z [Fiurášek, 2010] využila zejména problematiku přípravy testovacích dat.

Cílem práce *„Návrh metodiky zátěžového testování s nástrojem IBM Rational Performance tester“* [Vomáčko, 2012] je vytvořit metodiku zátěžového testování pro účely týmu studentů kompetenčního centra řízena kvality na VŠE s využitím nástroje IBM Rational Performance tester. Metodika vznikla rozšířením metodiky MMSP a rozšiřuje její prvky role, úlohy, pracovní produkty a životní cyklus o prvky specifické pro zátěžové testování. Zdrojem práce je metodika MMSP, odborné knihy a články, akademické práce, webové stránky i vlastní zkušenosti autora se zátěžovým testováním.

Další diplomové práce v souvislosti s problematikou testování jsou zaměřené na zvyšování zralosti procesů testování. Jedná se o práci ***„Zvyšování zralosti testovacích procesů v IT společnosti“*** **[Kučera, 2009]** a práci ***„Modely zlepšování procesů testování softwaru a zajištění kvality“*** **[Došek, 2012]**, kterou jsem využila zejména pro zpracování přílohy D a kapitoly 7 této práce, kde se o ni i více zmiňuji. Na nástroje pro podporu řízení testů, automatického funkčního a manuální testování a nástroje pro sledování chyb se zaměřuje diplomová práce ***„Nástroje na podporu testování“*** **[Faustová, 2009]**. Tuto práci jsem při návrhu vlastní metodiky využila pouze okrajově.

Při vytváření metodiky testování v nástroji pro správu metodik EPF Composer jsem také využila bakalářskou práci ***„Eclipse Process Framework Composer – nástroj na správu metodiky“*** **[Pospíšil, 2009]**, která je příručkou v českém jazyce pro začínající uživatele nástroje Eclipse Process Framework Composer.

### 1.6.2 Odborné knihy

Při zpracování diplomové práce jsem narazila na mnoho obsáhlých knih v angličtině, věnujících se oblasti testování softwaru.

Využila jsem například knihu ***„Managing the Testing Process“*** **[Black, 2002]**, která kompletně pokrývá proces testování od plánování testů, přes přípravu testů, řízení a orga-

nizaci testování, databázi chyb až po řízení testovacího týmu a zasazení testování do kontextu ekonomiky, životních cyklů, zralosti procesu a další záležitosti. V této knize jsem se inspirovala při rozpracování úlohy sestavení harmonogramu testování v navržené metodice a převzala jsem z ní a mírně upravila i životní cyklus defektu.

Dalším opěrným zdrojem byla pro mě kniha ***„Testing Computer Software“*** **[Kaner, 1993]**, ve které mě zaujala především kapitola Reportování a analýza chyb a vycházela jsem z ní při rozpracování úlohy analýza a reportování defektů.

Z knihy ***„The Art of Software Testing“*** **[Myers, 2004]**, která je v oboru testování rovněž uznávána a často citována, jsem čerpala inspiraci v kapitole o psychologii testování.

Užitečným zdrojem je také kniha ***„Software Testing Fundamentals: Methods and Metrics“*** **[Hutcheson, 2003]**, jejíž součástí je mimo jiné kapitola o základních metrikách pro testování softwaru, kterou jsem při vytváření metodiky využila testování při navrhování sady základních metrik pro testování.

Na problematiku testování komplexně se také zaměřuje velmi obsáhlá nedávno vydaná kniha ***„Software testing“*** **[Singh, 2012].**

V českém vydání jsou je dispozici kniha ***„Testování softwaru“*** **[Patton, 2002]**, kterou mohu doporučit jako první seznámení s oblastí testování. Kniha nejde příliš do hloubky, ale přesto obsahuje teorie, které jsou v testování dodnes uplatňovány. Obsahuje zejména vyčerpávající definici pojmu chyba, kterou uvádím i v navržené metodice v konceptu definice defektu a příbuzných termínů.

Obsáhlou knihou zaměřující se na zajišťování kvality softwaru je kniha ***„Software Quality Assurance – From Theory to Implementation“*** **[Galin, 2004],** ve které jsou mimo jiné zpracovány i některé standardy vztahující se k testování a zajišťování kvality softwaru, ale jelikož je z roku 2004, ještě se na rozdíl od této práce nezmiňuje o vývoji nového standardu ISO/IEC/IEEE 29119, který bude pokrývat celou oblast testování softwaru.

Výchozím zdrojem pro teorii standardů a metodik vývoje software byla kniha ***„Metodiky budování informačních systémů“*** **[Buchalcevová, 2009]**, ve které jsou přehledově zpracovány některé standardy, které mají vztah k testování a zajišťování kvality softwaru, je zde uvedena definice metodiky a prvky, které metodika obsahuje, kategorizace metodik a přehledové zpracování vybraných metodik spolu se systémem jejich hodnocení a výběru.

### 1.6.3 Odborné články

Východiskem pro kapitolu 3 teoretické části práce byl článek ***„Hodnocení metodik vývoje systémů z pohledu testování“*** **[Buchalcevová, 2008]**, ve kterém jsou z pohledu testování zhodnoceny tři metodiky vývoje softwaru. Podle sady kritérií hodnocení navržené v tomto článku jsem zhodnotila další dvě metodiky, kterými se článek nezabývá, viz příloha A.

Přínosem pro tuto práci byl i článek ***„ISTQB Certification: Why You Need It and How to Get It. Testing experience“*** **[Black, 2008]**, kde je mimo jiné diskutováno, jakou hodnotu přináší ISTQB certifikace nejen testerovi, ale i organizaci, která zaměstnává certifikovaného testera a samotné profesi testování.

Při rozpracování techniky Průzkumné testování v navržené metodice jsem kromě samotných materiálů ISTQB využila článku zaměřeným přímo na toto téma ***„Exploratory Testing Explained“*** **[Bach, 2003].**

V ACM library jsem nalezla zajímavý článek zaměřený na měření procesu testování ***„Effective Test Metrics for Strategy Evolution“*** **[Chen, 2004]**, ale vzhledem k tomu, že uvádí poměrné pokročilé metriky nad rámec této práce, pouze se na něj odkazuji v kapitole 6. 2. 1. 4. Článků o testování jsem v ACM library nalezla více, ale žádný další nebyl pro tuto práci relevantní.

# 2 Vymezení základních pojmů

V této kapitole jsou vymezeny základní pojmy, které jsou používány v textu této diplomové práce a které jsou důležité pro porozumění obsahu práce. Další pojmy, které se přímo vztahují k tématu práce, jsou definovány přímo v textu práce. Ostatní pojmy, které jsou v textu práce použity, ale není vhodné je v textu práce přímo definovat, jsou vysvětleny v poznámkách pod čarou. Souhrnný seznam termínů a zkratek, ve kterém jsou důležité pojmy a zkratky vysvětleny, je zařazen v závěru diplomové práce.

**Informační systém** (IS) podniku je systém je *„systém pro sběr, přenos, uchování, zpracování a poskytování dat (informací, znalostí) využívaných při činnosti podniku.“* Jedná se o systém informačních a komunikačních technologií, dat a lidí, jehož cílem je efektivně podporovat informační, rozhodovací a řídící procesy na všech úrovních řízení podniku. [Voříšek, 2008, s. 18] Ve spojení s informačními a komunikačními technologiemi se pro IS používá zkratka IS/ICT.

Informační systém je začleněn do organizační struktury podniku a využívá technického a programového vybavení neboli **software**. V kontextu vývoje softwaru se pro pojem software používá také pojem programový systém. **Programový systém** označuje softwarový produkt, *„který je tvořen množinou programových jednotek (modulů, komponent, objektů) a jejich vzájemných vazeb“*. [Buchalcevová, 2002, s. 135] V dalším textu této práce jsou používány pojmy software a systém jako synonyma, a systém je chápán jako programový systém.

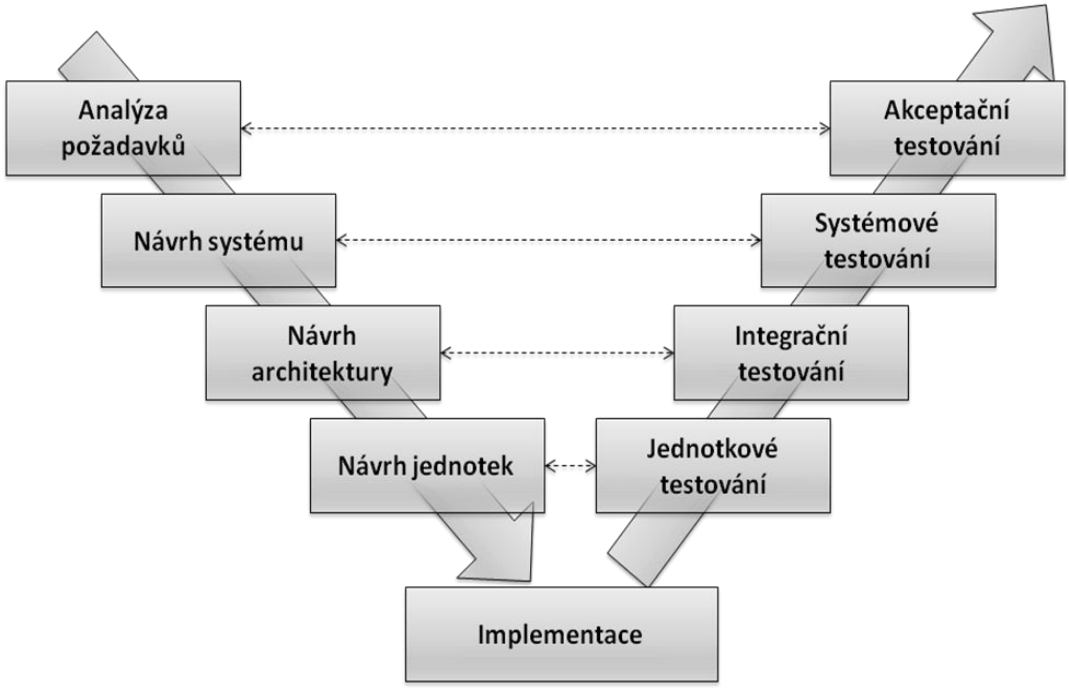
**Životní cyklus softwaru** je časové období, které začíná úmyslem vytvořit softwarový produkt a končí, když se software přestane nadále používat [ISTQB, 2012c, s. 39]. Životní cyklus vývoje softwaru je potom časové období, které začíná rozhodnutím vyvinout softwarový produkt a končí, jakmile je software dodán [IEEE, 1990, s. 67].

**Model životního cyklu** je rámec procesů a aktivit spojených s životním cyklem, které mohou být organizovány do stupňů. Model životního cyklu slouží také jako referenční rámec pro komunikaci a vzájemné dorozumívání. [ISO/IEC, 2008, s. 4] **Model životního cyklu softwaru** *„představuje rámec realizace procesů životního cyklu v časové posloupnosti“*. [Buchalcevová, 2009, s. 47] Mezi nejznámější modely životního cyklu patří dle [Buchalcevová, 2009, s. 47] model programuj a opravuj, vodopádový model, V-model, spirálový model, inkrementální model a evoluční model. Jejich charakteristiku uvádí [Buchalcevová, 2009, s. 47-54]. Pro účely této práce je nutné vymezit alespoň V-model.

**V-model** ilustruje, jak mohou být aktivity testování integrovány do každé fáze životního cyklu vývoje softwaru [ISTQB, 2012c, s. 49] Je zakreslen do tvaru písmene V, jak znázorňuje Obrázek č. 1.

V levé části probíhá shora dolů analýza požadavků, návrh systému, návrh architektury, návrh jednotek a implementace. V pravé části probíhá zdola nahoru testování,

kdy nejprve jsou testovány jednotky a následuje integrační testování, systémové testování a akceptační testování. [Buchalcevová, 2009, s. 49]




**Obrázek č. 1 V-model, zdroj: [Buchalcevová, 2009, s. 50]**

**Kvalita** je v nejširším pojetí definována jako *„stupeň splnění očekávání“*. Kvalitu systému, komponenty nebo procesu definuje [IEEE, 1990, s 60] jako *„stupeň naplnění specifikovaných požadavků“* a dále jako *„stupeň naplnění potřeb a očekávání zákazníka nebo uživatele“*. Na kvalitu se nejčastěji nahlíží jako na uspokojení požadavků a potřeb uživatelů, ale lze se na ni dívat i z jiných hledisek. David Garvin kvalitu vnímá z pěti různých pohledů. Kromě naplnění požadavků uživatelů, které nejsou u všech skupin uživatelů jednotné *(user based)*, je kvalita dále posuzována

- z hlediska splnění požadavků interních standardů a praktik pro vytváření produktu *(manufacturing based)*,
- z hlediska splnění objektivně stanovených, kvantifikovatelných a měřitelných charakteristických atributů daného produktu *(product based)*,
- z hlediska ochoty zákazníka produkt za danou cenu zakoupit *(value based)*,
- z hlediska intuice, což vystihují slova *„Neumím to definovat, ale vím to, když to vidím.“ (transcendental view)*. [Ross, 2009]

Pro účely této práce je kvalita chápána ze všech těchto úhlů pohledu.

**Zajišťování kvality softwaru** *(Quality Assurance, QA)* jsou plánované a systematické procesy, jejichž cílem je zajistit, aby byl software vhodný k jeho zamýšlenému účelu. [Havlíčková, 2009] Náplní zajišťování kvality softwaru je zkoumání a měření současného procesu vývo-

je softwaru a nalezení způsobů jeho zdokonalení za účelem zabránění vzniku defektů. [Patton, 2002, s. 70]

Pro testování softwaru existuje řada definic. Pro účely této práce je **testování softwaru** chápáno jako proces ověřování kvality softwaru (a případně i dalších součástí výsledného softwarového produktu) za účelem zjištění rozdílů mezi aktuálním a očekávaným chováním (resp. stavem), detekování případných defektů a poskytnutí informací o kvalitě softwaru a jeho součástí zainteresovaným stranám. Dle [ISTQB, 2012c, s. 47] se pod pojem testování řadí i statické techniky testování, jako jsou revize zdrojového kódu a pracovních produktů vznikajících při vývoji softwaru, nicméně v této práci je testování softwaru chápáno v dynamickém pojetí, tzn., ověřování kvality softwaru probíhá za běhu softwaru. Testování softwaru je součástí obecněji pojatého zajišťování kvality softwaru.

**Úroveň testování** je skupina testovacích aktivit, které jsou organizovány a řízeny dohromady a je vázána na zodpovědnosti v projektu. [ISTQB, 2012c, 45] Dle [ISTQB, 2011, s. 22] jsou definovány následující 4 úrovně testování:

- **Testování komponent**, známé také pod pojmem jednotkové testování *(Component testing, unit testing)* – testování samostatných softwarových komponent [ISTQB, 2012c, s. 13],
- **Integrační testování** *(Integration testing)* – testování vykonávané za účelem odhalení defektů na rozhraní a při interakci integrovaných komponent nebo systémů [ISQTB, 2012c, s. 26],
- **Systémové testování** *(System testing)* – proces testování celého integrovaného systému za účelem ověření, zda splňuje specifikované požadavky [ISTQB, 2012c, s. 42],
- **Akceptační testování** *(Acceptance testing)* – formální testování, při kterém jsou respektovány potřeby a požadavky uživatelů a podnikové procesy za účelem určení, zda systém splňuje akceptační kritéria a umožňuje uživatelům, zákazníkům nebo jiným oprávněným subjektům systém akceptovat [ISTQB, 2012c, s. 8].

# 3 Metodiky vývoje softwaru z pohledu testování

Testování softwaru je součástí procesu vývoje softwaru. Jelikož proces vývoje softwaru je velmi složitý, byly vytvořeny různé metodiky vývoje softwaru, které se snaží složitost řešit stanovením metod a postupů.

Metodika je tvořena *„uznávanými postupy a návody, které popisují činnosti při návrhu, vývoji, nasazování software, řízení projektu, atd. Cílem metodiky je formalizovat postupy, definovat zodpovědnosti a pravidla komunikace“* [Buchalcevová, 2002, s. 173].

**Metodika vývoje softwaru** definuje „*principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení*“ [Buchalcevová, 2009, s. 17][2]. Vývojem softwaru v tomto pojetí není myšlen pouze vývoj na „zelené louce“, ale také implementace již hotových řešení nebo integrace systémů, komponent a služeb [Buchalcevová, 2009, s. 17].

Metodik existuje v současnosti velké množství. Dle [Buchalcevová, 2009, s. 58] lze pro jejich kategorizaci využít následující kritéria: Zaměření metodiky, rozsah metodiky, váha metodiky, typ řešení, doména a přístup k řešení.

Podle váhy se rozlišují metodiky na lehké a těžké. **Váha metodiky** je dána součinem velikosti a hustoty metodiky. Velikost metodiky „*vyjadřuje počet kontrolních prvků obsažených v metodice“*. Hustota metodiky vyjadřuje „*míru podrobnosti a těsnost tolerance metodiky, požadovanou podrobnost a konzistenci prvků*“. [Buchalcevová, 2009, s. 62]

Propracované metodiky s přesně definovanými procesy, činnostmi a artefakty se označují jako **rigorózní**, neboli těžké metodiky. Předpokladem pro jejich zavedení je schopnost popsat softwarové procesy a předem definovat požadavky. Uplatnění nachází především na standardních a velkých projektech. [Buchalcevová, 2009, s. 70]

**Agilní**, neboli lehké metodiky, definují jen generativní pravidla, principy a praktiky. Předpokladem pro jejich zavedení je neschopnost popsat softwarové procesy a možnost předem definovat jen hrubé požadavky. Agilní metodiky je možné využít především na výzkumných projektech, u malých týmů a v okamžiku, kdy je prioritou uvést produkt na trh co nejrychleji *(time-to-market)*, nebo kde je potřeba rychle a pružně reagovat na měnící se požadavky. [Buchalcevová, 2009, s. 65, 70]

---

[2] V [Buchalcevová, 2009, s. 17] je uvedena definice metodiky budování IS/ICT, která *„definuje principy, procesy, praktiky, role techniky, nástroje a produkty používané při vývoji, údržbě a provozu informačního systému, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení“*. Z toho bylo odvozeno, že metodika vývoje softwaru je podskupinou metodik budování IS/ICT a nepokrývá tedy údržbu a provoz informačního systému.

V roce 2001 byl představiteli agilních přístupů k vývoji softwaru podepsán tzv. ***Manifest agilního vývoje softwaru***, který deklaruje hodnoty, na kterých je agilní vývoj postaven. Jeho česká verze zní následovně:

*„Objevujeme lepší způsoby vývoje software tím, že jej tvoříme a pomáháme při jeho tvorbě ostatním. Při této práci jsme dospěli k těmto hodnotám:*

- *Jednotlivci a interakce před procesy a nástroji,*
- *fungující software před vyčerpávající dokumentací,*
- *spolupráce se zákazníkem před vyjednáváním o smlouvě,*
- *reagování na změny před dodržováním plánu.*

*Jakkoliv jsou body napravo hodnotné, bodů nalevo si ceníme více.“* [Beck, 2001]

Přístup k testování je v různých metodikách odlišný. V rigorózních metodikách je testovací tým typicky oddělen od vývojového týmu, má vlastní pravomoci a zastává úlohu advokáta zákazníka proti vývojářům. V agilních metodikách jsou testeři součástí vývojového týmu, není zde test manažer, testování není odděleno od vývoje, a za vývoj i testování odpovídá tým jako celek [Doležel, 2013a].

Cílem této kapitoly je zhodnotit přístup k testování ve třech vybraných metodikách vývoje softwaru. Buchalcevová a Kučera v článku[3] ***„Hodnocení metodik vývoje systémů z pohledu testování“*** [Buchalcevová, 2008] definovali pro porovnání přístupu k testování následující sadu kritérií: Zaměření testování, proces testování, role, úrovně testování, trasovatelnost, životní cyklus vady, typy testů a testovací techniky, artefakty testování a užití nástrojů a automatizace.

Podle této sady kritérií následně zhodnotili metodiky Extrémní programování (XP), Rational Unified Process (RUP) a MSF for Agile Software Development.

Pro hodnocení přístupu k testování v této kapitole byly vybrány metodiky RUP, OpenUP a MMSP. Metodika RUP byla pro hodnocení vybrána, jelikož je velmi propracovaná, oblast testování plně pokrývá a je zástupcem ze skupiny rigorózních metodik. Podle metodiky RUP jsou navíc definovány role v praktické části práce, viz kapitola 6.4. Jelikož je tato metodika jednou z hodnocených metodik podle sady definovaných kritérií v uvedeném článku, v této kapitole uvádím pouze její stručnou charakteristiku a souhrnné hodnocení využitelnosti metodiky pro testování. Metodika OpenUP vznikla teprve v nedávné době (v roce 2006), je zástupcem agilních metodik a vychází z metodiky RUP. Oblast testování také pokrývá, ale ve srovnání s metodikou RUP pouze v malém rozsahu. Metodika MMSP byla vybrána, jelikož se jedná o novou (z roku 2011) českou metodiku, která vznikla lokalizací metodiky OpenUP. U metodik OpenUP a MMSP je uvedena nejen

[3] Terminologie této kapitoly je jednotná s terminologií použité v uvedeném článku. Článek používá termín vada, který je ekvivalentní s termínem defekt použitým ve zbylých částech této práce.

jejich stručná charakteristika, ale zároveň bylo provedeno i jejich hodnocení z pohledu testování podle definované sady kritérií. Účelem je rozšířit uvedený článek o hodnocení dalších dvou metodik z pohledu testování. Jelikož hodnocení podle kritérií jsou poměrně detailní, byly zařazeny do přílohy A. Stejně jako u metodiky RUP, i u metodik OpenUP a MMSP je uvedeno také souhrnné hodnocení využitelnosti metodik pro testování.

V roce 2008 byla publikována nová verze standardu *IEEE Standard for Software and System Test Documentation (IEEE 829)* [IEEE, 2008], což má dopad na hodnocení kritéria Artefakty testování. Článek je z roku 2008 a definuje dokumenty ještě podle verze standardu IEEE 829 z roku 1983. Podstata dokumentů zůstává stejná, nicméně v nové verzi standardu jsou některé názvy dokumentů odlišné. Dokumenty Test Plan a Test Summary Report jsou rozlišeny na Level Test Plan, resp. Level Test Report pro dokumentaci vztahující se pouze k jedné úrovni testování a Master Test Plan, resp. Master Test Report pro dokumenty skrze všechny úrovně testování. V této kapitole již jsou uváděny názvy dokumentů podle nové verze standardu IEEE 829 z roku 2008.

## 3.1 Testování v metodice RUP

Metodika Rational Unified Process (RUP) vznikla „*spojením přístupu Rational a metodiky Objectory Process Ivara Jacobsona*“ [Buchalcevová, 2009, s. 121]. Následně autoři Ivar Jacobson, Grady Booch a James Rumbaugh vytvořili její zobecněnou verzi Unified Process, kterou popsali v knize *„The Unified Software Development Process“* [Buchalcevová, 2009, s. 121]. Dle [Rational, 2011, s. 1] lze metodiku definovat jako softwarově inženýrský proces, který poskytuje disciplinovaný přístup k přiřazování úloh a zodpovědností v organizaci, zaměřenou na vývoj softwaru. Jejím cílem je zajistit vysokou kvalitu softwaru, kdy jsou uspokojeny potřeby koncových uživatelů v rámci stanoveného harmonogramu a rozpočtu.

Metodika je založena na následujících **6 praktikách** softwarového vývoje [Rational, 2011, s. 1-2]:

- Iterativní vývoj softwaru *(develop software iteratively)*,
- řízení požadavků *(manage requirements)*,
- použití komponentových architektur *(use component-based architectures)*,
- vizuální modelování softwaru *(visually model software)*,
- ověřování kvality softwaru *(verify software quality)*,
- řízení změn v softwaru *(control changes to software)*.

Dříve byla metodika RUP zařazována do skupiny rigorózních metodik, ale postupně byla doplňována o agilní principy a praktiky. V současnosti je rámcem pro definování metodik pro malé i velké projekty [Buchalcevová, 2008, s. 5].

Metodika RUP je v současnosti poskytována pouze komerčně a je součástí produktu IBM Rational Composer (RMC), kde lze metodiku nakonfigurovat pro různé účely použití. Metodika je také integrována s nástroji IBM Rational.

Přehlednou charakteristiku metodiky RUP a její struktury lze nalézt v [Buchalcevová, 2009, s. 121-127], popř. v [Buchalcevová, 2002, s. 180-183]. Podrobněji se metodikou RUP zabývá např. diplomová práce na téma „*Použití RUP pro malé SW projekty*“ [Julinek, 2008]. Speciálně na oblast testování v metodice RUP jsou zaměřeny bakalářské práce „*Testování softwaru a metodika RUP*“ [Kučera, 2006] a „*Metodika RUP a testování*“ [Randová, 2007].

Metodika RUP je jednou ze tří metodik hodnocených podle stanovených kritérií v článku „*Hodnocení metodik vývoje informačních systémů z pohledu testování*“ [Buchalcevová, 2008].

### 3.1.1 Souhrnné hodnocení využitelnosti metodiky pro testování

Metodika RUP je využitelná především pro velké projektové týmy, kde každý člen týmu má jasně přiřazené odpovědnosti. Metodika je velmi rozsáhlá. Poskytuje vzory procesů, množství praktik, rolí, návodů, šablon a dalších součástí. Před vlastním využíváním metodiky na projektech v praxi je nutná její konfigurace na míru projektu, organizace a cílové oblasti. Slouží tedy jako rámec, ze kterého se vybere jen potřebná množina prvků metodiky. Je možné metodiku nakonfigurovat pro velké i malé projekty.

Testování je v metodice RUP samostatnou disciplínou, obdobně jako např. implementace nebo projektový management. Testeři mají vlastní testovací tým v rámci projektového týmu, který je řízen manažerem testování. Pokud se v průběhu testování vyskytnou nějaké problémy, testeři mají možnost je eskalovat na test manažera, který může provést nápravné opatření buď přímo sám, nebo problém eskalovat dále na projektového manažera. Test manažer zastává v řešitelském týmu roli advokáta kvality, jedná v zájmu členů testovacího týmu a jako manažer skupiny lidí budí při vyjednávání s projektovým manažerem a ostatními členy projektového týmu větší respekt než jednotliví členové testovacího týmu samostatně.

Artefakty testování v podobě dokumentace svým rozsahem pokrývají většinu standardu pro testovací dokumentaci IEEE 829. Kromě samotného vytváření těchto artefaktů metodika zavádí i *„proces údržby a zkvalitnění testovací dokumentace, testovacích dat, vytvořených testů a dalších náležitostí testování (v metodice souhrnně označených jako Test Assets)“* [Buchalcevová, 2008, s. 5]. Vytvořené artefakty lze potom znovupoužít při následujících projektech.

Metodika RUP je postavena na nejlepších praktikách a konkrétně pro disciplínu testování je důležité zejména nepřetržité zaměření na kvalitu *(focus continuously on quality)* a předkládání výsledků v iteracích *(demonstrate value iteratively)* [RUP, 2010].

Za zajištění vysoké kvality produktu zodpovídají všichni členové projektového týmu společně. Testovací tým má za úkol pouze ověřování kvality softwaru, díky kterému může nasměrovat zbytek týmu k dosažení softwarové kvality. Nenese ovšem přímou odpovědnost za zajištění kvality.

Výhodou iterativního vývoje je, že testování je zapojeno do procesu vývoje softwaru prakticky již od začátku projektu a snižuje se tak riziko nutnosti opravy vad v pozdních fázích projektu, což vyžaduje mnohonásobně větší náklady a úsilí, než na počátku. Při iterativním vývoji je typicky nalezená vada záležitostí posledního přírůstku a její oprava vyžaduje zejména porovnání změn oproti předchozímu přírůstku. Iterativní vývoj klade nároky na neustálé testování nových přírůstků. Testeři proto musí být zběhlí v automatizaci testů, aby stíhali nejen zapracovat a provést testy na novou funkcionalitu, ale také regresně otestovat funkcionality implementované v předchozích sestaveních *(buildech)*.

Metodika obsahuje koncept, ve kterém jsou popsány výhody a nevýhody automatizace testů a kde je upozorněno na možné problémy a rizika automatizace [Buchalcevová, 2008, s. 6]. Výhodou metodiky RUP je přímá integrace s mnoha nástroji pro podporu automatizace a řízení testování. Tyto nástroje rodiny IBM Rational jsou ovšem stejně jako metodika komerční a nejsou prodávány v jednom balíku s metodikou. Ceny nástrojů se pohybují v tisících až desetitisících dolarů za licenci k jednomu nástroji. Tyto nástroje mohou být integrovány i navzájem mezi sebou a poskytují oproti nekomerčním nástrojům široké množství funkcí.

Při rozhodování o využití metodiky pro testování je pro mnoho firem často problémem právě cenová dostupnost metodiky RUP a nástrojů, které ji podporují. Obzvláště v menších firmách je vzhledem k vysoké ceně metodika RUP spolu s nástroji nedostupná. Při konfiguraci metodiky lze metodiku navázat i na jiné nástroje, ale vzhledem k robustnosti metodiky může být tento proces obtížný a zdlouhavý. Ani ve velkých firmách ale není nasazení metodiky jednoduché. Existuje sice řada pluginů metodiky pro konkrétní typy projektů označované jako *„out-of-the-box“* [Kroll, 2007], ale i tato řešení stále obsahují mnoho prvků a vazeb a je nutné se v nich zorientovat a přizpůsobit je konkrétní organizaci a projektu.

Oblast testování je metodikou RUP plně pokryta. Principy a praktiky RUP vycházejí z mnohaletých zkušeností světových odborníků zabývajících se softwarovým inženýrstvím, včetně testování. Metodiku ale není vhodné nasazovat na samotné testování, aniž by byly dotčeny ostatní disciplíny procesu vývoje softwaru. Disciplína testování je pouze jeden článek v procesu vývoje softwaru, má vazby na ostatní disciplíny, a pokud by nebyly metodicky pokryty i ostatní disciplíny, samotné metodicky zvládnuté testování nemůže zajistit kvalitní produkt.

## 3.2 Testování v metodice OpenUP

Metodika Open Unified Process (OpenUP) vznikla zeštíhlením metodiky Unified Process. Lze ji charakterizovat jako minimálně dostatečnou, kompletní metodiku pro vývoj softwaru, kterou je možné přizpůsobit a rozšířit podle potřeb konkrétní organizace nebo projektu. Stejně jako Unified Process, je založena na iterativním a inkrementálním modelu životního cyklu, případech užití, řízení rizik a architektuře. [Buchalcevová, 2009, s. 128]

OpenUP je postavena na **4 základních principech**, které korespondují s hodnotami definovanými v Manifestu agilního vývoje. Principy OpenUP ve vazbě na hodnoty v manifestu agilního vývoje zachycuje Tabulka č. 1.

**Tabulka č. 1 Srovnání principů OpenUP a Manifestu agilního vývoje, zdroj: [Balduino, 2007, s. 2]**

| Principy OpenUP | Manifest agilního vývoje |
|---|---|
| Držet v rovnováze protichůdné priority s cílem maximalizovat hodnotu pro zákazníka. | Spolupráce se zákazníkem před vyjednáváním o smlouvě. |
| Spolupracovat za účelem sladění zájmů a sdílení znalostí. | Jednotlivci a interakce před procesy a nástroji. |
| Včas se zaměřit na architekturu s cílem minimalizace rizik a organizace vývoje. | Fungující software před vyčerpávající dokumentací. |
| Rozvíjet kontinuální získávání zpětné vazby a zlepšování. | Reagování na změny před dodržováním plánu. |

Tvůrci OpenUP zařazují metodiku do skupiny agilních metodik. Přestože metodika OpenUP staví na principech agilního vývoje, je třeba si uvědomit, že zároveň vychází z metodiky Unified Process, která je původem rigorózní. Proto by OpenUP neměla být chápána jako metodika ryze agilní, ale jako kompromis mezi tradičními a agilními metodikami. Sjednocuje nejlepší praktiky pocházející z obou těchto skupin metodik. [Rejnková, 2011, s. 21]

Metodiku je možné dále upravovat a rozšiřovat v nástroji Eclipse Process Framework Composer (EPF Composer). Na rozdíl od metodiky RUP je OpenUP volně dostupná v podobě webové prezentace[4] publikované pomocí nástroje EPF Composer.

Přehledný popis základních principů, organizace a prvků metodiky lze nalézt např. v [Buchalcevová, 2009, s. 127-135]. Podrobněji je v českém jazyce metodika OpenUP zpracována také v bakalářské práci přímo na téma *„Metodika OpenUp“* [Kocura, 2010]. Speciálně na oblast testování dle metodiky OpenUP je zaměřena bakalářská práce *„Testování softwaru v metodice OpenUP“* [Chodura, 2010]. Při hodnocení metodiky OpenUP

[4] Metodika OpenUP je dostupná na stránce http://epf.eclipse.org/wikis/openup/.

z pohledu testování vedle samotné webové prezentace metodiky [OpenUP, 2012] proto těžím také z této práce [Chodura, 2010]. Hodnocení metodiky podle stanovených kritérií je zařazeno v příloze A.1.

### 3.2.1 Souhrnné hodnocení využitelnosti metodiky pro testování

Metodika je využitelná pro malé projektové týmy, které pracují na jednom místě a mohou snadno vzájemně sdílet informace. Zákazník je týmu k dispozici a může postupně zpřesňovat své požadavky. Testeři jsou součástí vývojového týmu a za kvalitu výsledného produktu zodpovídá tým jako celek.

Předpokladem pro úspěšné zavedení metodiky je zájem o vyvinutí kvalitního produktu ze strany všech členů projektového týmu. Testeři pomáhají vývojářům včas odhalit vady v softwaru a vývojáři nalezené vady následně ihned opravují. Problém nastává, pokud testeři shledají vyvinutý software jako nekvalitní a vývojáři ho považují za kvalitní. Nejsou zde žádné eskalační mechanismy, které by vývojáře donutily vady softwaru opravit podle požadavků testera. Projektový manažer se typicky přikloní k názoru většiny členů týmu a pohnutky testerů nemusí být uspokojeny. Testeři jsou sice v pozici zástupce zájmů zákazníka, ale zároveň musí dobře vycházet s ostatními členy týmu. Vývojový tým proto musí vnímat nálezy vad jako pomoc pro naplnění požadavků zákazníka a ne jako kritiku jejich práce, což vyžaduje určitou dávku velkorysosti.

Typickým rysem této metodiky je také neustálá interakce se zákazníkem. Při neshodách v týmu je proto třeba se obrátit na zákazníka, který upřesní, co přesně požaduje a podle toho se tým rozhodne, jak dále postupovat.

Obdobně jako u metodiky RUP, i metodika OpenUP je založena na iterativním vývoji, se kterým jsou spojeny nároky na automatizaci testů. Výhodou je zejména brzké zapojení testování do procesu vývoje softwaru a průběžné testování v rámci každé iterace.

Artefakty testování jsou omezeny na nezbytné minimum, což projektový tým ochuzuje o možnost čerpání získaných zkušeností na následujících projektech.

Oproti metodice RUP je OpenUP volně k dispozici veřejnosti i firmám pod open-source licencí, což je její obrovskou výhodou.

Z výše popsaných rysů metodiky OpenUP je patrné, že metodiku není vhodné nasazovat na oblast testování samostatně, aniž by byla nasazena i na ostatní oblasti procesu vývoje softwaru. Metodika OpenUP vychází z komplexní metodiky RUP, která vzešla z mnohaletých zkušeností světových odborníků zabývajících se softwarovým inženýrstvím včetně testování. Je proto využitelná nejen ve firmách, ale i k akademickým účelům jako pomůcka pro výuku kurzů softwarového inženýrství, včetně řízení kvality softwaru a testování.

## 3.3 Testování v metodice MMSP

Metodika MMSP (Metodika pro Malé Softwarové Projekty) je českou metodiku, která byla vyvinuta na půdě Katedry informačních technologií Vysoké školy ekonomické v Praze (VŠE). Vznikla lokalizací a přizpůsobením metodiky OpenUP především pro účely projektů probíhajících v rámci výuky softwarového inženýrství na VŠE, ale může být využívána i na reálných projektech v praxi [Rejnková, 2011, s. 26]. Jedná se o rozšiřitelnou metodiku založenou na iterativním a inkrementálním vývoji. Je určena především pro projekty s délkou do 6 měsíců, s velikostí do 10 členů týmu a pro projekty, resp. vyvíjené systémy, které nejsou extra kritické. [Rejnková, 2011, s. 34-35].

Metodika MMSP respektuje následující **principy agilního modelování** [Rejnková, 2011, s. 35]:

- Aktivní účast zainteresovaných stran na projektu,
- návrh architektury řešení,
- nepřetržitá dokumentace,
- dokumentovat co nejpozději,
- iterativní modelování,
- dostatečně dobré artefakty,
- více modelů,
- požadavky dle priorit,
- předvídání požadavků.

MMSP se řadí do skupiny agilních metodik. Kromě metodiky OpenUP je inspirována i dalšími agilními metodikami a praktikami.

Metodika byla vytvořena v nástroji EPF Composer, kde je možné ji dále upravovat a rozšiřovat. Její publikace v podobě webové prezentace[5] je volně dostupná k prohlížení veřejnosti. Komplexní popis celé metodiky je zpracován v diplomové práci na téma *„Lokalizace a přizpůsobení metodiky OpenUP“* [Rejnková, 2011], jejíž autorka je tvůrcem vlastní metodiky. Hodnocení metodiky podle stanovených kritérií je zařazeno v příloze A.2.

[5] Metodika MMSP je dostupná na stránce http://mmsp.czweb.org/ .

### 3.3.1 Souhrnné hodnocení využitelnosti metodiky pro testování

Metodika MMSP je využitelná pro malé projektové týmy do 10 členů a na nekritických projektech s délkou do 6 měsíců. Těmto podmínkám vyhovují zejména projekty probíhající v rámci výuky softwarového inženýrství. Testeři jsou součástí vývojového týmu a za kvalitu výsledného produktu zodpovídá celý tým, stejně jako v metodice OpenUP.

Jelikož MMSP vznikla lokalizací a rozšířením OpenUP, využitelnost metodiky MMSP pro testování a předpoklady pro úspěšné zavedení metodiky jsou obdobné jako u metodiky OpenUP.

Oproti OpenUP je disciplína testování rozšířena zejména o úlohu plánování testů, která je zařazena na začátku každé iterace spolu s plánováním celé iterace. Tím je usnadněna navazující příprava a provedení testů, protože je definováno, která část systému bude testována, jak se k testování bude přistupovat a které osoby budou za testování v rámci iterace zodpovědné.

Artefakty testování jsou v návaznosti na plánování testů rozšířeny o plán testů a pro přípravu testů jsou specifikovány seznam testovacích nápadů, testovací data, testovací případy a testovací sady. V OpenUP jsou pro přípravu testů specifikovány pouze testovací případy a testovací skripty. Navržené šablony pro jednotlivé artefakty testování jsou v MMSP přehledné a obsahují pouze nejdůležitější informace, takže se v nich snadno zorientují i studenti, kteří s testováním nemají zkušenosti.

Nepříliš vhodnou změnou oproti OpenUP je podle mého názoru návrh artefaktu Záznam výsledků testů, který slučuje informace o provedených testech a jejich výsledcích a podrobné reporty vad. Při provádění testů jsou výsledky testů průběžně zaznamenávány přímo ke krokům testovacích případů, z nichž vedou odkazy na reporty vad. Reporty vad ovšem v MMSP nejsou samostatným artefaktem, ale jsou zanořeny do artefaktu Záznam výsledků testů, který se kompletuje až na konci iterace. Aby vývojáři mohli nalezené vady opravovat ihned, bylo by lepší evidovat Reporty vad zvlášť a v artefaktu Záznam výsledků testů by bylo pouze shrnuto, jaké vady byly nalezeny a případně zdali již byly odstraněny.

MMSP je stejně jako metodika OpenUP volně dostupná a rozšiřitelná. Jedním z největších pozitiv je, že se jedná o českou metodiku. Zejména začínající testeři, kteří si potřebují rychle osvojit základy testování, mohou využít metodiku MMSP jako startovací bod.

## 3.4 Shrnutí kapitoly

V úvodu této kapitoly byla definována metodika vývoje softwaru a následně byly metodiky vývoje softwaru rozděleny podle váhy na rigorózní a agilní. Následně byla věnována pozornost třem vybraným metodikám vývoje softwaru, ve kterých je pokryta oblast testování. Cílem bylo zhodnotit přístup k testování ve vybraných metodikách RUP, OpenUP a MMSP. Každá z metodik byla nejprve stručně charakterizována. Pro hodnocení

metodik OpenUP a MMSP byla aplikována sada kritérií definovaná v článku *„Hodnocení metodik vývoje systémů z pohledu testování“* [Buchalcevová, 2008]. Metodika RUP je podle této sady kritérií zhodnocena přímo v uvedeném článku, tudíž není předmětem detailního hodnocení dle stanovených kritérií v této práci. Následně byly všechny tři metodiky souhrnně zhodnoceny z pohledu využitelnosti pro testování.

Všechny tři metodiky jsou koncipovány pro iterativní vývoj. Metodika RUP je velmi robustní a oblast testování plně pokrývá. Je ovšem poskytována pouze na komerční bázi. Metodiky OpenUP a MMSP odlehčují metodiku RUP a oblast testování nepokrývají zcela jako metodika RUP. Jsou ovšem na rozdíl od metodiky RUP volně dostupné. Metodika MMSP je na rozdíl od RUP a OpenUP v českém jazyce, což ji zvýhodňuje při zavedení v prostředí českých týmů.

# 4 Standardy ve vztahu k testování softwaru

Při organizaci procesu testování a zavádění metodiky testování není příliš efektivní vytvářet postupy, procesy a principy nově, „na zelené louce“, bez vědomí existujících znalostí v oboru testování. Ačkoliv každá organizace a projekt mají svá specifika, je vhodnější přizpůsobit obecné postupy podle specifických rysů organizace, než vynalézat znovu něco, co už bylo vynalezeno a osvědčilo se v odvětví testování. Jedním ze zdrojů znalostí skupiny profesionálů jsou kromě metodik také standardy. Na rozdíl od metodik, které popisují konkrétní návody „jak něco udělat“, standardy deklarují „co má být uděláno“. Dle [Buchalcevová, 2009, s. 57] představují standardy součást metodiky a metodika určuje, *„jaké standardy se budou používat, případně je může upravit*“. Metodikám tedy standardy slouží jako vodítka „čím se má metodika zabývat“.

Dle [Buchalcevová, 2009, s. 57] jsou standardy definovány jako *„publikované dokumenty na základě dohod. Zahrnují technické specifikace nebo jiná přesná kritéria, která se důsledně uplatňují jako pravidla, směrnice nebo charakteristické definice a zaručují, že materiály, produkty, procesy a služby splní své poslání.“*

K oblasti testování a zajišťování kvality softwaru se vztahuje mnoho standardů, které lze rozčlenit podle původu na mezinárodní, národní a doménově specifické [ISTQB, 2012a, s. 44].

Mezinárodní standardy jsou vydávány organizací ISO *(the International Organization for Standardization)* ve spojení s IEC *(the International Electrotechnical Commission)* a organizací IEEE *(the Institute of Electrical and Electronics Engineers)*. Těmto skupinám standardů jsou věnovány samostatné podkapitoly 4.1 a 4.2.

Mnoho zemí si vytvořilo své vlastní národní standardy, z nichž některé se vztahují i na oblast testování softwaru. Příkladem standardů, které jsou původem národní, ale rozšířily se do více zemí, jsou standardy Velké Británie publikované organizací BCS SIGiST *(British Computer Society Specialist Interest Group in Software Testing)*. Konkrétně se jedná o terminologický slovník pro testování softwaru s označením BS 7925-1 *Vocabulary of terms in software testing* a standard pro testování komponent softwaru s označením BS 7925-2 *Software component testing*. [The Testing Standards Working Party, 2005]

Do národních standardů se dále řadí také standardy, které vznikly lokalizací mezinárodních standardů. V České republice jsou standardy vydávány organizací ÚNMZ *(Úřad pro technickou normalizaci, metrologii a státní zkušebnictví)* pod označením ČSN *(české technické normy)*. V češtině jsou tak dostupné například normy ISO/IEC 90003 pod označením ČSN EN ISO/IEC 90003 nebo ISO/IEC 12207 pod označením ČSN ISO/IEC 12207. Tyto normy jsou charakterizovány v podkapitolách 4.1.1, 4.1.2.

Doménově specifické standardy vznikly buď vývojem přímo pro specifickou oblast nebo přizpůsobením mezinárodních nebo národních standardů pro určitou oblast. Některé z těchto doménově specifických standardů mají dopad i na oblast testování softwaru, jako například standard organizace U.S. Federal Aviation Administration s označením DO-178B *Software Considerations in Airborne Systems and Equipment Certification*, uplatňovaný v letectví, nebo standard organizace U.S. Food and Drug Administration s označením Title 21 CFR Part 820 *Quality System Regulation* pro lékařské systémy. [ISTQB, 2012a, s. 44]

V některých případech je shoda se standardy nejen doporučována, ale dokonce přímo vyžadována. [ISTQB, 2012a, s. 45] Přísná pravidla platí pro testování kritických systémů, kde jsou ohroženy například lidské životy, životní prostředí, nebo hrozí významné ekonomické ztráty. Test manažer by měl vědět, které standardy se na jeho oblast vztahují, jaký mají dopad na testování a zda jsou povinné, nebo jen doporučované.

V následujících podkapitolách se zaměřuji na mezinárodní standardy ISO/IEC a IEEE, které jsou relevantní pro testování a zajišťování kvality softwaru. Cílem je uvést přehled mezinárodních standardů a představit nejdůležitější z nich. Jelikož většina pracovních produktů v praktické části práce, kde navrhuji metodiku testování, vychází ze standardu IEEE 829, je tomuto standardu věnována větší pozornost než ostatním.

## 4.1 Standardy ISO/IEC

Organizace The International Organization for Standardization (ISO) [ISO, 2013] je tvůrcem nezávislých mezinárodních standardů pokrývajících téměř všechny aspekty technologií a byznysu od bezpečnosti potravin, přes počítače, zemědělství až po zdravotní péči. Účelem ISO standardů je pomáhat jednotlivým odvětvím k účinnosti a efektivitě prostřednictvím norem pro produkty a služby a specifikací osvědčených postupů. Organizace ISO je sítí národních normalizačních orgánů, které zastupují své země při vytváření ISO standardů a zároveň reprezentují ve svých zemích organizaci ISO. [ISO, 2013]

V roce 1987 byla založena technická komise ISO/IEC JTC 1, která vznikla spojením organizací ISO a IEC *(the International Electrotechnical Commission)* a soustředí se na vytváření světových standardů v oblasti informačních a komunikačních technologií. [ISO/IEC, 2008, s. 4] Některé standardy ISO/IEC jsou využitelné i pro softwarové testery. Jedná se zejména o standardy ISO 9000, včetně směrnice ISO/IEC 90003, ISO/IEC 12207 a řadu ISO/IEC SQuaRE. Ve schvalovacím řízení je v současné době také nový standard ISO/IEC/IEEE 29119, který se bude vztahovat na celou oblast testování softwaru.

### 4.1.1 Standardy řady ISO 9000

Standardy řady ISO 9000 se zaměřují na řízení kvality. Rodina standardů ISO 9000 byla přeložena do češtiny a vydána v podobě českých technických norem ČSN EN ISO

9000. V České republice patří tyto normy k nejznámějším a nejčastěji zaváděným [Buchalcevová, 2009, s. 43].

Základním standardem je ISO 9001:2008 *Quality management systems – Requirements*[6], který specifikuje požadavky na systém řízení kvality pro jakoukoli organizaci nezávisle na velikosti a odvětví. [ISO, 2013] Pouze podle požadavků deklarovaných tímto standardem se provádí certifikace organizací, čímž dle [Buchalcevová, 2009, s. 44] organizace prokazuje *„schopnost trvale poskytovat výrobek, který splňuje požadavky zákazníka a příslušné požadavky předpisů, zvyšovat spokojenost zákazníka a neustále zlepšovat své procesy“*. Dle této normy musí organizace *„vytvořit, dokumentovat, uplatňovat a udržovat systém managementu jakosti a neustále zlepšovat jeho efektivnost“* [ČNI, 2005, s. 11]. Tímto ovšem ještě není zaručeno, že daný systém řízení kvality navržený samotnou organizací je efektivní. Důraz je kladen především na precizní dokumentaci, nikoliv na vyspělost a efektivitu procesů řízení kvality.

Standard ISO 9001:2008 je dále doplněn standardy ISO 9000:2005 *Quality management systems – Fundamentals and vocabulary*[7], kde jsou uvedeny základní principy řízení kvality a terminologie, a ISO 9004:2009 *Managing for the sustained success of an organization – A quality management approach* [8], který poskytuje návody na zlepšování systému řízení kvality.

Pro organizace zabývající se počítačovým softwarem je významná zejména směrnice ISO/IEC 90003:2004 – *Guidelines for the application of ISO 9001:2000 to computer software*[9]. Tato norma je aktuálně platnou směrnicí, která obsahuje návod pro organizace při použití normy ISO 9001:2000 při akvizici, dodání, provozování a údržbě počítačového softwaru. Je nezávislá na technologii, procesech vývoje, modelech životního cyklu, posloupnosti činností a organizační struktuře a identifikuje problémy, kterými se má organizace zabývat. [ČNI, 2005, s. 7] Postupně opakuje klausule normy ISO 9001 a v zápětí vysvětluje, jak požadavky obecné normy ISO 9001 aplikovat na oblast softwaru.

Na oblast testování a zajišťování kvality softwaru se vztahují kapitoly 7.3.4 – Přezkoumání návrhu a vývoje, 7.3.5 – Ověřování návrhu a vývoje a 7.3.6 – Validace návrhu a vývoje. Validace softwaru má za účel „*poskytnout přijatelnou důvěru v to, že software bude splňovat požadavky na provozování*“ [ČNI, 2005, s. 30]. Testování softwaru je častou

---

[6] V České republice je dostupný pod označením ČSN EN ISO 9001:2008 *Systémy managementu kvality – Požadavky*.

[7] V České republice je dostupný pod označením ČSN EN ISO 9000:2005 *Systémy managementu kvality – Základní principy a slovník.*

[8] V České republice je dostupný pod označením ČSN EN ISO 9004:2009 *Řízení udržitelného úspěchu organizace – Přístup managementu kvality*.

[9] Směrnice se váže k verzi ISO 9001 z roku 2000. K aktuální verzi ISO 9001 z roku 2008 zatím nebyla tato směrnice vydána. V České republice je dostupná pod označením ČSN EN ISO/IEC 90003:2004 *Softwarové inženýrství – Směrnice pro použití ISO 9001:2000 na počítačový software*.

metodou validace softwaru a podle této normy může být prováděno následujících čtyřech úrovních [ČNI, 2005, s. 30]:

- Jednotkové testy[10], tj. testy samostatných softwarových komponent,
- integrační a systémové testy, tj. testy agregací softwarových komponent a kompletního systému,
- kvalifikační testy[11], tj. testy pro ověření, že dokončený softwarového produkt splňuje stanované požadavky před vlastním dodáním,
- akceptační testy, tj. testy, kdy zákazník ověřuje, zda výsledný software splňuje akceptační kritéria.

Mimo tyto čtyři úrovně je zde zmíněno také regresní testování, které slouží pro ověření, že způsobilost softwaru nebyla narušena změnami.

Norma nepopisuje, jak má být proces testování navržen. Uvádí pouze seznam bodů, které má organizace při plánování a provádění testů vzít v úvahu.

Tato norma se v mnoha místech odkazuje na standard ISO/IEC 12207, který definuje procesy životního cyklu softwaru a na ISO/IEC TR 15271, přílohu C, která obsahuje návod pro použití procesů z ISO/IEC 12207 v různých modelech životního cyklu [ČNI, 2005, s. 12]. Zároveň upozorňuje na možné nekonzistence těchto norem s normami řady ISO 9000.

### 4.1.2 Standard ISO/IEC 12207

Standard ISO/IEC 12207:2008 *Systems and software engineering – Software life cycle processes*[12] stanovuje rámec pro procesy v životním cyklu softwaru, spolu s terminologií, kterou lze uplatňovat v softwarovém odvětví. Obsahuje procesy, činnosti a úkoly, které mají být vykovány při pořízení softwarového produktu nebo služby a během dodávky, vývoje, provozu, údržby a odstranění softwarového produktu. [ISO/IEC, 2008, s. 1]

Tento standard lze aplikovat a přizpůsobit pro potřeby různých organizací a projektů. Nepředepisuje konkrétní model životního cyklu softwaru, metodiku vývoje, metodu, ani techniku. Poskytuje pouze návody na to, kam by měly jednotlivé procesy životního cyklu softwaru směřovat. Výběr modelu životního cyklu, včetně dalších specifik projektu a jejich mapování na procesy, aktivity a úkoly uvedenými v tomto standardu jsou v zodpovědnosti

[10] V české verzi normy je pojem „unit tests“ nesprávně přeložen jako „jednotka testování“ místo jako „jednotkové testy“.

[11] Úroveň kvalifikačních testů je obdobou **systémových testů** definovaných v kapitole 2.

[12] V České republice je dostupný pod označením ČSN ISO/IEC 12207 *Informační technologie – Procesy v životním cyklu softwaru* a vztahuje se na verzi ISO/IEC 12207 z roku 1995. Verze ISO/IEC 12207 z roku 2008 ještě přeložena do češtiny nebyla.

organizace, která aplikuje tento standard. Ani detailní dokumentaci procesů standard nepokrývá, ale alespoň se v tomto ohledu odkazuje na standard ISO/IEC 15289 *Systems and software engineering – Content of life cycle information products (documentation)*. [ISO/IEC, 2008, s. 1-2]

Podle [Galin, 2004, s. 513] je standard ISO/IEC 12207 konzistentní s následujícími koncepty TQM *(Total Quality Management)*:

- Kvalita je integrální součástí každého softwarového procesu.
- Každý proces má v sobě zabudované složky kvality, které jsou uplatňovány týmy zodpovědnými za jednotlivé fáze procesu.
- Procesy zajišťování kvality softwaru *(Software Quality Assurance)* jsou cíleny na dosažení souladu s požadavky na kvalitu, ale zároveň dávají organizaci volnost, jak kvality dosáhnout.

Na rozdíl od ISO 9001, standard ISO/IEC 12207 nespecifikuje žádné požadavky na certifikace a i když jsou procesy softwarové organizace v souladu s tímto standardem, není možné získat certifikát, který by soulad se standardem potvrdil. [Galin, 2004, s. 513]

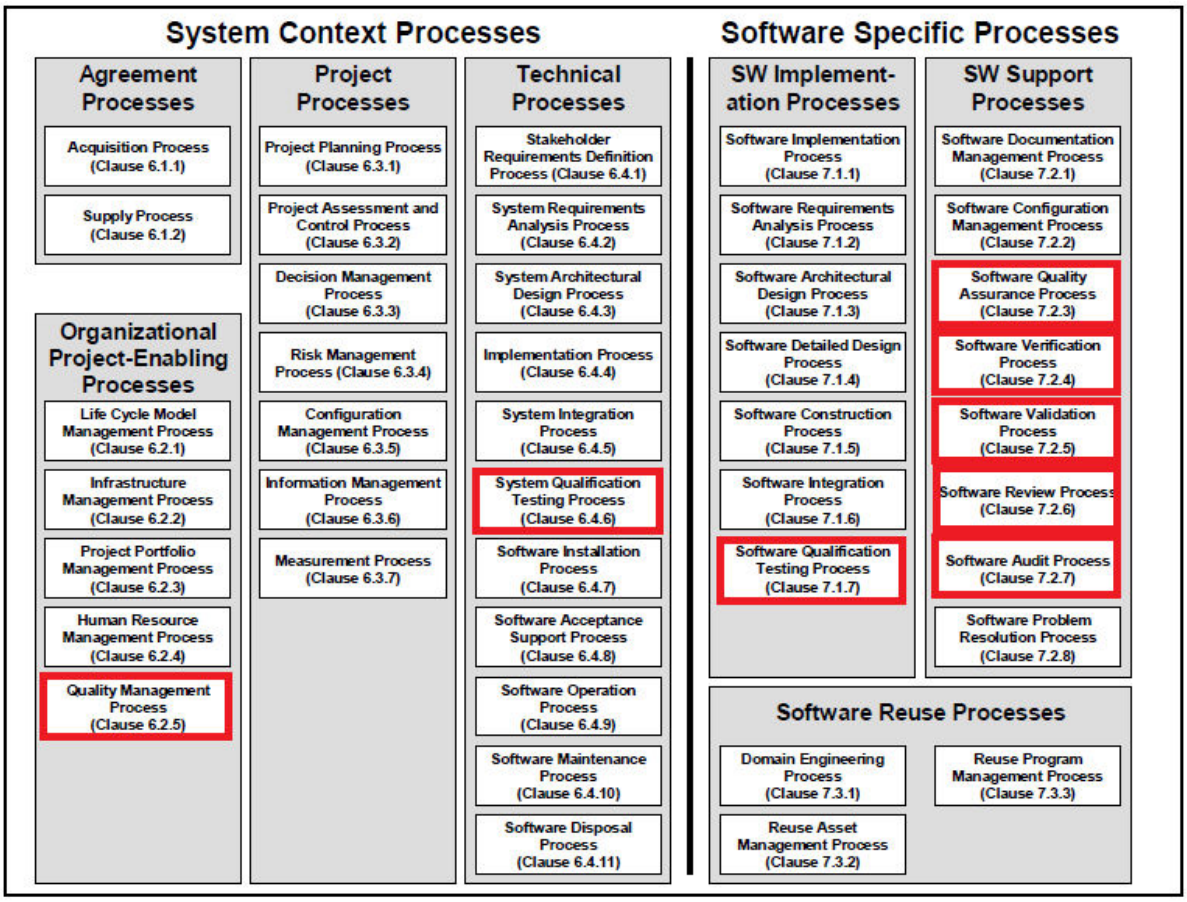
Standard ISO/IEC 12207:2008 definuje 43 procesů a rozčleňuje je do 7 procesních skupin: Smluvní procesy *(Agreement Processes)*, Procesy podporující projekty na úrovni organizace *(Organizational Project-Enabling Processes)*, Projektové procesy *(Project Processes),* Technické procesy *(Technical Processes)*, Procesy implementace softwaru *(SW Implementation Processes)*, Procesy podpory softwaru *(SW Support Processes)* a Procesy znovupoužití softwaru *(Software Reuse Processes)*. Stručná charakteristika těchto procesů je uvedena v [Buchalcevová, 2009, s. 28-33]. U každého procesu je uveden název, účel, výstupy, činnosti a úkoly. Tyto procesy a procesní oblasti znázorňuje Obrázek č. 2. Červeně orámované procesy životního cyklu softwaru se týkají oblasti testování a zajišťování kvality softwaru.

Účelem procesu Řízení kvality *(Quality Management Process)* je zajistit, že produkty, služby a procesy životního cyklu softwaru odpovídají cílům kvality organizace a vedou k uspokojení zákazníků [ISO/IEC, 2008, s. 31].

Účelem procesu Testování systému *(System Qualification Testing Process)* je zajistit, že všechny implementované požadavky na systém jsou otestovány a systém je připraven pro předání [ISO/IEC, 2008, s. 49].

Účelem procesu Testování softwaru *(Software Qualification Testing Process)* je ověřit, že integrovaný softwarový produkt splňuje definované požadavky [ISO/IEC, 2008, s. 65].

Účelem procesu Zajištění kvality softwaru *(Software Quality Assurance Process)* je zajistit, že pracovní produkty a procesy jsou v souladu s předem definovanými předpisy a plány [ISO/IEC, 2008, s. 69].




**Obrázek č. 2 Procesy životního cyklu softwaru podle skupin a zvýraznění procesů se vztahem k testování a řízení kvality softwaru, zdroj: [ISO/IEC, 2008, s. 14]**

Účelem procesu Verifikace softwaru *(Software Verification Process)* je potvrdit, že softwarový produkt a/nebo služba splňují specifikované požadavky [ISO/IEC, 2008, s. 71].

Účelem procesu Validace softwaru *(Software Validation Process)* je ověřit, že softwarový produkt lze používat v souladu se specifikovanými požadavky na použití [ISO/IEC, 2008, s. 73].

Účelem procesu Revize softwaru *(Software Review Process)* je *„poskytnout všem zainteresovaným stranám informace o stavu projektu ve vztahu k odsouhlaseným cílům a plánu“* [Buchalcevová, 2009, s. 32]. Revize se týkají jak řízení projektů, tak i technických záležitostí a konají se v průběhu celého projektu [ISO/IEC, 2008, s. 74].

Účelem procesu Audit softwaru *(Software Audit Process)* je nezávislé posouzení, zda jsou dané produkty a procesy v souladu s požadavky, plány a smlouvami [ISO/IEC, 2008, s. 76].

### 4.1.3 Standardy ISO/IEC SQuaRE

Série mezinárodních standardů ISO/IEC 250nn se zaměřuje na kvalitu systémů a softwaru. Na rozdíl od řady norem ISO 9000, které jsou zaměřeny na řízení kvality procesu vývoje produktu, řada ISO/IEC SQuaRE se soustředí na kvalitu výsledného produktu a vztahuje se pouze na softwarovou oblast.

Základním standardem této řady je směrnice ISO/IEC 25000:2005 *Software product Quality Requirement and Evaluation (SQuaRE) – Guide to SQuaRE*, která poskytuje návod pro použití této série standardů. Obsahuje celkový přehled SQuaRE, referenční modely a definice, vztahy mezi jednotlivými dokumenty a slouží k tomu, aby uživatelé této příručky porozuměli, jak s celou sadou SQuaRE pracovat. Ostatních 11 standardů sady SQuaRE se vždy vztahuje k nějakému konkrétnímu účelu, jako jsou specifikace požadavků, plánování a řízení, nebo měření a vyhodnocování. [ISO, 2013]

Tato řada vznikla sloučením a nahrazením předchozích standardů ISO/IEC 9126 *Software engineering – Product quality*[13], který se zaměřoval na metriky kvality softwarového produktu a ISO/IEC 14598 *Information technology – Software product evaluation*[14], který byl zaměřen na hodnocení softwarového produktu. [ISO, 2013]

Oproti standardům ISO 9000 a ISO/IEC 12207 řada ISO/IEC SQuaRE zatím není příliš rozšířená a ani v České republice dosud nebyla vydána v podobě českých technických norem.

### 4.1.4 Standard ISO/IEC/IEEE 29119

Cílem standardu ISO/IEC/IEEE 29119 *Software and systems engineering – Software testing* je vydání jednoho definitivního standardu pro testování softwaru, který bude obsahovat terminologický slovník a definovat procesy, dokumentaci, techniky a model hodnocení procesů pro testování softwaru, který bude využitelný pro všechny modely životního cyklu softwaru. [SoftwareTestingStandard, 2012]

Tento standard dosud nebyl publikován, ale čtyři jeho části již jsou ve schvalovacím řízení, takže budou pravděpodobně brzy vydány:

- *Part 1: Concepts and definitions (Část 1: Koncepty a definice),*
- *Part 2: Test process (Část 2: Proces testování),*
- *Part 3: Test documentation (Část 3: Testovací dokumentace),*

[13] V České republice je dostupný pod označením ČSN ISO/IEC 9126 *Softwarové inženýrství – Jakost produktu*.

[14] V České republice je dostupný pod označením ČSN ISO/IEC 14598 *Softwarové inženýrství – Hodnocení produktu.*

- *Part 4: Test techniques (Část 4: Techniky testování).*

Pátá část standardu s označením *Part 5: Keyword-Driven Testing*[15] je teprve ve fázi příprav. [ISO, 2013]

Tento standard má nahrazovat existující standardy *IEEE Standard for Software and System Test Documentation (IEEE 829)*, *Standard for Software Unit Testing (IEEE 1008)*, a britské standardy pro testování softwaru BS 7925-1 *Vocabulary of Terms in Software Testing*, BS 7925-2 *Software Component Testing Standard*. [SoftwareTestingStandard, 2012]

## 4.2 Standardy IEEE

Organizace The Institute of Electrical and Electronics Engineers (IEEE) [IEEE, 2013a] je největším světovým profesním sdružením, které usiluje o vzestup a inovace technologií ve prospěch lidstva. Byla založena v USA a jejími členy jsou zástupci z více než sta zemí. [IEEE, 2013a] Sdružuje dobrovolníky, kteří na vytváření finálního produktu pohlíží rozdílnými způsoby a mají na finálním produktu různé zájmy. Publikace vydané organizací IEEE, konference, technologické standardy i profesní a vzdělávací aktivity pod záštitou IEEE jsou zdrojem inspirace pro společnost na celém světě. [IEEE, 2008] Standardy IEEE nejsou překládány do češtiny. Na území České a Slovenské republiky je organizace IEEE zastoupena Československou sekcí IEEE, která pořádá řadu konferencí, vydává časopisy a pomáhá tuzemským organizacím navazovat mezinárodní kontakty. [IEEE, 2013b]

Mnoho standardů, které organizace IEEE vydala, nachází uplatnění také v oblasti testování a zajišťování kvality softwaru. Jejich aplikace v organizacích je čistě dobrovolná a není spjata s nárokem na certifikace.

Standardy IEEE se vztahem k testování a zajišťování kvality softwaru jsou následující:

- IEEE 829-2008 *Standard for Software and System Test Documentation*,
- IEEE 1008-1987 *Standard for Software Unit Testing*,
- IEEE 1012-2012 *Standard for System and Software Verification and Validation*,
- IEEE 1028-2008 *Standard for Software Reviews and Audits*,
- IEEE 1044-2009 *Standard Classification for Software Anomalies*,

[15] **Keyword-Driven Testing** je způsob vytváření automatizovaných testů, kdy tester pouze nahrává sekvenci akcí a případně k nim přidává parametry. Není zapotřebí programování skriptů a spouštění testů probíhá tak, že nástroj pro automatizaci testů simuluje nahrané akce testera a zaznamenává výsledky testu.

- IEEE 1044.1-1995 *Guide to Classification for Software Anomalies*,
- IEEE 830-1998 *Recommended Practice for Software Requirements Specifications*,
- IEEE 29148-2011 *Systems and software engineering – Life cycle processes - Requirements engineering,*
- IEEE 730-2002 *Standard for Software Quality Assurance Plans*,
- IEEE 1061-1998 *Standard for a Software Quality Metrics Methodology*,
- IEEE 610.12-1990 *Standard Glossary of Software Engineering Terminology*.

Standardy IEEE 1012 a IEEE 1028 jsou popsány v knize *Software Quality Assurance From theory to implementation* [Galin, 2004, s. 514-521]. V této kapitole se více zaměřuji pouze na charakteristiku standardu IEEE 829, z něhož vychází většina pracovních produktů metodiky testování v praktické části této práce.

### 4.2.1 Standard IEEE 829[16]

Hlavním cílem standardu IEEE 829-2008 *Standard for Software and System Test Documentation* je specifikovat, co má obsahovat testovací dokumentace vytvářená při jednotlivých úlohách procesu testování a jak má být používána.

Standard definuje rámec pro procesy testování, činnosti a úlohy, které pokrývají celý životní cyklus softwaru včetně pořízení, dodání, vývoje provozu a údržby. Definuje různé úrovně integrity, prostřednictvím nichž je kvantifikována důležitost softwaru pro zainteresované strany. Důležitost software se určuje na základě vybraných atributů jako je komplexnost, hodnocení rizik, úroveň bezpečnosti a spolehlivosti, datová integrita, požadovaný výkon, cena, velikost a další charakteristiky softwaru. Jako příklad poskytuje vzorové čtyřstupňové schéma sestavené na základě závažnosti důsledků selhání a možnosti snížení pravděpodobnosti selhání nebo zmírnění důsledků selhání. Jednotlivé úrovně tohoto schématu spolu s jejich popisem zachycuje Tabulka č. 2.

Pro každou úroveň tohoto schématu standard doporučuje minimální množství úloh, které by měly být v rámci testování vykonávány. Na základě množství úloh testování se potom odvíjí i množství testovací dokumentace.

[16] Text této kapitoly vychází ze zdroje [IEEE, 2008].

**Tabulka č. 2 Schéma úrovní integrity založené na důsledcích selhání, zdroj: [IEEE, 2008, s. 13-14]**

| Úroveň | Popis |
|---|---|
| **4 - Katastrofická** ***(Catastrophic)*** | Software musí běžet správně, protože jinak hrozí hluboce závažné následky (ztráty na životech, ztráta systému, škoda na životním prostředí, ekonomická nebo sociální ztráta). Není možné žádné zmírnění následků selhání. |
| **3 – Kritická** ***(Critical)*** | Software musí běžet správně, protože jinak neplní účel použití, což má vážné následky (permanentní zranění, znehodnocení hlavního systému, škoda na životním prostředí, ekonomický nebo sociální dopad). Je možné částečné až kompletní zmírnění následků selhání. |
| **2 – Marginální** ***(Marginal)*** | Software musí běžet správně, protože jinak neplní požadované funkce, což má zároveň vedlejší dopady. Je možné kompletní odstranění následků selhání. |
| **1 – Zanedbatelná** ***(Negligible)*** | Software musí běžet správně, protože jinak neplní požadované funkce, což má ale pouze zanedbatelné dopady. Zmínění následků selhání není zapotřebí. |

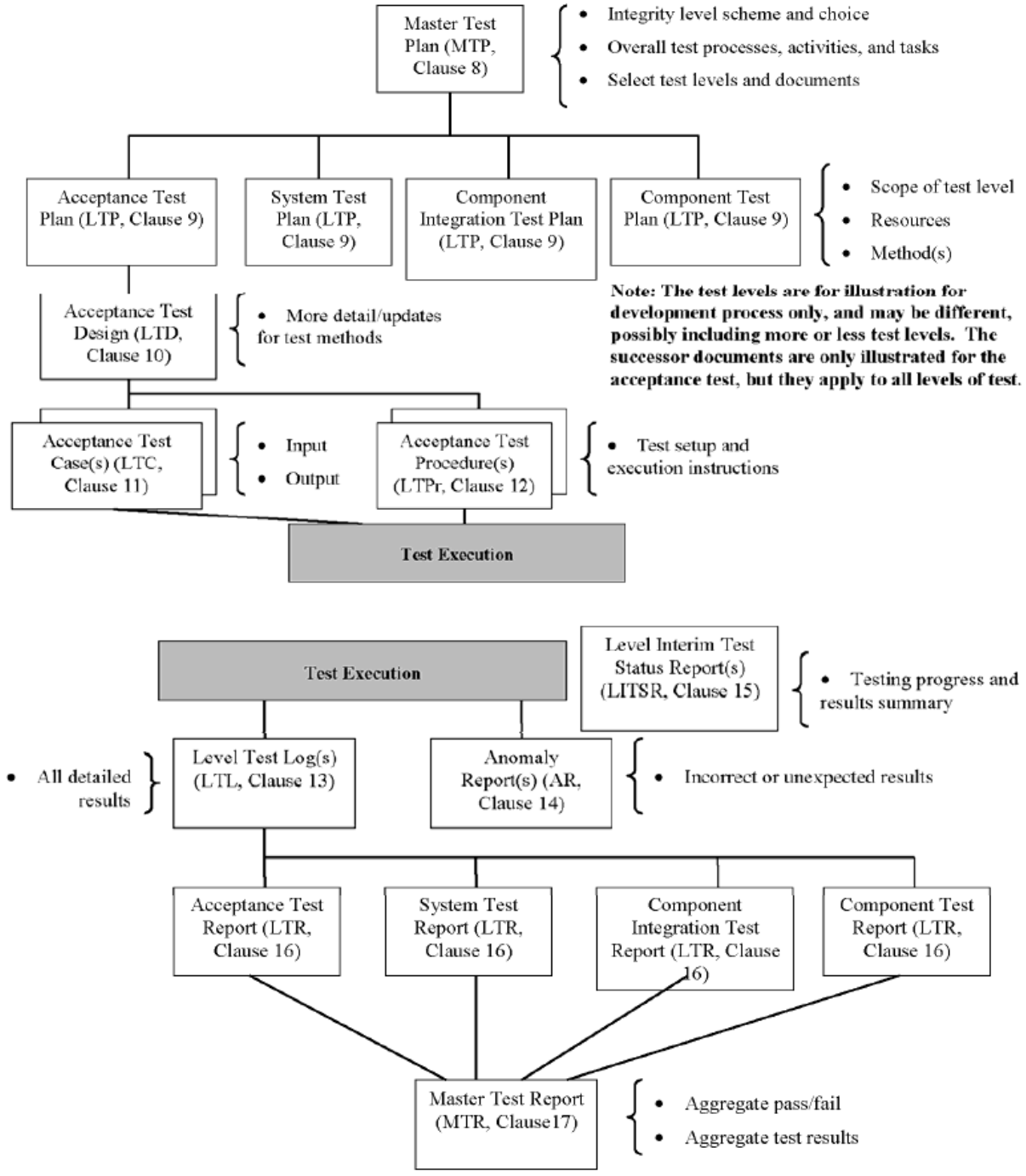
Standard definuje celkem 10 typů dokumentů pro testování rozdělené do čtyř úrovní, jak znázorňuje Obrázek č. 3. Na nejvyšší úrovni stojí Hlavní plán testování *(Master Test Plan)*, který identifikuje, kolik bude úrovní testování. Pro jednotlivé úrovně testování definované v Hlavním plánu testování jsou potom vytvářeny samostatné plány *(Level Test Plan)*. Na každý Plán úrovně testování navazují detailnější dokumenty jako Design úrovně testování *(Level test Design)* a dále Testovací případy a procedury pro danou úroveň testování *(Level Test Case a Level Test Procedure)*, které poskytují detailní informace pro provádění testů. Během provádění testů jsou vytvářeny Reporty o stavu testování *(Interim Level Test Status Report)* a následně potom Test logy *(Level Test Log)*, Reporty o anomáliích *(Anomaly Report)* a Reporty pro úroveň testování *(Level Test Report).* Všechny reporty za jednotlivé úrovně testování jsou nakonec spojeny do výsledného Hlavního reportu za testování *(Master Test Report)*, který obsahuje závěrečné shrnutí za všechny provedené úrovně testování.

Pro každý typ dokumentu standard uvádí stručnou charakteristiku a účel použití a specifikuje jednotlivé prvky, které by měl dokument obsahovat. Dokumenty je možné kombinovat, nebo vynechat některé prvky dokumentu doporučené standardem.

Standard může být aplikován na všechny systémy založené na softwaru, počítačový software, hardware i jejich rozhraní. Může být použit na vývoj softwaru, údržbu softwaru i znovupoužitý software. Cílem procesů testování softwaru nebo systému je určit, zda jsou

výsledky určité aktivity v souladu s požadavky na tuto aktivitu a zda vyvinutý produkt naplňuje potřeby uživatelů a plní požadovaný účel použití. Procesy testování mohou zahrnovat inspekci, analýzu, demonstrování, verifikaci a validaci softwaru nebo systému na softwaru založeném.

Tento standard není svázán s žádným konkrétním modelem životního cyklu softwaru a je využitelný pro všechny modely životního cyklu softwaru.




**Obrázek č. 3 Přehled testovací dokumentace, zdroj: [IEEE, 2008, s. 34]**

## 4.3 Shrnutí kapitoly

Standardy reflektují zkušenosti a znalosti skupiny profesionálů a pro test manažery mohou být užitečné jako vodítka při zavádění metodiky testování v organizaci[17]. V této kapitole byly standardy rozčleněny podle původu na mezinárodní, národní a doménově specifické, přičemž pozornost byla věnována především standardům mezinárodním. Byl uveden přehled mezinárodních standardů vztahujících se k oblasti testování a zajišťování kvality softwaru a byly charakterizovány standardy řady ISO 9000, ISO/IEC 12207, ISO/IEC SQuaRE, ISO/IEC/IEEE 29119 a IEEE 829.

Pokud se uvažuje o použití více standardů zároveň, je třeba ověřit, zda jsou jednotlivé standardy ve vzájemném souladu. Některé standardy mohou být nekonzistentní s jinými, nebo s nimi mohou být dokonce v rozporu. Je třeba, aby test manažer dokázal vybrat ze standardů jen takové specifikace a kritéria, které vhodně podpoří testování v kontextu dané organizace.

[17] Standardy jsou dále důležité i pro získání certifikace či posouzení způsobilosti a zralosti organizace.

# 5 Certifikace v oboru testování

Vzhledem k rostoucí konkurenci na trhu práce se uchazeči o práci snaží působit co nejatraktivněji pro potenciální zaměstnavatele. Podmínkou přijetí na většinu pozic, nejen v oboru informačních technologií, je zaslání životopisu, kde uchazeč uvádí dosažené vzdělání, dosavadní praxi a další získané dovednosti. Každá pozice typicky vyžaduje specializované dovednosti. Školní vzdělávací systémy ovšem většinou poskytují pouze obecný rozhled v daném oboru a nespecializují se konkrétně na danou pozici. Prohloubení znalostí potřebných pro výkon konkrétní pozice je ponecháván na studentech samotných. Jedním ze způsobů, jak prokázat, že daný uchazeč má skutečně určitou úroveň a rozsah znalostí v dané disciplíně, je získání a následné předložení certifikátu.

Certifikát je dle [Košař, 2011, s. 9] definován jako „*listina vydaná certifikační autoritou, která dokazuje určité teoretické nebo (i) praktické dovednosti a znalosti jejího držitele*.“

Certifikace je potom *„Proces potvrzení, že komponenta, systém nebo osoba splňují specifikované požadavky.“* [ISTQB, 2012c, s. 12] U osob se tedy jedná o formální posouzení odbornosti v určité oblasti.

V některých oborech je dokonce zisk certifikátu předpokladem pro povýšení nebo získání nové pozice. Pro přijetí na pozici softwarového testera není certifikát podmínkou, ale může být výhodou v konkurenci ostatních uchazečů.

Na druhou stranu je třeba si uvědomit, že získání certifikátu nevypovídá nic o pracovním nasazení daného testera a reálných schopnostech testovat software v praxi. Certifikát pouze prokazuje, že tester má určitou sadu znalostí, umí je zasadit do kontextu a případně aplikovat na příkladech, což je předpokladem pro jejich úspěšné uplatnění při testování softwaru v praxi. Pro přijetí na pracovní pozici kromě teoretických a praktických zkušeností může hrát roli i dosažené vzdělání, jazykové dovednosti i osobnost testera jako člověka.

Světově nejuznávanějším systémem pro certifikaci testerů a nejvyšším počtem certifikovaných osob v oboru testování se pyšní **organizace ISTQB**. Vzhledem k tomu, že praktická část diplomové práce vychází z materiálů pro certifikace ISTQB, je certifikacím ISTQB věnována samostatná podkapitola.

## 5.1 Certifikace ISTQB

Organizace International Software Testing Qualifications Board (ISTQB) byla založena roku 2002 jako nezisková asociace. Je tvůrcem celosvětově uznávaného schématu pro certifikaci softwarových testerů, díky němuž se dostala do pozice vedoucí organizace zabývajících se certifikací kompetencí pro oblast testování softwaru. [ISTQB, 2013a]

V prosinci roku 2012 dosáhl celkový počet vydaných certifikátů sumy 283 697, což organizaci ISTQB řadí celkově na třetí místo dle počtu certifikovaných osob v oboru informačních technologií (první dvě místa patří certifikacím PMI a ITILU) a v oblasti testování jednoznačně na první místo [ISTQB, 2013b, s. 37-38]. Její globální význam dokládá i rozšíření do 70 zemí světa a nárůst počtu certifikací okolo 10 000 nových certifikací během jednoho čtvrtletí. [ISTQB, 2013b]

Vizí ISTQB *je „Neustále zlepšovat a posouvat profesi testování softwaru kupředu: Definováním a udržováním souboru znalostí, které testerovi umožňují být certifikován na základě nejlepších praktik, které spojují mezinárodní komunitu testování softwaru a podporují výzkum“* [ISTQB, 2013a].

Misí organizace ISTQB je mimo jiné neustálý posun souboru znalostí o testování směrem kupředu založený na inovativních průzkumech a nejlepších praktikách, které jsou dostupné v odvětví testování. Tyto znalosti, ideje a inovace jsou volně sdíleny, což utváří konzistentní náhled na oblast testování napříč zeměmi celého světa. [ISTQB, 2013a]

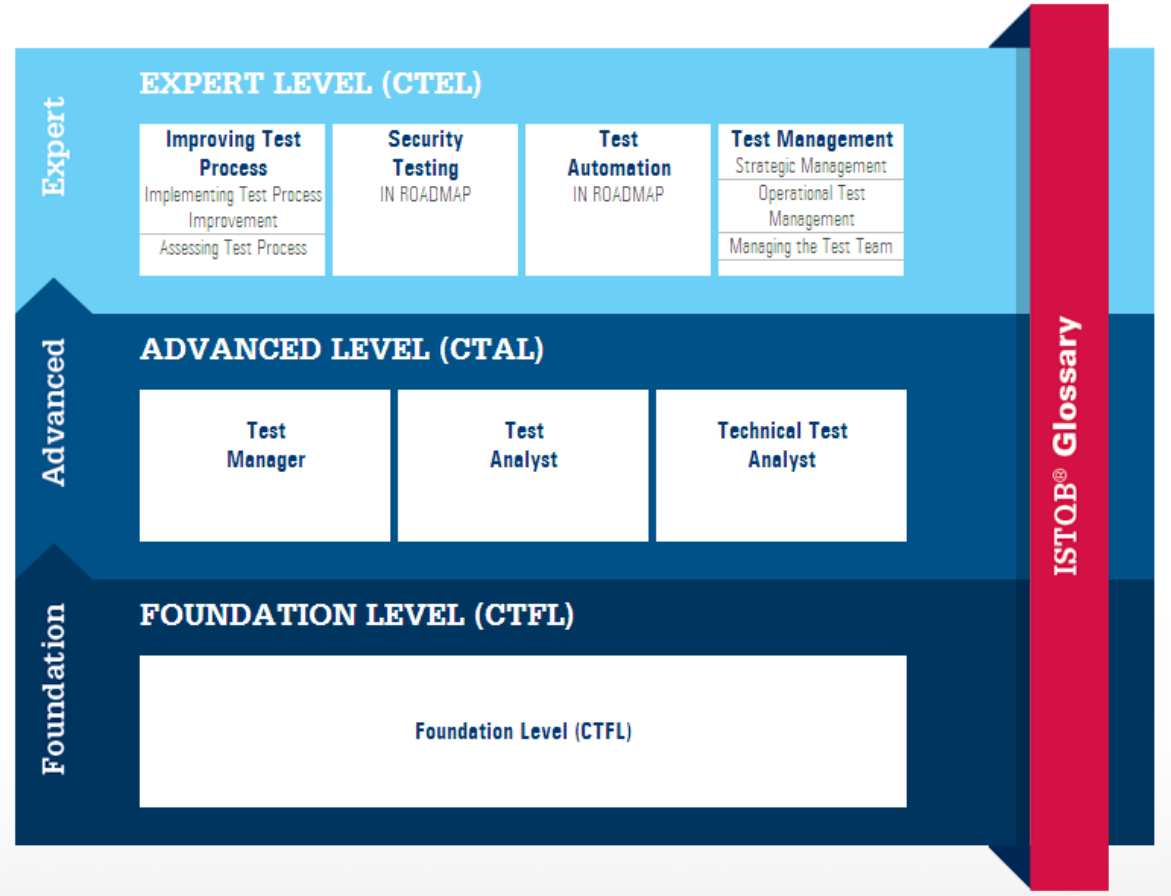
**Kvalifikační schéma** pro certifikace ISTQB zachycuje Obrázek č. 4.

Nejníže je základní úroveň *(Foundation Level)*. Materiálem na přípravu pro základní úroveň certifikace je *Foundation Level Syllabus* [ISTQB, 2011].

Na pokročilé úrovni *(Advanced Level)* si testeři mohou vybrat ze třech zaměření: Test manažer *(Test Manager)*, Test analytik *(Test Analyst)* a Technický test analytik *(Technical Test Anylyst)*. Pro každé z těchto zaměření je zpracován speciální materiál pro přípravu, tedy *Advanced Level Syllabus Test Manager* [ISTQB, 2012a], *Advanced Level Syllabus Test Analyst* [ISTQB, 2012b] a *Advanced Level Syllabus Technical Test Analyst*. Vstupním požadavkem pro získání certifikátu na pokročilé úrovni je certifikát základní úrovně.

Na expertní úrovni *(Expert Level)* je možnost výběru ze zaměření na Zlepšování procesu testování *(Improving Test Process)*, Bezpečnostního testování *(Security testing)*, Automatizace testování *(Test Automation)* a Řízení testování *(Test Management)*. Vytváření materiálů pro přípravu na certifikaci v jednotlivých zaměřeních vzniká postupně. Zatím existuje materiál jen pro Zlepšování procesů testování *Expert Level Syllabus – Improving the Testing* a pro Řízení testování *Expert Level Syllabus – Test Management* (oba ve verzi z roku 2011). Stávající čtyři oblasti na expertní úrovni plánuje organizace ISTQB rozšířit o další oblasti na základě potřeb trhu. Předpokladem pro získání certifikace pro oblasti Zlepšování procesu testování a Řízení testování je získání certifikátu na pokročilé úrovni se zaměřením Test Manažer. Pro oblasti Automatizace testování a Bezpečnostní testy bude předpokladem pokročilá certifikace se zaměřením Technical Test Analyst. Pro zaměření Test Analytik zatím neexistuje odpovídající zaměření na expertní úrovni.

Pro všechny ISTQB materiály platí jednotný slovník termínů (ve verzi z roku 2012). Sylaby jednotlivých úrovní jsou vzájemně v souladu, takže například pokročilá úroveň Test analytik navazuje na základní úroveň, přičemž se zaměřuje Proces testování, Řízení testování, Testovací techniky, Testování charakteristiky softwarové kvality, Revize, Defekt management a testovací nástroje.




**Obrázek č. 4 Kvalifikační schéma pro certifikace ISTQB, zdroj: [ISTQB, 2013a]**

Certifikačním orgánem pro ISTQB certifikace v České republice je CSQ-CERT při České společnosti pro jakost. V současnosti je možné skládat zkoušky a získat certifikát na základní úroveň a na všechny podtypy úrovně pokročilé. Zkoušky probíhají písemně a u základní úrovně je možnost vykonávat zkoušku v českém jazyce. Materiál pro přípravu ale v české verzi dosud není. Pokročilá úroveň je dostupná pouze v angličtině.

Pro testery je získání ISTQB příležitostí, jak prokázat znalost a schopnost aplikovat nejlepší praktiky a koncepty v oblasti testování, což může pomoci při jejich kariéře. Mohou se tak odlišit od množství méně kvalifikovaných testerů a obstát v konkurenci zájemců o pozici testera na trhu práce. Certifikovaní testeři jsou přínosem i pro samotné organizace, které je zaměstnávají. Vědí, jak dodat testování konzistenci a znovupoužitelnost, jak redukovat rizika a snížit náklady nedostatečné kvality softwaru a jsou tak schopni zajistit, aby testování bylo efektivní. [Black, 2008, s. 27]

Mimo tyto přínosy ISTQB také přináší hodnotu samotné profesi softwarového testování. Spojuje nejlepší praktiky a koncepty oboru testování softwaru, které se formují už přes 30 let. Staví na konceptech Glenforda Myerse, Richarda Bendera, Billa Hetzela, Borise Beizera, kteří položili základní kameny testování softwaru a na konceptech, které se z těchto základů postupem času vyvinuly. Na tomto základě formuje praktiky a koncepty, které definují profesi testování a které by profesionální testeři měli znát a umět aplikovat. [Black, 2008, s. 28]

## 5.2 Shrnutí kapitoly

Získání certifikátu softwarovému testerovi nezaručí, že získá požadované pracovní místo. Může mít v tomto ohledu výhodu oproti ostatním uchazečům, kteří nejsou certifikovaní, ale pro přijetí na pracovní místo hrají roli i další faktory jako je dosažené vzdělání, odpovídající praxe, specifické dovednosti a znalosti, ale i osobnost člověka a motivace pro danou práci. Certifikát je pouze formálním posouzením odbornosti.

Organizace ISTQB nabízí propracovaný systém certifikace. Vysoký počet certifikovaných testerů organizací ISTQB je znakem velkého zájmu o získání certifikátu. Ovšem přínosem ISTQB jsou podle mého názoru hlavně dobře zpracované materiály pro přípravu na certifikace, ve kterých jsou sumarizovány nejlepší praktiky a aktuality z oboru testování, a jejich sdílení s širokou veřejností. Vnímání principů testování se tak stává jednotným v zemích celého světa, což umožňuje spolupráci lidí napříč různými národy.

# 6 Metodika testování podle mezinárodních praktik a standardů

V oboru softwarového inženýrství již bylo vyvinuto mnoho metodik. Většinou jsou zaměřeny na celý proces vývoje softwaru, kdy testování je jednou z částí, které není věnována dostatečná pozornost. V metodice RUP je sice testování plně pokryto, nicméně tato metodika a nástroje pro její podporu jsou poskytovány pouze na komerční bázi a pro mnoho firem se stává nedostupnou. Nevýhodou při zavádění RUP může být i přílišná robustnost metodiky a s tím spojené vysoké náklady a úsilí na přizpůsobení metodiky pro potřeby organizace. Na principech RUP vznikly metodiky OpenUP i MMSP, které odstraňují robustnost RUP a jsou volně dostupné, nicméně jsou koncipovány stejně jako RUP pro iterativní vývoj, což je omezením, pokud vývoj softwaru v iteracích neprobíhá. Určitým omezením pro použití existujících metodik je i jazyk, což je u drtivé většiny metodik angličtina, bez potřebné lokalizace do češtiny[18].

Metodiky v češtině, které se soustředí výhradně na testování, vznikají interně ve firmách, kde jsou uzpůsobeny na konkrétní podmínky organizace a nejsou přístupné veřejnosti. Kromě firemních metodik vznikají různé metodiky pro testování i v diplomových pracích. Ty obvykle vznikají lokalizací a rozšířením již existujících metodik, které doplňují o znalosti z odborných knih, vědeckých článků a praktické zkušenosti autorů, ale nejsou vytvářeny podle standardů pro obor testování a nepoužívají standardizovanou terminologii. Některé z nich jsou přímo přizpůsobeny pro potřeby konkrétní firmy.

Zde navržená metodika testování na uvedené nevýhody existujících metodik reaguje. Je soustředěna výhradně na oblast testování softwaru, pokrývá proces testování z hlediska řízení, přípravy i provádění testů a není vázána na konkrétní model životního cyklu vývoje softwaru, i když nejvíce účinná je při aplikaci na V-model. Metodika vychází primárně z **materiálů pro přípravu na certifikace ISTQB**, které spojují nejlepší praktiky a koncepty v oboru testování softwaru od vzniku prvních pramenů testování až po současnost. Konkrétně se jedná o materiál pro přípravu na základní úroveň certifikace [ISTQB, 2011] a pokročilé úrovně se zaměřením test manažer [ISTQB, 2012a] a test analytik [ISTQB, 2012b]. Tyto materiály[19] byly vytvořeny skupinou profesionálů a zároveň jsou v souladu se standardy pro testování, na které se v mnoha místech odvolávají. Metodika přebírá terminologii ISTQB [ISTQB, 2012c] a dbá na její důsledné dodržování.

Materiály ISTQB nemají charakter metodiky, nicméně jedním z témat popisovaných v materiálech je proces testování rozdělený do hlavních aktivit. Na tomto procesu je postavena tato metodika, přičemž z jednolitě popsaných hlavních aktivit procesu testování

[18] To neplatí pro výše uvedenou metodiku MMSP – ta vznikla právě lokalizací metodiky OpenUP do češtiny.

[19] Zdroje ISTQB jsou východiskem pro celou metodiku. V textu této kapitoly jsou proto uváděny jen na nezbytných místech, nebo pokud se jedná o doslovné citace z těchto zdrojů.

v ISTQB vyčleňuje a rozpracovává jednotlivé úlohy a pracovní produkty a dává tak procesu testování ISTQB metodický rámec. Jelikož specifikace pracovních produktů není součástí ISTQB materiálů a ISTQB se za tímto účelem odvolává na **standard IEEE 829**, specifikace pracovních produktů v této metodice vychází ze standardu *IEEE Standard for Software and System Test Documentation (dále jen IEEE 829)* [IEEE, 2008] a jsou zdůvodněny případné odchylky od standardu, resp. využití alternativních variant. U některých pracovních produktů jsou navíc přiloženy i vzorové šablony. Za jednotlivé úlohy a pracovní produkty jsou stanoveny zodpovědnosti v podobě rolí, které v ISTQB nejsou dostatečně pokryty, a proto jsou inspirovány metodikou RUP. Součástí metodiky jsou i techniky, které ISTQB doporučuje využívat při testování. Technik ISTQB popisuje celou řadu, nicméně pro tuto metodiku byly vybrány čtyři nejpoužívanější a je možné metodiku rozšiřovat o další techniky pro testování. Mimo výše uvedené prvky bylo potřebné do metodiky zařadit i koncepty, které doplňují ostatní prvky metodiky o terminologii a další specifikace.

Vysvětlení výše uvedených **prvků metodiky** uvádí Tabulka č. 3.

**Tabulka č. 3 Prvky metodiky, zdroj: [autor; IBM, 2005; MMSP, 2011]**

| Název | Popis | Příklady |
|---|---|---|
| **Hlavní aktivita**[20] | Seskupení úloh, které se vztahují k určité oblasti zájmu a jsou vykonávány ve stejném časovém období. | Plánování testů, implementace testů |
| **Úloha** | Nejmenší jednotky práce, které má smysl definovat jako samostatný proces. Jsou vykonávány rolemi a jedná se o *„popis jak pracovat, aby bylo dosaženo stanovených cílů či byly vytvořeny určité pracovní produkty“* [MMSP, 2011]. | Návrh testovacích případů, příprava testovacích dat, analýza a reportování defektů |
| **Pracovní produkt** | Výstup z procesu testování, který je v průběhu testování vytvářen, modifikován a používán [IBM, 2005]. | Plán testování, defekt report |
| **Technika** | Postupy a přístupy k tomu, jak dosáhnout určitého cíle, jako je snížení míry rizika, redukce množství testovacích případů nebo zvýšení účinnosti a efektivity testování. | Rozdělení tříd ekvivalencí, průzkumné testování |

[20] U všech hlavních aktivit kromě monitorování, reportování a řízení testování se zároveň jedná i o fáze testování. Nicméně z toho důvodu, že monitorování, reportování a řízení testování je aktivitou, která prochází napříč všemi fázemi testování, je přesnější použít termín hlavní aktivita.

| | | |
|---|---|---|
| **Role** | Abstraktní vyjádření souboru zodpovědností za jednotlivé úlohy a pracovní produkty, které mohou být naplněny jednou nebo více osobami [IBM, 2005]. | Test manažer |
| **Koncept** | Poskytují informace, na které se odkazují ostatní prvky metodiky a které je vhodné vyčlenit zvlášť. | Klasifikace defektů |
| **Šablona** | Pomůcka pro opakované vytváření pracovních produktů. Definuje náležitosti, které má pracovní produkt mít a při vytváření vlastního pracovního produktu stačí tyto náležitosti doplnit bez potřeby řešit jejich formu. | Šablona pro testovací případ |

Kromě materiálů ISTQB, které definují především rámec procesu testování, a standardu IEEE 829 metodika čerpá i z dalších zdrojů, jako jsou jiné metodiky, mezinárodně uznávané odborné knihy a články, diplomové práce i články na webu. Inspirací při tvorbě metodiky byly i přednášky kurzu Řízení kvality softwaru a struktura metodiky MMSP. Mimo oficiálních zdrojů je v některých částech metodiky upozorněno i na odchylky a nedostatky procesu testování v praxi proti standardizovaným postupům, které vypozorovala autorka metodiky během své dvouleté praxe na pozici test analytik/tester.

Metodika se soustředí na testování softwaru v **dynamickém pojetí**, tzn. veškeré úlohy a pracovní produkty směřují k řádnému ověření spuštěného softwaru. Není zde explicitně pokryto statické testování, které se zaměřuje na revize[21] a analýzy zdrojového kódu a ostatní projektové dokumentace, aniž by byl systém za běhu. Metodika pouze upozorňuje na nutnost prověření správnosti podkladů testování před vlastním vytvářením pracovních produktů pro testování. Účelem metodiky je pokrytí **úrovně systémových testů**, která ve srovnání s ostatními úrovněmi testování vyžaduje nejvyšší stupeň formálnosti a potřebu proces testování vhodně organizovat i strukturovat. Pro účely ostatních úrovní testů je možné metodiku přizpůsobit a nadbytečné prvky metodiky vynechat.

Pro ilustraci uplatnění této metodiky v praxi je ukázáno, jakým způsobem je možné metodiku podpořit nástrojem pro řízení testů **IBM Rational Quality Manager**. Stručná charakteristika tohoto nástroje je zařazena v příloze B. Ačkoliv je právě na tomto nástroji ukázán možný způsob aplikace metodiky, není metodika na IBM Rational Quality Manager fixována. Je možné pro podporu metodiky použít i jiné nástroje pro řízení testů. Metodika nebyla vytvořena na míru funkcionalitě IBM Rational Quality Manager, ale

[21] **Revize** je dle [IEEE, 1990, s. 64] definována jako *proces nebo mítink, během kterého je určitý pracovní produkt prezentován členům projektového týmu, manažerům, uživatelům, zákazníkům, nebo jiným zájmovým skupinám s cílem získat jejich připomínky či souhlas.“* Typově lze rozlišit revize kódu, revize návrhu, formální revize způsobilosti systému, revize požadavků a revize připravenosti na testování.

opačným způsobem. Principy metodiky byly vytvořeny obecně a až následně je nastíněno jejich použití na konkrétním nástroji. Jediným omezením při aplikaci metodiky v kombinaci s jiným nástrojem je nutnost úpravy šablon pracovních produktů dle možností vybraného nástroje. Zde zařazené vzorové šablony byly vytvořené právě v nástroji IBM Rational Quality Manager a v jiných nástrojích mohou být jinak strukturovány. Nicméně metodika uvádí náležitosti pracovních produktů, a pokud se bude postupovat podle nich, není složité podobné šablony nadefinovat i v jiných nástrojích.

Navržená metodika **není** určena **pro konkrétní společnost**. Metodika je postavena na mezinárodních praktikách a standardech a lze ji **přizpůsobit** pro potřeby organizací s různým zaměřením. V plné šíři je využitelná zejména na středních a větších projektech, kde za testování zodpovídá nezávislý testovací tým. Členové testovacího týmu jsou specializovaní testeři, v organizaci je kladen důraz na kvalitu softwaru a testovací tým je vnímán jako strážce kvality a je respektován všemi členy projektového týmu. Pro malé projekty a projektové týmy, kde za testování často zodpovídají vývojáři, tato metodika není příliš vhodná, jelikož by vzhledem k množství definovaných prvků představovala přílišnou zátěž. Z pohledu kritičnosti testovaných aplikací je tato metodika určena pro méně kritické aplikace, kde případné defekty nemají za důsledek katastrofické scénáře. Metodika ovšem není vhodná přímo pro nekritické aplikace, kde případné selhání aplikace nemá žádné dopady. Vzhledem k tomu, že metodika obsahuje i prvky zaměřené na rizika kvality produktu, je vhodná pro aplikace, u kterých je důsledek případného selhání větší, než náklady spojené s odstraněním defektu v průběhu testování.

Uživateli této metodiky mohou být všichni, kdo jsou zainteresováni na procesu testování od vedoucích projektů, manažerů vývoje, vývojářů a zejména test manažeři, test analytici, testeři i správci testovacího prostředí a případně další zájemci.

Metodiku je možné využít i v rámci kompetenčního centra Software Quality Assurance. Kompetenční centrum sdružuje studenty VŠE, kteří mají zájem si vedle studia osvojit i praktické zkušenosti a VŠE jim v tomto směru nabízí možnost zapojení do reálných projektů. Kompetenční centrum se zaměřuje na různé typy testů - testy funkcionalit a výkonnostní testy[22]. Bezpečnostní testování je vzhledem k potřebě jiných nástrojů a certifikací prováděno jiným týmem. [Vomáčko, 2012, s. 10] Metodiku speciálně zaměřenou na zátěžové testování pro kompetenční centrum vyvinul v rámci své diplomové práce Vít Vomáčko [Vomáčko, 2012]. Metodika testování podle mezinárodních praktik a standardů navržená v této práci se, na rozdíl od metodiky zátěžového testování, nespecializuje na konkrétní typ testů a je možné ji využít pro řízení procesu testování komplexně.

Studenti v kompetenčním centru mají přístup ke komerčním nástrojům firmy IBM a mohou tedy využívat pro řízení testování nástroj IBM Rational Quality Manager. Ten je

[22] Do **výkonnostních testů** se dle [IBM, 2005] zahrnují testy výkonnostních profilů, benchmarking testy a zátěžové (load) testy. Rozdělení testů do jednotlivých typů podle metody FURPS je zahrnuto v kapitole 6. 2. 1. 3.

spojným bodem i s touto metodikou. Metodika udává směr, jakým je vhodné při testování postupovat a kompetenční centrum se může touto metodikou při vlastním řízení testování inspirovat.

Kromě výše zmíněných možností využití má metodika přínos i pro studenty kurzů řízení kvality softwarů a testování, kteří se chtějí seznámit s tím, jak může probíhat proces testování a které pracovní produkty, role a techniky může obsahovat. Metodika může sloužit i jako pomůcka při přípravě na základní úroveň certifikace ISTQB, jelikož rozebírá proces testování dle ISTQB a na rozdíl od materiálů ISTQB je v českém jazyce. Nicméně materiál ISTQB má širší rozsah a věnuje se i mnoha dalším tématům než je proces testování, které nejsou metodikou pokryty, takže při přípravě na ISTQB certifikace je vhodné tuto metodiku použít jako doplňkový materiál k oficiálním ISTQB materiálům.

Metodika byla zpracována kromě dokumentové podoby i do podoby webové prezentace za pomoci nástroje pro vytváření a správu metodik EPF Composer[23]. Náhledy na webovou podobu metodiky jsou zařazeny v příloze E. V tomto nástroji může být metodika nadále rozšiřována nebo upravována. Její aktuální verze je volně dostupná na webových stránkách http://metodikamezitest.asp2.cz/.

## 6.1 Koncepty

Koncepty poskytují informace, na které se odkazují ostatní prvky metodiky. Jedná se o souvislosti termínů používaných napříč metodikou a specifikace hodnot, kterých mohou nabývat pole defekt reportu. Je zde uvedena definice defektu ve vztahu k příbuzným termínům, následuje klasifikace defektů podle závažnosti a priority a posledním konceptem je životní cyklus defektu, kde jsou specifikovány stavy, kterými defekt prochází od založení do systému až po řádné uzavření, zamítnutí nebo odložení.

### 6.1.1 Definice defektu a příbuzných termínů

V odvětví testování se lze setkat s různými termíny pro chyby v softwaru. V této metodice jsou používány pojmy dle termininologie ISTQB: anomálie *(anomaly)*, defekt *(defect)*, selhání *(failure)* a omyl *(error)*.

Při provádění testů testeři srovnávají aktuální výsledky s očekávanými. **Anomálie** (taktéž nazývaná jako incident) je dle [ISTQB, 2012c, s. 9] *„každý rozdíl aktuálního chování systému oproti očekáváním založeným na základě specifikací požadavků, designových dokumentů, uživatelských příruček, standardů atd. nebo na základě vnímání a zkušeností testera“*. Každou anomálii je třeba důkladně prošetřit a zjistit, zda se jedná o selhání způ-

[23] Souhrnný popis principů, koncepce a způsobů užití nástroje **Eclipse Process Framework Composer** je možné nalézt v [Pospíšil, 2009] nebo v [Rejnková, 2011, s. 13-20].

sobené defektem v systému či komponentě, nebo jen o omyl na straně testovacího týmu. Anomálie tedy může, ale i nemusí být přímo defektem.

Pokud se zjistí, že příčinou rozdílu aktuálního a očekávaného chování je skutečně defekt, je zapotřebí založit defekt report[24] a daný problém dále řešit. **Defekt** je dle [ISTQB, 2012c, s. 18] definován jako *„chyba v komponentě nebo v systému, která může způsobit, že systém selže při vykonávání požadované funkce, jako například nekorektní příkaz nebo definice dat“*. V tomto pojetí je pojem defekt ekvivalentní s pojmem vada, popř. chyba *(fault)*. Český výraz defekt má foneticky blíže k anglickému pojmu *defect*, používanému v materiálech ISTQB, a proto je jeho použití v této metodice vhodnější oproti alternativám vada či chyba.

S pojmem defekt je úzce spjat i pojem **selhání**. Pokud se při běhu komponenty či systému narazí na odchylku oproti očekávané dodávce, službě nebo výsledku, jedná se o selhání, které zhoršuje či znemožňuje používání systému [ISTQB, 2012c, s. 22]. Většina defektů je odhalena právě při pátrání po příčině selhání.

Jak ovšem tester pozná, že pozorované chování je v rozporu s chováním očekávaným? Způsob, jakým lze identifikovat, že se jedná o selhání softwaru, velice dobře vystihuje definice softwarové chyby, kterou uvádí Patton:

*„O softwarové chybě hovoříme tehdy, je-li splněna jedna nebo více z následujících podmínek:*

1. *Software nedělá něco, co by podle specifikace produktu dělat měl.*
2. *Software dělá něco, co by podle údajů specifikace produktu dělat neměl.*
3. *Software dělá něco, o čem se produktová specifikace nezmiňuje.*
4. *Software nedělá něco, o čem se produktová specifikace nezmiňuje, ale měla by se zmiňovat.*
5. *Software je obtížně srozumitelný, těžko se s ním pracuje, je pomalý, nebo – podle názoru testera – jej koncový uživatel nebude považovat za správný.“* [2002, s. 14]

Podle této definice je chyba (ve smyslu selhání) stav, kdy se software zachová jinak, než předepisuje specifikace, nebo jinak než je očekáváno [Šplíchalová, 2008, s. 49]. Je zde zároveň vidět dvojí perspektiva při hledání chyb a určování správnosti chování software. První čtyři body představují selhání nalezená při verifikaci software. Cílem procesu **verifikace** je potvrdit, že software vyhovuje zadané specifikaci (systém je vytvořen správně) [Patton, 2002, s. 42]. Pátý bod se zaměřuje na selhání nalezená na základě **validace** software, což je kontrola, zda software vyhovuje požadavkům uživatele (je vytvořen správný

[24] Specifikace pracovního produktu **defekt report** je uvedena v kapitole 6. 5. 5. 2.

systém) [Patton, 2002, s. 42]. Při hledání chyb a určování, zda se jedná či nejedná o selhání softwaru, musí tester pamatovat na verifikaci, ale i na validaci.

Příčinou defektů, selhání a anomálií jsou lidské omyly. **Omyl** je dle [ISTQB, 2012c, s. 20] definován jako *„akce člověka, která způsobuje nekorektní výsledek“*. Může se jednat například o nesprávné pochopení požadavků, kdy vývojář implementuje požadavek v kódu jinak, než bylo zamýšleno a tak vznikne defekt. Na druhé straně se omylu může dopustit i tester, když vyhodnotí chování softwaru jako selhání, ale ve skutečnosti se o selhání nejedná.

### 6.1.2 Klasifikace defektů

V kapitole 6. 5. 5. 2. je uvedena typická struktura defekt reportu. Na základě polí defekt reportu lze defekty klasifikovat z různých hledisek. Pro účely monitorování a reportování je nejdůležitější klasifikace defektů dle závažnosti a priority.

**Závažnost defektu** je dle [ISTQB, 2012c, s. 38] definována jako dopad defektu na vývoj nebo provoz komponenty či systému. Závažnost je hodnocena na základě objektivně stanovené klasifikace. V IBM Rational Quality Manager je defaultně přednastavená pětistupňová škála hodnot. Ve standardu *IEEE Standard Classification for Software Anomalies (dále jen IEEE 1044)* je taktéž doporučena pětistupňová škála hodnot. Hodnoty na čtvrtém a pátém stupni jsou ve standardu IEEE 1044 pojmenovány jako méně významný *(minor)* pro čtvrtý stupeň a bezvýznamný *(inconsequential)* pro pátý stupeň [2010, s. 8]. Dle mého názoru je pro účely této metodiky lepší ponechat názvy z IBM Rational Quality Manager, tedy středně významný *(normal)* pro čtvrtý stupeň a méně významná *(minor)* pro pátý stupeň, protože pokud by tester nějaký defekt označil s příznakem bezvýznamný *(inconsequential)*, vývojář se jistě jeho opravou nebude zabývat, ani kdyby mu na opravu zbyl čas. Jednotlivé klasifikační stupně závažnosti, které uvádí Tabulka č. 4, jsou specifikovány na základě definic uvedených ve standardu IEEE 1044.

**Tabulka č. 4 Definice stupňů závažnosti, zdroj: [IEEE, 2010, s. 8]**

| Stupeň závažnosti | Popis |
|---|---|
| **Blokující** ***(blocker)*** | Defekt brzdí nebo zastavuje další testování a je nutné čekat, až bude opraven, nebo alespoň provizorně vyřešen. |
| **Kritická** ***(critical)*** | Základní operace programu jsou vyřazeny z provozu, nebo jsou významně narušeny, je ohrožena spolehlivost a bezpečnost programu. Defekt nelze obejít alternativní cestou. |
| **Velmi významná** ***(major)*** | Jsou zasaženy základní operace programu, ale je možné přes ně projít, nebo je obejít. Obcházení defektu ovšem klade na uživatele značné nároky. |

| | |
|---|---|
| **Středně významná (*normal)*** | Jsou narušeny operace, které nejsou pro program zásadní. Pokud defekt zasahuje do kritických procesů programu, lze jej jednoduše obejít. |
| **Méně významná *(minor)*** | Defekt nemá zásadní vliv na chod programu. Jedná se o drobné odchylky uživatelského rozhraní proti požadavkům bez dopadu na funkčnost programu. |

**Priorita defektu** je dle [ISTQB, 2012c, s. 33] definována jako *„úroveň významnosti defektu z pohledu byznysu uživatele“*. Klasifikace podle priority záleží na potřebách daného projektu a zákazníka, takže není objektivní, napříč všemi projekty, jako tomu je u závažnosti. Priorita přiřazená k defektu zpravidla určuje prioritu opravy defektu, tzn., který defekt je zapotřebí opravit co nejdříve a který defekt případně může zůstat neopravený. V nástroji IBM Rational Quality Manager je defaultně přednastavená třístupňová škála hodnot. Taktéž ve standardu IEEE 1044 je doporučena třístupňová škála hodnot [2010, s. 8]. Klasifikační stupně priority, které uvádí Tabulka č. 5, jsou specifikovány na základě definic uvedených ve standardu IEEE 1044.

**Tabulka č. 5 Definice stupňů priority, zdroj: [IEEE, 2010, s. 8]**

| Stupeň priority | Popis |
|---|---|
| **Vysoká *(high)*** | Defekty ohodnoceny nejvyšší prioritou pro analýzu a vyřešení. |
| **Střední *(medium)*** | Defekty, které budou analyzovány a řešeny až po defektech s prioritou vysoká. |
| **Nízká *(low)*** | Defekty, které budou analyzovány a řešeny až na konec. Není bezpodmínečně nutné tyto defekty opravit. |

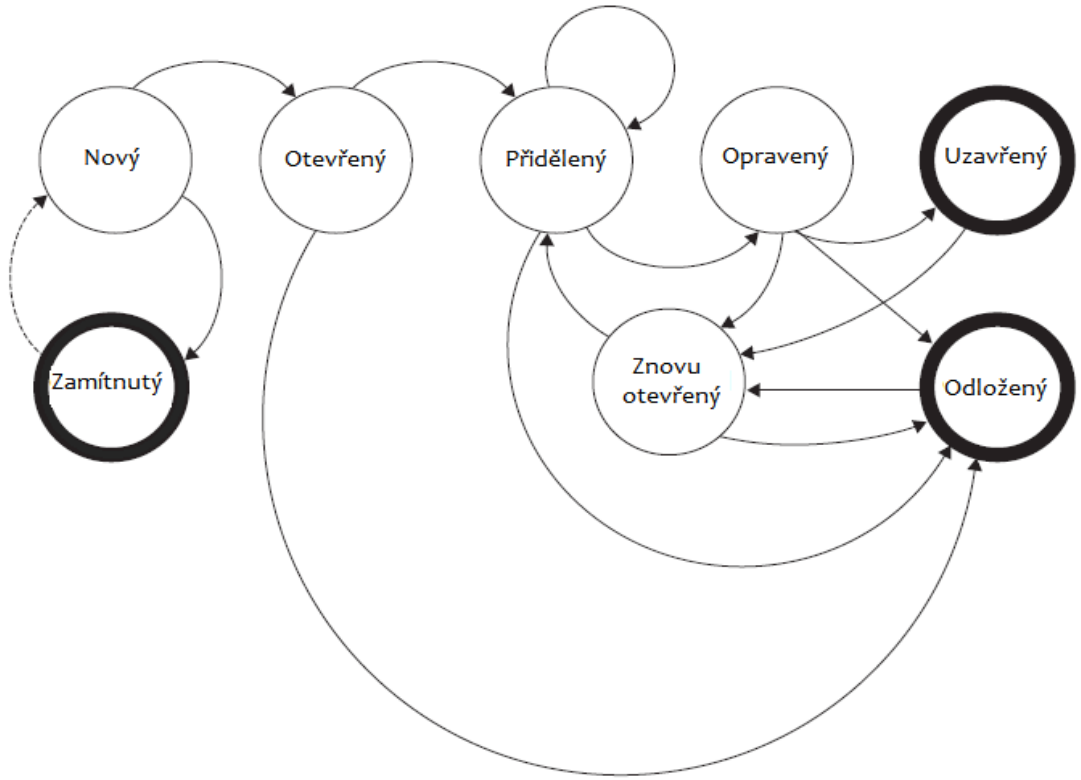
Závažnost a priorita opravy defektu spolu většinou souvisí. Pokud například zhavaruje proces online platby zákazníka zadavatelské firmy, defekt bude mít závažnost Critical (kritická) a prioritu High (vysoká), protože se jedná o selhání kritické operace programu se závažným dopadem na byznys zadavatele. Ne vždy ovšem musí být obě hodnoty na stejné úrovni. V případě, kdy v aplikaci zadavatele bude rozmazané firemní logo, závažnost defektu bude Minor (méně významná), kdežto priorita může být High (vysoká), protože firemní logo se podílí na utváření image firmy a hrozí zde, že nepěkné logo odradí mnohé potenciální zákazníky firmy zadavatele [Ramdeo, 2011].

### 6.1.3 Životní cyklus defektu

Každý defekt report prochází určitým životním cyklem, od nahlášení defektu testerem, přes opravu defektu vývojáři, až po uzavření defektu.

V kapitole 6. 5. 5. 2. je uvedena typická struktura defekt reportu. Jednou z položek defekt reportu je pole „*Status*“, ve kterém se mapuje a sleduje stádium životního cyklu defektu. ISTQB neuvádí žádná doporučení, jak by měl životní cyklus defektu vypadat. Každá organizace má jinou kulturu, pravidla i organizační strukturu a neexistuje žádná standardizovaná podoba životního cyklu defektu. Nicméně každý tester se v praxi s nějakým životním cyklem defektu setkává a je nutné, aby jej organizace měla nadefinovaný a všichni členové projektového týmu aby byli obeznámeni s tím, do jakých stavů se defekt může dostat, které stavy jsou počáteční a koncové a jaké jsou přechody mezi jednotlivými stavy. Protože tato metodika vychází především z mezinárodních standardů a praktik, inspirovala jsem se životním cyklem defektu, který uvádí ve své knize Rex Black [2002, s. 135], uznávaný odborník, který je mimo jiné členem pracovní skupiny organizace ISTQB.

Obrázek č. 6 znázorňuje jednotlivé stavy a přechody mezi nimi. Výchozím stavem je *„Nový“*. Koncové stavy jsou označeny tučným orámováním a jedná se o stavy, kdy se defekt stane neaktivním a už nejsou vyžadovány ani se neočekávají další změny stavů, ačkoliv možné jsou. I uzavřený defekt report je možné znovu otevřít, pokud se například spolu s opravou jiných defektů daný defekt opětovně zanese. Přechody mezi jednotlivými stavy jsou značeny šipkami.




**Obrázek č. 5 Životní cyklus defektu, zdroj: upraveno podle [Black, 2002, s. 135]**

Tabulka č. 6 uvádí použití jednotlivých stavů, životního cyklu defektu. Vychází ze zdroje [Black, 2002, s. 133-134], ale byla upravena pro potřeby této metodiky podle zkušeností autorky práce.

Při přechodu mezi jednotlivými stavy je vhodné používat komentáře, aby osoba, na kterou je defekt report aktuálně přiřazen, věděla, co se od ní očekává. Při dokumentaci přechodů ve formě komentářů je možné v případě potřeby také zpětně dohledat historii defektu (např. pokud se již uzavřený defekt za nějaký čas objeví znovu).

**Tabulka č. 6 Stavy v životním cyklu defektu a jejich použití, zdroj: inspirováno [Black, 2002, s. 133-134]**

| Stav | Kdy se používá |
|---|---|
| **Nový**[25] | Jakmile tester založí do systému nový defekt report. V tomto stavu není defekt report viditelný členům mimo testovací tým a čeká na schválení odpovědným členem testovacího týmu.<br>(Schvalování je zapotřebí zejména u začínajících testerů, zkušení testeři mají oprávnění defekt report posunout do dalšího stavu sami). |
| **Zamítnutý** | Pokud schvalovatel defekt reportu uzná za vhodné, aby byl defekt report přepracován, tzn. doplněn o další informace, znovu ověřen nebo přeformulován. Defekt je zamítnut a předán zpět testerovi, který musí defekt report přepracovat a znovu přepnout do stavu *„Nový“* pro další schválení. |
| **Otevřený** | Pokud tester defekt kompletně charakterizoval a izoloval daný problém, schvalovatel defekt report posune do stavu *„Otevřený“*, kdy je viditelný všem členům projektového týmu. |
| **Přidělený** | Člen projektového týmu, který má na starost přidělování defekt reportů, přidělí defekt report na manažera vývojového týmu a ten jej předá k řešení vývojáři. Pokud vývojář nebo manažer vývoje zjistí, že je potřeba aby testeři defekt report doplnili, mohou jej předat zpět na testera k doplnění. |
| **Opravený**[26] | Jakmile vývojáři defekt opraví, předají ho testerům k potvrzení, že oprava proběhla v pořádku a defekt byl kompletně odstraněn. |

[25] Ve zdroji [Black, 2002, s. 133] je tento stav pojmenován jako *„Review“*. V češtině by to bylo přeloženo jako *„K revizi“*. Nicméně ve většině životních cyklů, se kterými jsem měla možnost se setkat, je výchozím stavem *„Nový“* a má obdobný význam. Proto jsem zvolila u výchozího stavu v této metodice místo pojmu *„K revizi“* pojem *„Nový“*.

[26] Ve zdroji [Black, 2002, s. 134] je tento stav pojmenován jako *„Test“*, což vyjadřuje, že byl opraven a předán testovacímu týmu k potvrzení opravy a k regresnímu testování. Zde jsem zvolila název *„Opravený“*, protože jinak by se stav musel jmenovat „ K testování“, což nezní dobře a v praxi se většinou setkávám s názvem *„Opravený“*.

| | |
|---|---|
| **Znovu otevřený** | Pokud testeři při potvrzení opravy defektu zjistí, že defekt přetrvává, nebo nebyl odstraněn kompletně, předají defekt report zpět na vývojáře, aby defekt kompletně odstranili. |
| **Uzavřený** | Jakmile jsou opravy defektů testery potvrzeny, je možné defekt report uzavřít. |
| **Odložený** | Pokud členové projektového týmu uznají, že se o defekt se skutečně jedná, ale má buď malou prioritu, nebo se jeho oprava naplánuje až na následující release[27], je defekt report odložen. Do stavu *„Odložený“* je možné přejít kdykoliv během životního cyklu. |

## 6.2 Hlavní aktivity a úlohy

Obrázek č. 6 zachycuje hlavní aktivity procesu testování počínaje plánováním testů až po uzavření testování.

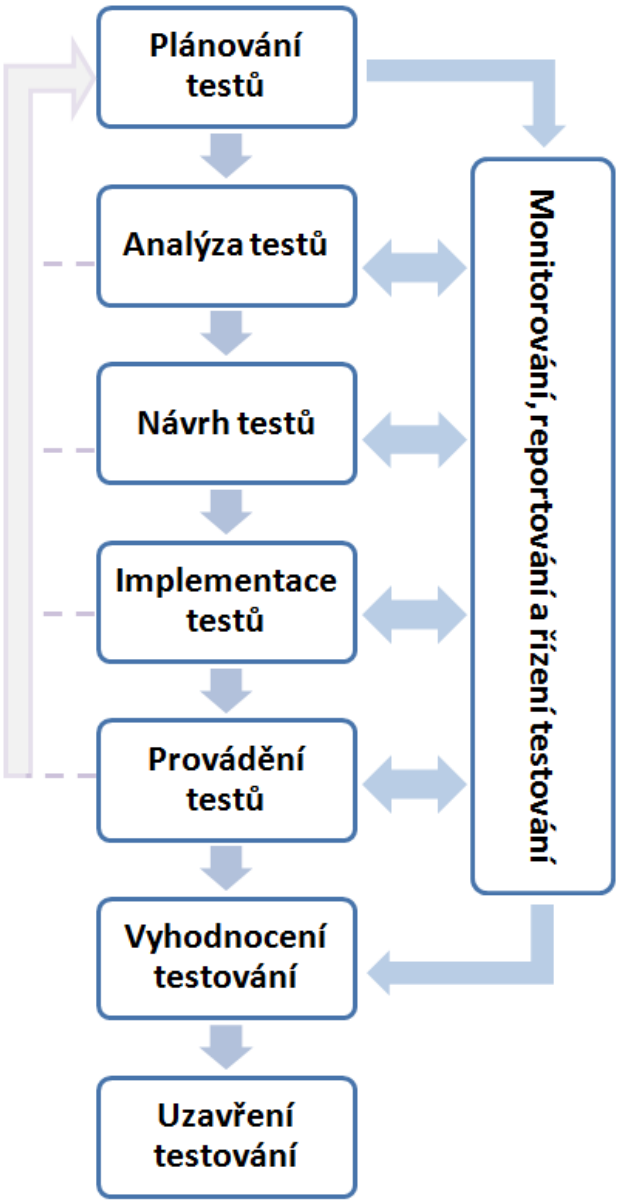
Ačkoliv nejviditelnější aktivitou procesu testování je provádění testů, je zapotřebí testování nejdříve naplánovat a připravit, aby bylo jasné, co se bude testovat, jak se k testování bude přistupovat, jak bude testování rozvrženo v čase a kdo bude za testování zodpovědný. Plánování, analýza, návrh a implementace testů zvyšují účinnost a efektivitu samotného provádění testů. Po provedení testů následuje vyhodnocení testování, kde je účelem posoudit, zda testování naplnilo stanovené cíle a uzavření testování, kdy se ověří, zda všechny testovací práce byly dokončeny a jejich výstupní produkty se zarchivují. V průběhu celého procesu testování probíhá monitorování, reportování a řízení testování a jsou revidovány, doplňovány a aktualizovány plány testování.

Tyto hlavní aktivity testování jsou seskupením úloh, což jsou nejmenší jednotky práce, které má smysl definovat jako samostatný proces.

V rámci každé hlavní aktivity je popsáno několik úloh, které dohromady naplňují poslání dané hlavní aktivity. V procesu testování jsou hlavní aktivity zakresleny v sekvenčním sledu. Nicméně v některých případech se mohou tyto aktivity částečně překrývat, mohou být prováděny souběžně, nebo mohou být prováděny opakovaně. Pokud je například v průběhu provádění testů nalezen defekt, který vyžaduje přidání nové funkcionality, je zapotřebí nejen provést testy na novou funkcionalitu, ale je potřeba nejdříve testy opětovně zanalyzovat, navrhnout a naimplementovat. Některé úlohy mohou být za určitých podmínek zařazeny do jiných hlavních aktivit, než jsou zařazeny v této metodice. Například příprava testovacích dat je primárně zařazena do implementace testů, ale někdy může být prováděna už při návrhu testů. Hlavní aktivity, které definuje tento proces testování, se

[27] **Release** je určitá verze nastavení položky, která byla vytvořena za nějakým konkrétním účelem (například pro účely testování). [ISO/IEC, 2008, s. 6]

vyskytují v určité míře na všech úrovních testování, nicméně na jednotlivých úrovních testování se liší úrovní formálnosti a množstvím dokumentace. Vývojáři, kteří jsou většinou zodpovědní za jednotkové testy, se jistě nebudou zabývat dokumentací plánů testování nebo vytvářením testovacích podmínek a testovacích případů, ale budou se soustředit primárně na provádění testů a vyhodnocení testů. Naopak na úrovni systémových testů, za kterou zodpovídá testovací tým, má plánování, analýza, návrh, implementace, vyhodnocení i uzavření testů svůj účel.




**Obrázek č. 6 Hlavní aktivity procesu testování, zdroj: inspirováno [ISTQB, 2012c, s. 9]**

Hlavní aktivity a úlohy v této metodice jsou podrobně popsány a jsou určeny primárně pro úroveň systémových testů, kde je zapotřebí proces testování vhodně organizovat a strukturovat. Konkrétní míra formálnosti procesu testování potom závisí nejen na úrovni

testování, ale také na kontextu testování softwaru v organizaci a rizicích, které se na software vztahují.

### 6.2.1 Hlavní aktivita plánování testů

Plánování testů je aktivita, jejímž hlavním cílem je vymezit záměry a cíle testování a identifikovat a naplánovat všechny aktivity a zdroje, které povedou k naplnění stanovených cílů. Plánování testů je vstupní bránou do procesu testování a mělo by začínat co nejdříve je to možné, ideálně již v době plánování projektu.

Výstupním pracovním produktem plánování testů je plán testování, viz kapitola 6. 5. 1. 1.

Ačkoliv je v rámci plánování testů kladen důraz zejména na začátek projektu, mělo by se jednat o průběžnou aktivitu, která provází proces testování od samého počátku až po kompletaci závěrečných aktivit. S postupem projektu je k dispozici čím dál více informací, detailnějších informací, mohou nastat i změny původních parametrů projektu nebo může být zapotřebí reagovat na probíhající proces testování. Je důležité, aby na tyto a mnohé další skutečnosti bylo včas reagováno a aby se v plánu testování udržovaly aktuální informace.

Primární rolí zodpovědnou za plánování testů je test manažer, přičemž na některých dílčích aktivitách se může podílet i test analytik. Příkladem těchto dílčích aktivit může být specifikace typů testů, nástrojů a testovacích technik, revize odhadů na testování, plánování testování konfigurací, plánování testování uživatelské dokumentace a instalačních procedur, identifikace rizik produktu nebo identifikace vazeb mezi podklady pro testování a pracovními produkty z procesu testování a vazeb mezi pracovními produkty z procesu testování navzájem.

Při plánování testování je třeba zohlednit celou řadu faktorů. Úkolem test manažera je zasadit proces testování do kontextu daného projektu tak, aby byl v souladu s potřebami a pracovními produkty ostatních zainteresovaných stran na projektu a aby respektoval model životního cyklu vývoje softwaru. S ohledem na zájmy zainteresovaných stran se definuje mise testování a na základě mise se stanoví konkrétní cíle testování. Pokud v organizaci existují dokumenty politika testování[28] a strategie testování[29], mělo by se z nich při plánování testů vycházet, nebo zdůvodnit případné odchylky. Zároveň by při plánování testů neměla být opomenuta analýza rizik, která může velmi významně přispět k definici přístupu k testování a k prioritizaci testovacích aktivit. Pro následné monito-

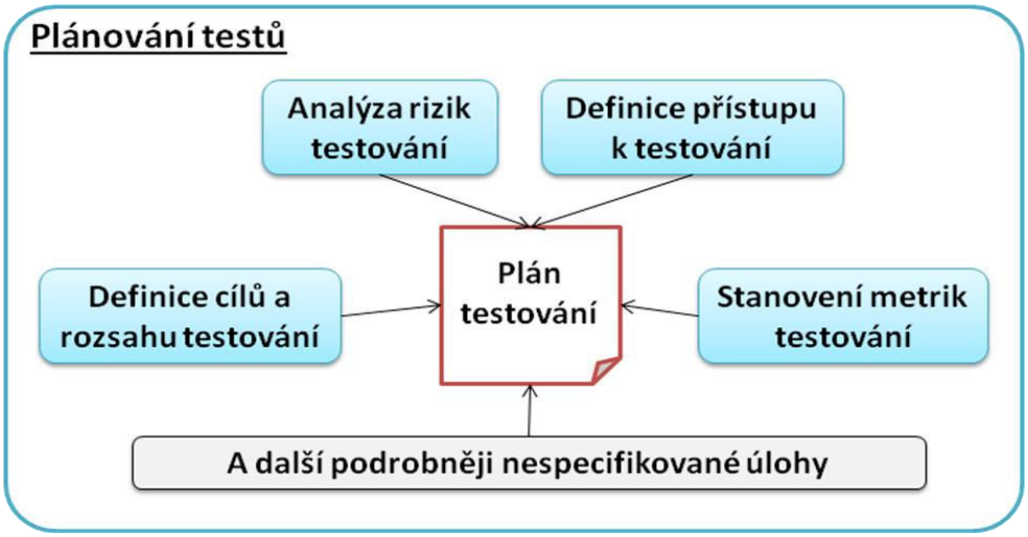
[28] **Politika testování** je dokument zaměřený na dlouhodobé cíle testování, za který odpovídá vrcholový management organizace (popř. vedení útvaru informatiky) [Doležel, 2013b] a popisuje principy, přístup a hlavní cíle organizace z pohledu testování [ISTQB, 2012, s. 46].

[29] **Strategie testování** je dokument zaměřený na střednědobé cíle testování, za který odpovídá quality manažer nebo test manažer (popř. metodik testování) [Doležel, 2013b] a popisuje jednotlivé úrovně testování,

rování a řízení procesu testování je rovněž zapotřebí nadefinovat vhodné metriky, kterými se ověří skutečný stav a cíle testování oproti plánovaným a jejichž hodnoty budou signálem pro nutnost zásahu do procesu testování ze strany test manažera.

Z definice cílů, rozsahu, přístupu k testování a metrik se poté odvíjí plánování dalších položek, jako je plánování testovacího prostředí, sestavení harmonogramů, rozpočtu a časových odhadů testování, specifikace vstupních a výstupních kritérií a kritérií pro přerušení testování a jeho následnou obnovu, plánování činností včetně určení zodpovědností za jednotlivé činnosti a pracovní produkty, plánování školení členů týmu, definice požadavků na dokumentaci pracovních produktů, určení způsobu hlášení defektů, určení vazeb mezi pracovními produkty a aktivitami týmu testování a produkty ostatních týmů projektu a plánování případných dalších pro daný projekt podstatných charakteristik. Jednotlivé úlohy v procesu plánování jsou vzájemně velmi úzce provázány, a proto je třeba na ně nahlížet jako na komplex, nikoliv jako na chronologický sled. V kapitolách 6. 2. 1. 1., 6. 2. 1. 2., 6. 2. 1. 3 a 6. 2. 1. 4 jsou charakterizovány čtyři hlavní úlohy plánování testů, na které ostatní úlohy plánování testů navazují, tj. definice cílů a rozsahu testování, analýza rizik testování, definice přístupu k testování a stanovení metrik testování.

Úlohy hlavní aktivity plánování testů ve vztahu k pracovním produktům zachycuje Obrázek č. 7. Šedivě jsou znázorněny další úlohy, které je třeba při plánování testů vykonávat (plánování testovacího prostředí, sestavení harmonogramů, rozpočtů a časových odhadů testování a další, viz výše), ale které metodika přímo nespecifikuje.




**Obrázek č. 7 Vztah úloh a pracovních produktů hlavní aktivity plánování testů, zdroj: [autor]**

ve kterých má testování v organizaci probíhat a jak má vypadat testování uvnitř jednotlivých úrovní [ISTQB, 2012, s. 47].

#### *6. 2. 1. 1. Úloha definice cílů a rozsahu testování*

Obecně základním cílem testování je *„ověřit, že se systém chová při svém použití cílovými zákazníky a uživateli tak, jak se od něj očekává“* [Doležel, 2013c] Testování tedy plní tři základní funkce:

- prokázání kvality a korektního chování,
- detekce případných problémů a umožnění následných oprav defektů,
- poskytnutí klíčových informací pro rozhodování managementu.

Doplňkovou funkcí je potom ještě shromažďování informací použitelných pro prevenci defektů do budoucna. [Doležel, 2013c]

Dle [Myers, 2004, s. 10] by se testování nemělo omezovat jen na potvrzení, že systém defekty neobsahuje. Naopak. Jednou z podmínek úspěšného testování, které přispěje ke zvýšení kvality softwarového produktu, je předpoklad, že software defekty obsahuje. Hlavním zájmem by tedy měla být snaha najít defektů co možná nejvíce.

Testování samo o sobě kvalitu softwarového produktu nezajistí. K tomu, aby testovací proces přinesl přidanou hodnotu, je zapotřebí konkrétní cíle a záměry testování sladit se všemi na projektu zainteresovanými stranami. Testování je pouze jedna součást z celkového množství prací na projektu. Test manažer by si měl být vědom potřeb, aktivit, pracovních produktů a priorit ostatních zainteresovaných stran a zároveň by měl zajistit, aby si i tyto strany byly vědomy hodnoty, kterou testování projektu přináší a snažily se ji naplnit. Tím je myšleno zejména včasné dodávání pracovních produktů a sestavení produktu, součinnosti při konfiguraci testovacího prostředí, opravy nalezených defektů a zamezení blokování testů kritickými defekty. Mezi zainteresované strany na projektu se typicky řadí skupiny vývojářů architektů, marketingových analytiků a byznys analytiků, vrcholový management, produktoví manažeři a sponzoři projektu, projektoví manažeři, technická a zákaznická podpora, help-desk a přímí a nepřímí uživatelé softwaru [ISTQB, 2012a, s. 18].

Při definici rozsahu testování musí test manažer vycházet především z projektového plánu, kde musí zohlednit zejména rozpočtová a časová omezení a omezení zdrojů na projektu.

Úkolem test manažera je určit seznam položek, které jsou předmětem testování a seznam testovaných, ale také netestovaných vlastností softwaru nebo systému, včetně odůvodnění, proč nejsou předmětem testování. Testovanými položkami jsou myšleny kromě samotného softwaru např. i instalační příručky, uživatelské příručky, příručky pro administraci systému, hardwarová rozhraní, apod. Testovanými vlastnostmi jsou myšleny takové vlastnosti, které jsou viditelné z pohledu uživatele, jako jsou funkce nebo procesy, oproti položkám, které jsou popsány technicky.

V reálných projektech je na testování často velmi omezený čas a rozpočet. Proto je potřeba se zaměřit zejména na prioritní položky a vlastnosti, které je reálné možné v rámci daných omezení otestovat. Určení priorit by mělo vycházet jednak z potřeb výše zmíněných na projektu zainteresovaných stran a jednak také by se na tomto místě měly zohlednit identifikovaná produktová rizika, na která se zaměřuje úloha analýza rizik testování, viz kapitola 6. 2. 1. 2.

#### *6. 2. 1. 2. Úloha analýza rizik testování*

V rámci plánování testů je úkolem test manažera správně vybrat, co se bude testovat, rozvrhnout testy a určit jejich priority. Vzhledem k tomu, že existuje prakticky nekonečný počet testovacích podmínek a jejich kombinací, je zapotřebí rozvrhnout konečnou množinu testů tak, aby objem úsilí vynaložený na pokrytí každé podmínky testovacími případy a pořadí, ve kterém budou jednotlivé testovací případy prováděny, byly co nejúčinnější[30] a nejefektivnější[31]. Jedním z faktorů, který by měl test manažer při prioritizaci testů vzít v úvahu, jsou rizika.

Riziko je možné definovat jako „*možnost, že nastane událost, nebezpečí, hrozba nebo situace, která má nechtěný dopad nebo vyústí v potencionální problém"* [ISTQB, 2011, s. 53]. Úroveň daného rizika je potom určena pravděpodobností, že nepříznivá událost nastane a jejím dopadem, tzn. škodou, která v důsledku této události vznikne. Rizika existují všude tam, kde se může vyskytnout nějaký problém, který sníží kvalitu produktu nebo úspěšnost projektu z pohledu zákazníka, uživatele, účastníka projektu nebo jiné zainteresované strany.

Podle primárního vlivu rizika se rozlišují produktová rizika (taktéž nazývaná jako rizika kvality produktu nebo rizika kvality) a projektová rizika (rizika plánování).

Produktová rizika jsou všude tam, kde je potenciální možnost selhání softwaru nebo systému. Může se jednat o následující [ISTQB, 2011, s. 53]:

- software, který je náchylný k selhání,
- možnost, že software nebo hardware způsobí škodu jednotlivci nebo společnosti,
- nedostatečná úroveň funkcionality, spolehlivosti, použitelnosti, výkonnosti či jiných vlastností softwaru,
- nedostatečná datová integrita a kvalita,

[30] **Účinnost *(efficiency)*** softwarového produktu je definována jako „*schopnost softwarového produktu poskytovat odpovídající výkon vzhledem k objemu zdrojů využívaných za určitých podmínek*." Účinnost procesu je „*schopnost procesu produkovat očekávaný výsledek vzhledem k objemu využívaných zdrojů"* [ISTQB, 2012c, s. 20].

[31] **Efektivita *(effectiveness)*** je definována jako *„schopnost produkovat očekávaný výsledek"* [ISTQB, 2012c, s. 19] V kontrastu k účinnosti je efektivita determinována bez vztahu ke zdrojům.

- software, který nefunguje tak, jak bylo původně zamýšleno.

Projektová rizika jsou taková rizika, která mají vliv na úspěch projektu, tedy na to, aby byly naplněny cíle projektu. Pod tímto si je možné představit [ISTQB, 2011, s. 53]:

- organizační rizika jako jsou např. dovednosti, školení, nedostatek pracovníků, osobní záležitosti, politické záležitosti nebo nesprávná očekávání od testování,
- technická rizika jako jsou problémy při definici požadavků, rozsah požadavků, které za daných omezení nemohou být splněny, nepřipravené testovací prostředí ve stanovený čas, opoždění konverze dat nebo nízká kvalita návrhu, kódu konfiguračních dat, testovacích dat a testů,
- rizika ze strany dodavatelů jako jsou selhání třetí strany nebo záležitosti ohledně smluv.

Přístup k testování založený na rizicích *(risk-based testing),* kterému je věnována kapitola 6.3.1., je zaměřen pouze na snížení úrovně produktových rizik. Projektová rizika, která tým testování identifikuje, jsou obvykle eskalována na projektového manažera, který se postará o snížení jejich úrovně. V silách test manažera nicméně může být snížení úrovně rizik, jako je připravenost testovacího prostředí a nástrojů, dostupnost kapacit pro testování a jejich kvalifikace, dostatek standardů, pravidel nebo technik pro testování, atp.

#### *6. 2. 1. 3. Úloha definice přístupu k testování*

Definice celkového přístupu k testování udává směr všem činnostem v rámci testování. Specifikuje úrovně testování, které jsou plánem testování pokryty, což v této metodice bude převážně pouze úroveň systémových testů. Dále definuje jeden nebo více přístupů, které budou následovány při výběru a prioritizaci testů, určuje typy testů, které budou prováděny, techniky pro návrh testů a mohou zde být uvedeny také nástroje pro automatizaci testů, příp. jiné nástroje pro podporu procesu přípravy a provádění testů.

S definicí přístupu k testování souvisí i definice vstupních a výstupních kritérií procesu testování.

**Vstupní kritéria** definují, kdy je možné zahájit danou úroveň testování, nebo kdy je sada testů připravena na spuštění. Typicky jsou vstupní kritéria zaměřena na pokrytí

- dostupnosti a připravenosti testovacího prostředí a nástrojů pro testování,
- dostupnosti kódu, který již může být otestován,
- dostupnosti testovacích dat. [ISTQB, 2011, s. 49]

Následují některé typické příklady vstupních kritérií pro úroveň systémových testů [Black, 2002, s. 55]:

- Jsou připraveny nástroje pro řízení testování a zaznamenávání nalezených defektů.

- Vývojový tým dokončil vývoj všech požadovaných vlastností systému a opravil všechny defekty plánované pro danou release.
- Vývojový tým provedl jednotkové testy všech požadovaných vlastností a opravených defektů pro danou release.
- Provozní tým provedl konfiguraci testovacího prostředí, včetně všech hardwarových komponent a subsystémů a poskytnul testovacímu týmu přístupy do těchto systémů.
- Testovací tým má připravena testovací data.

**Výstupní kritéria** definují, kdy je možné uznat testování za dokončené nebo kdy sada testů splnila stanovený cíl. Jsou obvykle napasována na

- metriky, jako je pokrytí kódu, funkcionality nebo rizik,
- odhady hustoty defektů[32] dané komponenty nebo systému,
- zbývající rizika v dané oblasti jako jsou dosud neopravené defekty nebo chybějící pokrytí dané oblasti testy,
- náklady,
- harmonogram – zejména tam, kde prioritou projektu je rychlé uvedení produktu na trh. [ISTQB, 2011, s. 50]

Následují některé typické příklady výstupních kritérií pro úroveň systémových testů [Black, 2002, s. 56]:

- Testovací tým provedl všechny plánované testy.
- Vývojový tým vyřešil všechny defekty s nejvyšší závažností a prioritou.
- Testovací tým zkontroloval, že všechny defekty v nástroji pro správu defektů jsou vyřešeny a retestovány, nebo odloženy do další release, nebo byly schváleny jako permanentní omezení a dále se řešit nebudou.
- Metriky testování ukazují, že bylo dosaženo stabilního a spolehlivého produktu, jelikož byly dokončeny všechny plánované testy a tyto plánované testy pokrývají všechna kritická rizika kvality produktu.
- Projektový manažer odsouhlasil, že systémové testy byly dokončeny.

[32] **Hustota defektů *(defect density)*** je definována jako *„počet defektů identifikovaný v komponentě nebo v systému dělený velikostí komponenty nebo systému (vyjádřené ve standardních výrazech pro měření jako jsou například řádky kódu, počet tříd nebo funkčních položek)“* [ISTQB, 2012c, s. 18].

Výstupní kritéria mohou být rozdělena na kritéria, která „musí“ být splněna a kritéria, které „by měla“ být splněna, což je důležité zejména pro vyhodnocení výsledků testování, viz kapitola 6. 2. 7. 1.

Zvolený **přístup k testování** na projektu, resp. kombinace několika přístupů, prostupuje celým procesem testování, pomáhá určit priority a pořadí různých aktivit testování a řídí se podle něj i výběr a priority jednotlivých testů.

Při použití přístupu k testování založeném na rizicích [ISTQB, 2012a, s. 9] jsou aktivity a testy zaměřeny na maximální pokrytí identifikovaných rizik. To znamená, pokud je například vysoká pravděpodobnost ohrožení závažnými bezpečnostními defekty, testování by se mělo zaměřit na bezpečnostní testy. Obdobně pokud je rizikem výkon systému, tak by se měl testovací tým zaměřit na výkonnostní testy a to, pokud možno, co nejdříve je k dispozici kód pro otestování.

Kromě přístupu k testování, který je založen na rizicích, ale existují i jiné přístupy.

Pokud je k dispozici kvalitní specifikace požadavků, ideálně včetně priorit jednotlivých požadavků, je možné využít přístupu založeného na požadavcích [ISTQB, 2012a, s. 30]. Ten může být dále doplněn vytvářením provozních profilů a profilů užití systému nebo přístupem založeným na modelech (např. UML modely, procesní modely nebo datové modely), což pomáhá k pochopení, jak se bude daný systém ve skutečnosti používat [ISTQB, 2012a, s. 30].

Jedním z možných přístupů je také reaktivní přístup [ISTQB, 2012a, s. 31]. To znamená nejvíce času a úsilí je věnováno provádění testů, kdežto analýza, návrh a implementace testů jsou značně omezeny. Testovací tým se soustředí na produkt, který je aktuálně nasazený na testovacím prostředí a snaží se najít co nejvíce defektů. Následně se testování soustředí zejména na oblasti, kde je defektů nejvíce a méně na oblasti, ve kterých je defektů málo. Použití výhradně tohoto přístupu je ovšem zrádné, protože může opomenout důkladné otestování některých důležitých oblastí, ve kterých se zdá, že je defektů málo, ale přitom právě tyto defekty a další dosud neobjevené defekty mohou mít zásadní dopad na uživatele systému.

Nelze říci, který z přístupů k testování je nejvýhodnější, protože každý má své výhody a omezení. Výběr přístupů, které budou při testování používány, závisí na kvalitě podkladů pro testování, na modelu životního cyklu vývoje softwaru, ale i na časových a nákladových omezení testování na projektu.

**Testy lze rozčlenit** dle účelu na následující **typy** [Doležel, 2013c]:

- testy funkcionality,
- testy nefunkcionálních charakteristik kvality,
- testy v souvislosti se změnami, tj. například

- testy pro potvrzení správnosti oprav defektů (retesty),
- testy pro potvrzení, že při změnách v kódu nejsou dotčené i části systému, které neměly být měněny a před změnami fungovaly korektně (regresní testy).

Jako pomůcka při určení prováděných typů testů může sloužit **metoda FURPS**, která se zaměřuje na ověřování funkcionálních[33] (funkcionalita) i nefunkcionálních[34] (použitelnost, spolehlivost, výkonnost, podporovatelnost) charakteristik kvality:

- **Funkcionalita *(Functionality)*** je dle [Faustová, 2009, s. 11] definována jako *„schopnost softwaru zabezpečit za předem stanovených podmínek určité funkce“*. Do funkcionálních testů se dle [IBM, 2005] zahrnuje testování sad funkcí systému, scénářů použití systému a testování bezpečnosti.
- **Použitelnost *(Usability)*** je dle [Faustová, 2009, s. 11] definována jako *„míra úsilí, které musí uživatel vynaložit, aby mohl software používat ke stanoveným cílům za stanovených podmínek“*. Testy použitelnosti se dle [IBM, 2005] zaměřují na lidské faktory, estetiku, konzistenci uživatelského rozhraní, online a kontextovou nápovědu, školící materiály a uživatelskou dokumentaci.
- **Spolehlivost *(Reliability)***, někdy překládána též jako bezporuchovost, je dle [Faustová, 2009, s. 11] definována jako *„schopnost softwaru být za daných podmínek používán, bez toho, aby došlo k jeho výpadku či chybě“*. Pro efektivní testování spolehlivosti jsou zapotřebí specializované nástroje. Do testů spolehlivosti se dle [IBM, 2005] řadí testy integrity, testy struktury, stress testy, testy konfliktů a kapacitní testy.
- **Výkonnost *(Performance)*** je dle [Faustová, 2009, s. 11-12] definována jako *„schopnost softwaru poskytovat požadovaný výkon vzhledem k použití stanovených zdrojů za stanovených podmínek“*. Do výkonnostních testů se dle [IBM, 2005] zahrnují testy výkonnostních profilů, benchmarking testy a zátěžové (load) testy.
- **Podporovatelnost *(Supportability)*** je dle [Faustová, 2009, s. 12] *„kritériem o přenositelnosti vyvíjeného softwaru do různých prostředí“*. Do tohoto typu testů dle [IBM, 2005] spadají testy instalací a konfigurací.

**Techniky návrhu testů** jsou vodítkem, co se přesně bude testovat, tedy pro co budou připraveny testy. Technik návrhu testů je celá řada a využívají se při identifikaci testovacích podmínek, při přípravě testovacích případů i při přípravě testovacích dat.

---

[33] **Funkcionální charakteristiky** kvality se vztahují na funkcionalitu software, tedy určení *“co má software dělat”*.

[34] **Nefunkcionální charakteristiky** kvality na rozdíl od funkcionálních charakteristik určují *“jak má software pracovat“* při vykonávání požadované funkcionality.

Základní způsob kategorizace technik rozlišuje techniky černé skříňky a techniky bílé skříňky [ISTQB, 2011, s. 39].

Techniky černé skříňky jsou založené na analýze podkladů pro testování a zahrnují funkcionální i nefunkcionální testování. Při návrhu testů technikami černé skříňky se nepoužívají informace týkající se vnitřní struktury komponenty nebo systému a daná komponenta či systém je zkoumána pouze skrze jeho vstupy a výstupy. Do této skupiny technik se řadí rozdělení tříd ekvivalencí, analýza hraničních hodnot, testování rozhodovacích tabulek, testování přechodů stavů a testování založené na případech užití nebo testování založené na příbězích užití.

Techniky bílé skříňky jsou naopak založené na analýze vnitřní struktury komponenty nebo systému. Do této skupiny technik se řadí testování příkazů, testování větví a testování rozhodovacích podmínek.

Techniky černé a bílé skříňky mohou být dále efektivně kombinovány s technikami založenými na zkušenostech a s technikami založenými na defektech.

#### *6. 2. 1. 4. Úloha stanovení metrik testování*

Známé manažerské moudro říká, že co nelze měřit, nelze ani řídit. Proces testování je potřeba měřit a řídit stejně jako jiné procesy, protože co není změřeno, často není uděláno a může být jednoduše ignorováno. Proto je důležité, aby byla sestavena sada metrik i pro proces testování. Metrika je dle [ISTQB, 2012c, s. 28] definována jako „*rozsah měření a metoda používaná pro měření*“. Pomocí vhodné sady metrik je možné kontrolovat, jakým směrem se testování ubírá z hlediska času, rozsahu, nákladů a kvality a včas zasáhnout, pokud jsou zjištěny odchylky od plánu testování. Používání metrik také testerům usnadňuje konzistentní reportování stavu testování na projektu ostatním členům projektového týmu a na projektu zainteresovaným stranám. Metriky testování jsou také obvykle prezentovány na projektových mítincích, kterých se účastní výkonný management.

Je důležité, aby metriky byly správně pochopeny a interpretovány. Nesprávná interpretace metrik může vést k chybným rozhodnutím a nenaplnění mise testování, strategie a cílů. Pro publikaci metrik jsou vhodné grafy, tabulky a různá schémata, ve kterých se lze snadno zorientovat a porovnávat jednotlivé hodnoty v čase nebo podle jiných parametrů.

Výběr konkrétních metrik a způsob jejich získávání, analýzy a reportování závisí na specifikách daného projektu a potřebách jeho členů. Při výběru metrik a stanovování jejich cílových hodnot je třeba zvážit, jak se daná měření využijí. Není účelem navrhnout vyčerpávající sadu metrik, které budou testovací tým jen zatěžovat a které nebudou mít pro řízení testování přínos, ale naopak navrhnout takovou kombinaci metrik, která s minimálním úsilím efektivně pokryje potřeby řízení testování.

Základní metriky, resp. skupiny metrik, které navrhuji sledovat v každém projektu, popisuje Tabulka č. 7.

**Tabulka č. 7 Základní metriky procesu testování,
zdroj: [Hutcheson, 2003; Doležel, 2013a; ISTQB, 2012a, Borovcová, 2008]**

| Metrika nebo skupina metrik | Popis |
|---|---|
| **Pokrytí požadavků testovacími případy** | Při návrhu testovacích případů jsou jednotlivé testovací případy propojovány s požadavky, které pokrývají. Metrika slouží ke sledování nárůstu pokrytí požadavků testy oproti časovému plánu a k finální kontrole, zda před zahájením provádění testů jsou všechny požadavky pokryty alespoň jedním testovacím případem. |
| **Stav provádění testovacích případů** | Při provádění testů jsou do test logů zaznamenávány výsledky jednotlivých testů. Z výsledků testů je vidět, kolik z připravených testovacích případů již bylo provedeno, kolik z nich skončilo úspěšně či neúspěšně a kolik testovacích případů ještě provedeno nebylo. Tyto informace slouží ke sledování pokroku testování oproti časovému plánu a přispívají k vyhodnocení kvality produktu. |
| **Hustota defektů na komponentu** | Při vytváření defekt reportů jsou jednotlivé defekty klasifikovány mimo jiné dle zasažené komponenty a závažnosti, viz kapitola 6.1.2. Rozložení počtu aktuálně otevřených defektů dle závažnosti a podle jednotlivých komponent produktu je užitečným nástrojem pro určování, kam přednostně soustředit úsilí a zdroje na další vývoj a testování na projektu. Terčem zájmu budou ty komponenty, ve kterých je ve srovnání s ostatními komponentami nejvíce defektů vysokého stupně závažnosti. |
| **Bug fix rate** | Nalezené defekty jsou v průběhu testování postupně opravovány. Bug fix rate vyjadřuje, kolik procent defektů bylo opraveno.<br><br>*Bug fix rate = (počet opravených defektů / počet nalezených defektů) * 100*<br><br>V praxi se místo s počtem opravených defektů často počítá spíše s počtem uzavřených defektů, což zahrnuje jak opravené, tak i zamítnuté defekty. [Borovcová, 2008, s. 66] Sledování trendu nalézání nových defektů a uzavírání nahlášených defektů pomáhá zjistit, jak rychle se defekty stíhají uzavírat a zdali jsou více nalézány nové defekty a nebo projekt spěje k vyřešení stávajících defektů a nové defekty už příliš nepřibývají. |

Při sestavování sady metrik konkrétního projektu lze vybírat z širokého spektra metrik různého zaměření. Podle zaměření lze metriky kategorizovat na projektové, produktové, procesní a metriky zaměřené na lidi [ISTQB, 2012a, s. 38]. Projektové metriky měří pokrok testování směrem ke stanoveným výstupním kritériím. Produktové metriky jsou zaměřeny na měření některých atributů produktu, jako je například hustota defektů. Procesní metriky jsou zaměřeny na měření účinnosti procesu testování, jako je například procento defektů detekované testovacím týmem. Metriky zaměřené na lidi hodnotí schopnosti a úsilí jedinců a skupin.

Jiným způsobem klasifikace metrik jsou dimenze, které mohou být předmětem monitorování. Pět hlavních dimenzí, na které se mohou vázat metriky, jsou produktová rizika, defekty, testy, pokrytí a důvěra [ISTQB, 2012a, s. 39]. Metriky založené na rizicích, defektech, testech a pokrytí jsou monitorovány a reportovány na základě objektivních informací z nástrojů pro řízení testování. Důvěra je měřitelná skrze průzkumy a jedná se o subjektivní pocity členů testovacího týmu z průběhu projektu a z kvality produktu, který testují.

Pokročilá měření mohou být zaměřena dále na hodnocení kvality práce testerů a vývojářů, rychlost nalezení a reportování defektů, efektivnost testu, procentuelní přínos testu, cenu hlášení defektu a další charakteristiky.

Podrobné vysvětlení zde uvedených metrik je možné nalézt např. v [Hutcheson, 2003] nebo v [Borovcová, 2008]. Pro načerpání inspirace pro pokročilá měření doporučuji článek „*Effective Test Metrics for Strategy Evolution“* [Chen, 2004], který shrnuje měření procesu testování na reálném projektu týmu testování IBM Electronic Commerce Development a analyzuje účinnost použité sady metrik. Měření procesu testování je zde zaměřeno na náklady, čas a kvalitu produktu.

### 6.2.2 Hlavní aktivita analýza testů

Analýza testů je aktivita, jejímž hlavním cílem je z obecných cílů a záměrů, které byly definovány v rámci plánování testů, sestavit konečnou množinu testovacích podmínek a určit tak, co se bude testovat. Analýza testů může začít v okamžiku, kdy je k dispozici plán testování, ve kterém je specifikován předmět a rozsah testování, a jsou vytvořeny podklady pro testování. Rozsah analýzy testů závisí na rozsahu testování na projektu, přiděleném rozpočtu, časových limitech a omezení zdrojů. Obzvláště na menších projektech, kde lze podklady pro testování jednoduše vztáhnout přímo na testovací případy, je možné analýzu testů omezit, či vynechat a přejít rovnou k návrhu testů. Často jsou také tyto dvě aktivity vykonávány souběžně.

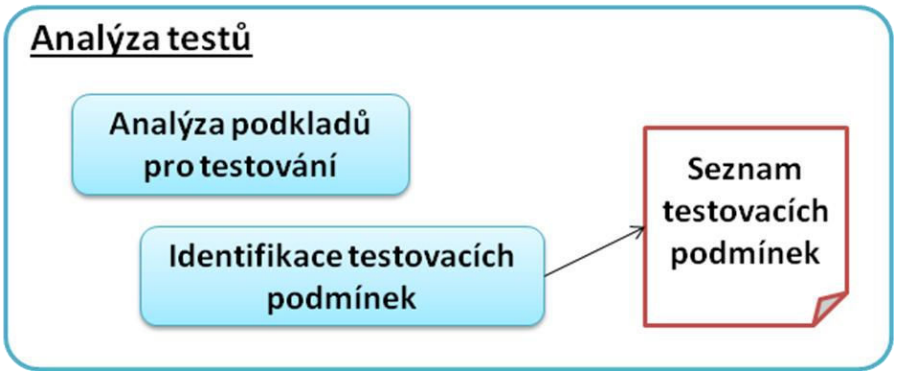
Výstupním produktem z analýzy testů je seznam testovacích podmínek, viz kapitola 6. 5. 2. 1.

Primární rolí, zodpovědnou za analýzu testů je test analytik. V případě, že není k dispozici dostatek podkladů pro testování, nebo jsou zastaralé, je nutné, aby test analytik

při identifikaci testovacích podmínek spolupracoval s odpovídajícími zainteresovanými stranami na projektu.

V rámci analýzy testů jsou v postupném sledu vykonávány úlohy analýza podkladů pro testování a identifikace testovacích podmínek.

Úlohy hlavní aktivity analýza testů ve vztahu k pracovním produktům zachycuje Obrázek č. 8.




**Obrázek č. 8 Vztah úloh a pracovních produktů hlavní aktivity analýza testů, zdroj: [autor]**

#### *6. 2. 2. 1. Úloha analýza podkladů pro testování*

Cílem analýzy podkladů pro testování je zrevidovat podklady pro testování a vyhodnotit, zda je možné dle těchto podkladů pro daný předmět testování připravit testy ve stanoveném rozsahu, které povedou k naplnění cílů testování a k pokrytí produktových rizik.

Podklady pro testování mohou být veškeré dokumenty, na jejichž základě je možné odvodit požadavky na software nebo systém a následně tak připravit odpovídající pracovní produkty pro testování. Můžete se jednat o specifikace požadavků, reporty z analýzy rizik, specifikace architektury a rozhraní, apod.

Revize podkladů pro testování rovněž přispívá ke včasnému odhalení případných defektů a nesrovnalostí již na úrovni podkladů. Včasná oprava defektů ještě v podkladech stojí projektový tým mnohem méně úsilí a nákladů, než kdyby defekty byly implementovány v kódu a zjištěny až při selhání v průběhu provádění testů. Kromě defektů mohou podklady obsahovat i nejednoznačné nebo nekompletní informace, které je také zapotřebí včas vyjasnit a sjednotit, aby jim všechny strany rozuměly stejně.

#### *6. 2. 2. 2. Úloha identifikace testovacích podmínek*

Jakmile jsou podklady pro testování zrevidovány, je možné přejít k identifikaci testovacích podmínek. Testovací podmínky v jednoduché formě popisují, co bude testováno a typicky jsou identifikovány na základě podkladů pro testování, cílů testování, strategických cílů a měly by se snažit pokrýt produktová rizika.

Stejně tak jako při plánování testů, i při identifikaci a prioritizaci testovacích podmínek je vhodné následovat některé z analytických přístupů jako je testování založené na rizicích, testování založené na požadavcích, nebo založené na případech užití. To, jaký přístup či kombinace přístupů bude pro identifikaci testovacích podmínek zvolena, závisí na přístupu, který je definován v plánu testování a na dostupnosti podkladů pro testování.

Kromě podkladů pro testování mohou jednotlivé testovací podmínky vycházet i z předchozích zkušeností s chováním a strukturou dané položky testování nebo ze zkušeností s obdobnými položkami testování. Například pro webové aplikace je charakteristické následující chování a pravidla:

- Naběhnou okamžitě po spuštění (zapsání URL adresy do prohlížeče) a není nutná jejich instalace.
- Logo aplikace je v levém horním rohu a odkazuje na úvodní stránku. Navigační menu je typicky po levém boku.
- Kromě samotné funkcionality aplikace je možné ovládat aplikaci i standardními prvky prohlížeče (tlačítko zpět a dopředu, zvětšení/zmenšení textu, zvětšení/zmenšení okna, otevírání jednotlivých stránek ve více záložkách nebo oknech).
- Výpočetní operace jsou prováděny na serveru, tudíž je zapotřebí vysoký výpočetní výkon serveru a na klientských počítačích není vysoký výpočetní výkon zapotřebí.
- Aplikace, které jsou v síti internet, jsou ohroženy na bezpečnosti - typickými defekty jsou např. nekontrolované vstupy od uživatele, použití neinicializovaných proměnných, kdy data z vnějšku mohou změnit obsah neinicializované proměnné, nebo chybějící ochrany session proti cross-site scriptování[35] a SQL injection[36] [Kosek, 2011].

Na závěr analýzy testů by test analytik měl vědět, jaké specifické testy musí být navrženy a s jakou prioritou, aby byly splněny cíle a uspokojeny potřeby testování na projektu.

---

[35] **Cross-site scriptování** je útok na weby, kde jsou zobrazována data vkládaná uživateli. Jedná se například o diskusní fóra, kam může útočník umístit odkaz, ve kterém je vložen škodlivý skript. Když potom ostatní uživatelé na tento odkaz kliknou, skript se spustí a může například zkopírovat cookies uživatelů a zaslat je danému útočníkovi. [Imperva, 2013]

[36] **SQL injection** je technika, která využívá neověření vstupních dat při dynamickém generování SQL dotazů na základě vstupů uživatelů. Útočník může vložit libovolný SQL příkaz jako vstupní data aplikaci. Aplikace se zachová, jako kdyby se jednalo o její vlastní příkaz a provede příslušnou operaci na backendu databáze. Útočník se tímto způsobem může dostat například k citlivým datům uživatelů nebo dokonce provést v databázi změnu.

### 6.2.3 Hlavní aktivita návrh testů

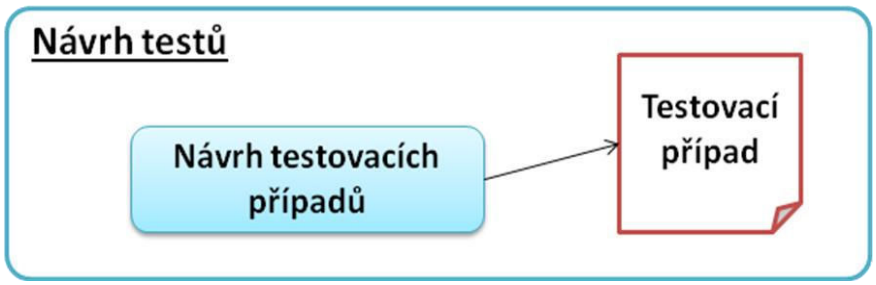
Návrh testů je aktivita, jejímž hlavním cílem je definovat, jak bude něco otestováno. Konkretizují se abstraktní testovací podmínky, které byly identifikovány v rámci analýzy testů, do podoby testovacích případů. Pro optimální pokrytí testovacích podmínek testovacími případy se používají techniky návrhu testů, identifikované v plánu testování. Návrh testů je buď separátní aktivita, která navazuje na aktivitu analýza testů, nebo je s analýzou testů sloučen. Na vyšších úrovních testování, jako je úroveň systémových testů, je nicméně vhodné tyto dvě aktivity oddělovat. K omezení, vynechání nebo sloučení analýzy testů by se mělo přistupovat pouze na menších projektech, kde lze podklady pro testování jednoduše vztáhnout na testovací případy. Do návrhu testů mohou být někdy také integrovány některé úlohy, které se jinak provádějí během implementace testů, jako je vytváření testovacích skriptů nebo příprava testovacích dat.

Výstupním produktem aktivity návrh testů jsou testovací případy, viz kapitola 6. 5. 3. 1.

Primární rolí, zodpovědnou za návrh testů je test analytik. Při návrhu testů postupuje v rámci rozsahu určeném v plánu testování a řídí se stanoveným přístupem k testování.

Hlavní úlohou v rámci návrhu testů je návrh testovacích případů. V průběhu návrhu testovacích případů jsou identifikována potřebná testovací data a také mohou být definovány detailní požadavky na infrastrukturu pro testování. Pod označením infrastruktura pro testování si lze představit vše, co je potřeba zabezpečit před prováděním testů, tedy například zajištění místností, vybavení, personálu, software, nástrojů, periferií, komunikačního vybavení nebo zajištění přístupů různých uživatelů. Nicméně samotná příprava testovacích dat a ověření, že infrastruktura pro testování je připravena pro provádění testů, je úlohou, která spadá do aktivity implementace testů. Na tomto místě je věnována zvláštní podkapitola 6. 2. 3. 1. pouze hlavní úloze návrh testovacích případů.

Úlohy hlavní aktivity návrh testů ve vztahu k pracovním produktům zachycuje Obrázek č. 9.




**Obrázek č. 9 Vztah úloh a pracovních produktů hlavní aktivity návrh testů, zdroj: [autor]**

#### *6. 2. 3. 1. Úloha návrh testovacích případů*

Jakmile jsou identifikovány testovací podmínky a je k dispozici dostatek informací a podkladů pro testování, je možné přejít k návrhu testovacích případů. Testovací případy specifikují, jak má být něco otestováno a je vhodné je přímo či nepřímo (přes testovací podmínky) vztáhnout na podklady pro testování, produktová rizika, strategické cíle a cíle testování.

Jako první krok před zahájením samotné tvorby testovacích případů je zapotřebí si ujasnit míru detailu testovacích případů v jednotlivých oblastech testování, protože ne ve všech oblastech testování a situacích musí být míra detailu stejná.

Konkrétní testovací případy (ve velkém detailu) poskytují veškeré specifické informace, data a skripty, které jsou potřebné při provádění testovacích případů a verifikaci výsledků [ISTQB, 2012b, s. 13]. Ve větší míře detailu jsou testovací případy navrhovány v případě, kdy požadavky na systém jsou dobře definované, testeři, kteří provádějí testy, nemají příliš zkušeností, nebo pokud jsou testovací případy předmětem auditu. Výhodou konkrétních testovacích případů je snadná ověřitelnost a reprodukovatelnost, kdy při provedení daného testu různými osobami v různém čase je dosaženo stále stejných výsledků. Nevýhodami jsou značný objem úsilí při vytváření a údržbě testovacích případů a potlačení flexibility testera při provádění testů.

Logické testovací případy (v malém detailu) poskytují pouze návody na to, co by mělo být otestováno, přičemž tester, který provádí testovací případ, může obměňovat testovací data i samotný skript přímo při provádění testu [ISTQB, 2012b, s. 13]. V menší míře detailu jsou testovací případy navrhovány v případě, kdy požadavky jsou špatně definovány, testeři, kteří provádějí testy, mají dostatek zkušeností s daným produktem i s testováním a pokud testovací případy nejsou předmětem auditu. Výhodou logických testovacích případů je možnost vyššího pokrytí testovacích podmínek, protože testovací případy se mohou modifikovat při každém spuštění. Větší flexibilita testera při provádění testů také umožňuje, aby se tester zabýval potenciálně zajímavými oblastmi, které v danou chvíli uzná za vhodné otestovat. Nevýhodou méně detailních testovacích případů je naopak malá reprodukovatelnost.

Problém s určením optimální míry detailu testovacích případů lze vyřešit postupným zpřesňováním testovacích případů od logických ke konkrétním. Při provádění testů se potom postupuje podle nejdetailnějších testovacích případů.

Druhým krokem před tvorbou testovacích případů je určení technik pro návrh testů, které optimálně pokryjí testovací podmínky. Technik, které je možné při návrhu testů využít, je celá řada. V kapitole 6. 2. 1. 3. byla uvedena jejich základní kategorizace. Na úrovni plánu testování jsou techniky identifikovány v rámci přístupu k testování. Úkolem test analytika je uvážit, které z technik použije při návrhu testovacích případů pro konkrétní oblasti a v případě potřeby identifikovat další techniky, které dosud v plánu testování identifikovány nejsou. Známými a na úrovni systémových testů často používanými technikami, jsou

techniky rozdělení tříd ekvivalencí a analýza hraničních hodnot. Jejich popisu jsou věnovány samostatné podkapitoly 6.3.2. a 6.3.3. Podrobný popis dalších technik lze nalézt například v [Singh, 2012].

Po určení míry detailu a technik pro návrh testovacích případů je možné přistoupit k samotnému vytváření testovacích případů. To, jak má testovací případ vypadat je popsáno v kapitole 6. 5. 3. 1. Zde uvádím doporučení, na která je vhodné při návrhu testovacích případů pamatovat:

- Testovací případy spolu s testovacími skripty by měly být výstižné a srozumitelné, aby je bez potíží mohly provádět i jiní testeři než jen autor testovacího případu. Nejen testerům, ale i ostatním zainteresovaným stranám by měly být testovací případy a skripty srozumitelné. Srozumitelnost je důležitá např. pro vývojáře, kteří při opravách defektů potřebují znát kroky, které předcházely selhání, ale také pro strany, které mají zájem na způsobu ověření kvality softwaru (auditoři, manažer produktu, projektový manažer, sponzor projektu, apod.).
- Při návrhu testovacích případů je třeba pokrýt různé interakce softwaru s jeho aktéry. Aktérem však nemusí být pouze koncový uživatel, ale i jiné systémy. Kromě defektů, které se objevují skrze selhání uživatelského rozhraní, mohou vznikat defekty při dávkovém zpracování, při komunikaci interních procesů, nebo na rozhraní několika objektů testování. Proto je důležité navrhnout testovací případy tak, aby i tato rizika byla pokryta.

Na závěr aktivity návrh testů by měl test manažer ověřit, zda navržené testovací případy vedou ke splnění výstupních kritérií testování na projektu a zda je příprava testů dostatečná k posunu do fází implementace a provádění testů.

### 6.2.4 Hlavní aktivita implementace testů

Implementace testů je aktivita, jejímž hlavním cílem je zkompletovat návrh testů a připravit infrastrukturu pro provádění testů. Testovací případy jsou rozšiřovány o manuální nebo automatizované testovací skripty, jsou připravována testovací data a následně jsou testovací případy a skripty organizovány do pořadí, ve kterém budou prováděny. Dále je zapotřebí ověřit korektnost nastavení testovacího prostředí a sestavit harmonogram provádění testů. Přestože ISTQB řadí vytváření testovacích skriptů a přípravu testovacích dat až do aktivity implementace testů [ISTQB, 2011, s. 16], někdy mohou být tyto aktivity prováděny souběžně s návrhem testovacích případů v rámci aktivity návrh testů. Jedná se zejména o projekty, kde množina testovacích případů, skriptů a dat není příliš rozsáhlá a nejsou využívány automatizované nástroje. Tam, kde je naopak součástí přípravy testů i automatizace testů, je častěji návrh testovacích případů od vytváření automatizovaných skriptů a dat oddělen.

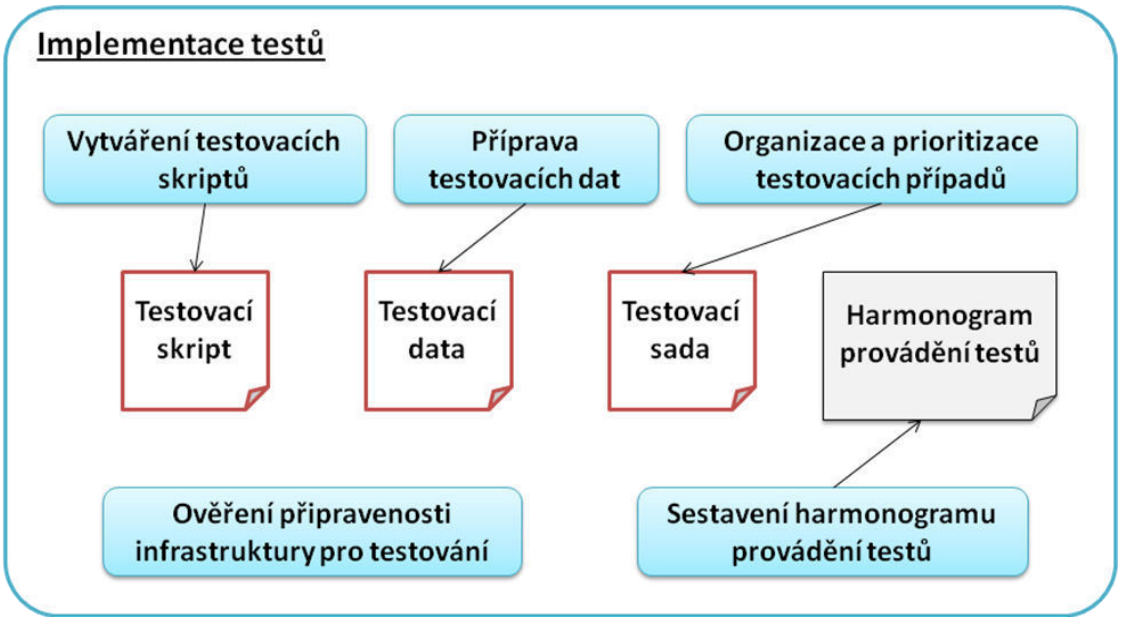
V rámci implementace testů je důležité také zkontrolovat, zda jsou splněna vstupní kritéria pro zahájení provádění testů a zda připravené testy splňují výstupní kritéria definovaná plánem testování. Pokud by kontroly splnění vstupních a výstupních kritérií byly vynechány, hrozí, že se při provádění testů narazí na problém, který bude vyžadovat neočekávané úsilí a povede ke zpoždění testování oproti plánovanému harmonogramu.

Výstupními pracovními produkty aktivity implementace testů jsou testovací skripty, testovací data a testovací sady, viz kapitoly 6. 5. 4. 1., 6. 5. 4. 2. a 6. 5. 4. 3. Během implementace testů je také vytvářen harmonogram provádění testů. Jeho podoba může být v závislosti na okolnostech projektu různá. Jde o to naplánovat, jak dlouho potrvá jeden testovací cyklus, kolik testovacích cyklů proběhne a jak budou na jednotlivé testovací cykly alokovány zdroje. Více se sestavování harmonogramu provádění testů věnuje kapitola 6. 5. 4. 4. Pracovní produkt harmonogram provádění testů nepovažuji za přínosné více rozebírat v samostatné kapitole jako ostatní pracovní produkty.

Za implementaci testů je primárně zodpovědný test analytik. Na přípravě testovacích dat se ale mohou podílet i zkušení testeři. Testeři samotní jsou také zodpovědní za ověření připravenosti infrastruktury pro testování. Při vytváření testovacích skriptů může být zapotřebí některé skripty automatizovat. Na vytváření automatizovaných skriptů je zapotřebí osob technického zaměření se znalostí skriptování a nástrojů pro automatizaci testů. Test analytik bez technického zaměření může požádat o pomoc s automatizací skriptů vývojáře, nebo povolat přímo specializované technické testery. Úlohou test manažera v rámci implementace testů je sestavení harmonogramu provádění testů a kontrola, že navržené testy splňují výstupní kritéria.

Pořadí úloh v rámci implementace testů není striktně dáno. Úloha vytváření testování skriptů většinou probíhá souběžně s úlohou příprava testovacích dat, ale mohou na sebe i navazovat. Úlohy organizace a prioritizace testovacích případů a sestavení harmonogramu provádění testů mohou probíhat souběžně s vytvářením testovacích skriptů, ale výsledný harmonogram testů by měl zohledňovat množství vytvořených skriptů a jejich závislosti. Na závěr implementace testů by mělo proběhnout ověření připravenosti infrastruktury pro testování.

Úlohy hlavní aktivity implementace testů ve vztahu k pracovním produktům zachycuje Obrázek č. 10. Šedivě je znázorněn pracovní produkt harmonogram provádění testů, který je vytvářen v úloze sestavení harmonogramu provádění testů, ale který metodika nespecifikuje.




**Obrázek č. 10 Vztah úloh a pracovních produktů hlavní aktivity implementace testů, zdroj: [autor]**

#### *6. 2. 4. 1. Úloha vytváření testovacích skriptů*

Při návrhu testovacích případů bylo určeno, které testovací případy budou konkretizovány do větší míry detailu a které zůstanou v menší míře detailu pouze jako návody na to, co by mělo být otestováno. K testovacím případům s požadavkem na větší míru detailu jsou následně vytvářeny testovací skripty, které specifikují jednotlivé kroky při provádění testovacích případů.

Vytváření testovacích skriptů spočívá v tom, že se v postupném sledu za sebou specifikují jednotlivé akce testera a jim odpovídající očekávané reakce systému či komponenty. Akce testera často spočívají v zadávání vstupních dat do systému. Vstupní data je možné zapsat přímo do jednotlivých kroků, nebo se z testovacího kroku odkázat na připravenou datovou sadu. Obdobně i data na výstupu systému mohou být buď přímou součástí kroku, nebo v podobě odkazu na datovou sadu.

Při vytváření testovacích skriptů je třeba se také zamyslet nad možností automatizace skriptů. Při rozhodování, zda skripty automatizovat, či vytvářet manuální skripty, je potřeba zvážit výhody a nevýhody.

Postoje různých expertů v oboru testování obecně k automatizaci testů se liší. Daniel Hoffman a spol., citován v [Singh, 2012, s. 494], uvádí:

*"The assurance of software reliability partially depends on testing. However, it is interesting to note that testing itself also needs to be reliable. Automating the testing process is a sound engineering approach, which can make the testing efficient, cost effective and reliable."*

V překladu:

*„Zajištění spolehlivosti softwaru částečně závisí na testování. Je však zajímavé si také povšimnout, že je třeba, aby testování samotné bylo spolehlivé. Automatizace procesu testování je rozumný inženýrský přístup, díky kterému je testování efektivní, nákladově úsporné a spolehlivé.“*

Oproti tomu Michael Bolton, citován v [Soundararajan, 2008], se k automatizaci testů staví následovně:

*"People write code to find bugs on another code. They write code that is buggy which claims to find bugs on another code."*

*"Writing test scripts is another development project. Most of them don't realize that they are running two development projects in parallel and hence their main development project suffers."*

V překladu:

*„Lidé píší kód, aby nalezli chyby v jiném kódu. Píší kód plný chyb a tvrdí, že nalézá chyby v jiném kódu.“*

*„Psaní testovacích skriptů je další vývojový projekt. Většina z těch lidí si neuvědomuje, že vedou dva paralelní vývojové projekty, a proto jejich hlavní vývojový projekt trpí.“*

S oběma těmito tvrzeními lze za jistých okolností souhlasit, ale i nesouhlasit.

Automatizace testování je efektivní a nákladově úsporná za předpokladu, že úsilí, náklady a čas vynaložené na vytvoření testovacích skriptů je menší než na manuální vytvoření a následné provádění testovacích skriptů. Vyplatí se tedy zejména u testů, které jsou dlouhodobě opakovány. U testů, které proběhnou pouze jednou, se investice úsilí, času a nákladů do automatizace skriptů příliš nevyplatí.

Spolehlivost automatizovaných skriptů je vyšší než u manuálních skriptů za předpokladu, že jsou skripty napsány správně. Nástroj, který spouští skripty, je na rozdíl od člověka neúnavný a dává stále stejné výsledky. Nevýhodou však je, že na rozdíl od člověka, který při provádění testů přemýšlí a experimentuje, nástroj pracuje pořád stejně, což může vést k tzv. paradoxu pesticidů. Software si vypěstuje odolnost proti navrženým testům podobně jako škůdci proti pesticidům a testy přestanou nacházet nové defekty. To ovšem neznamená, že by nové defekty nevznikaly. Mohou se objevit v těch částech, které nejsou pokryty testy. Paradoxu pesticidů je možné zabránit občasným obměňováním testů.

Je pravda, že kritickým faktorem úspěchu automatizovaných testů, jsou správně implementované skripty. Nástroje pro automatizaci testů obvykle mají funkci nahrávání a spouštění testovacích skriptů. Spouštění nahraných testů pak simuluje uživatele a může zahrnovat verifikační body, automatické logování průběhu testů a zaznamenávání defektů.

[Faustová, 2009, s. 39] V případě skriptů nahraných v nástroji pro automatizaci příliš nehrozí, že by skripty byly nekorektní. Problém může nastat v případě, kdy testerům pouhé nahrání nestačí a doplňují si automaticky vygenerované skripty o další funkce. Zde hrozí, že při vlastním kódování skriptů zanesou do skriptů defekty.

Nutno poznamenat, že volba mezi manuálními a automatizovanými testy se vztahuje na funkcionální typ testů. V případě, že má být daný systém či jeho komponenta podrobena výkonnostním nebo bezpečnostním testům, automatizace skriptů ve specializovaných nástrojích je nevyhnutelná.

Nástroj IBM Rational Quality Manager umožňuje vytvářet manuální i automatizované skripty. U automatizovaných skriptů nabízí možnost přímého napojení skriptů na jiné nástroje, ve kterých byly skripty vytvořeny, například napojení na IBM Rational Functional Tester, IBM Rational Performance Tester, IBM Rational Service Tester for SOA Quality, IBM Rational Robot a IBM Rational AppScan Tester Edition. [IBM, 2012]

#### *6. 2. 4. 2. Úloha příprava testovacích dat*

Součástí přípravy na provádění testů je mimo jiné příprava testovacích dat. V některých případech může být popis testovacích dat přímou součástí testovacích skriptů, ale většinou je lepší vytvářet testovacích skripty odděleně od testovacích dat. Sady testovacích dat mohou být rozsáhlé a je vhodné je připravovat pohromadě a následně se na ně z testovacích skriptů odkazovat. Testovací data mohou být připravována manuálně, nebo za podpory automatických generátorů dat, které je vhodné využít při vytváření rozsáhlých datových sad.

Automatické generátory dat mají tu výhodu, že jsou schopné ve velmi krátkém čase vytvořit množství dat podle zadaných kritérií a případně je i napumpovat do požadovaného datového zdroje. Další výhodou je, že jsou na rozdíl od lidí neúnavné, neztrácejí pozornost a šetří čas, který by testeři museli jinak strávit nad manuální přípravou dat. Naopak nevhodné jsou tam, kde je zapotřebí vytvořit data pro specifické situace nebo pro určitý cíl, jako je například prověření chování algoritmů v určitých situacích nebo při testování bezpečnosti a konzistence aplikace. V takových případech je pak nutná manuální příprava dat. [Fiurášek, 2010, s. 66-67]

K přípravě testovacích dat lze přistupovat různými způsoby. Příliš se nedoporučuje využívat reálná data, protože mohou být citlivého charakteru nebo pod ochranou zákona. Mnohem lepší způsob je použití upravených reálných dat, kde jsou nahrazeny všechny citlivé a chráněné údaje, ale struktura dat zůstává zachována a mohou být přidána i další data pro pokrytí extrémních situací. Kromě přímého využití nebo úpravy reálných dat, může být zvolen také přístup k přípravě testovacích tzv. na zelené louce. I při tomto přístupu je ovšem vhodné převzít alespoň strukturu dat z databáze obsahující reálná data. [Fiurášek, 2010, s. 65]

Přístup ke generování dat může být statický i dynamický. Při statickém přístupu není zapotřebí běh daného systému či komponenty, kdežto při dynamickém přístupu ano. Nejjednodušším způsobem generování testovacích dat je vygenerování náhodných dat bez ohledu na vnitřní strukturu systému a funkcionalitu zdrojového kódu. Existují ale i sofistikované nástroje, které dokážou automatizovat generování dat dle zvolených technik návrhu testů jako například rozdělení tříd ekvivalencí nebo analýza hraničních hodnot, viz kapitoly 6.3.2 a 6.3.3. [Singh, str. 496]

Specializovaných nástrojů pro automatické generování dat je celá škála. K dispozici jsou jak komerční generátory, tak i freeware či open-source generátory. Funkci na automatické generování dat mohou mít zabudovanou i CASE nástroje[37] a nástroje přímo pro datové modelování. Příkladem specializovaných generátorů dat jsou DTM Data Generator, Tnsgen, forSQL Data Generator, Test Data Generator TDG, DB Data Generator, GS Data generátor, MIDOAN. [Fiurášek, s. 66-67]

#### *6. 2. 4. 3. Úloha organizace a prioritizace testovacích případů*

V rámci návrhu a implementace testů je vytvářeno množství testovacích případů a skriptů. Na určení pořadí, v jakém se budou provádět, může mít vliv mnoho faktorů. Někdy má smysl uspořádat testovací případy do testovacích sad. To napomáhá k tomu, aby testovací případy, které mají vazbu na jiné testovací případy, byly prováděny v logickém sledu za sebou, nebo aby testovací případy byly organizovány do skupin, které mají společný účel (např. kolekce testů pro ověření sestavení, kolekce testů jedné platformy, testy pro jedno rozhraní, apod.). Při sestavování pořadí provádění testovacích případů je třeba také určit jejich priority a testovací případy s vyšší prioritou zařadit na začátek, aby v první řadě byly otestovány kritické vlastnosti a až potom vše ostatní.

V praxi se často stává, že čas původně vyhrazený na testování, je náhle zkrácen z důvodu zpoždění plánovaného nasazení verze dané komponenty či systému pro otestování. Jednou z cest, jak se s tímto problémem vypořádat, je postupovat při testování dle priorit a v případě, že se nestihne otestovat vše, co bylo původně připraveno, může testovací tým provést alespoň testovací případy s největší prioritou. Priority k jednotlivým testovacím případům jsou přiřazovány na základě přístupu k testování. Mohou být určeny prioritou jednotlivých rizik, které pokrývají, nebo na základě priorit pokrytých požadavků.

Pořadí provádění testovacích případů může záviset i na dalších faktorech, jako je dostupnost osob, od kterých je při testování vyžadována součinnost, dostupnost testovacího prostředí a nástrojů pro testování a dostupnost testovaného systému či komponenty. Vyvíjené komponenty nebo systém nemusí být nasazovány jako celek, ale po jednotlivých částech a organizace testů se potom musí takovým plánům nasazení přizpůsobit.

---

[37] **Case (Computer-aided Software Engineering) nástroje** jsou nástroje pro podporu procesu vývoje softwaru. Zahrnují nástroje pro podporu návrhu softwaru, správu požadavků, vytváření kódu, testování, dokumentaci a jiné aktivity softwarového inženýrství. [IEEE, 1990, s. 19]

#### *6. 2. 4. 4. Úloha sestavení harmonogramu provádění testů*

Při sestavování harmonogramu provádění testů je klíčové dobře odhadnout, jak dlouho potrvá jeden průchod všemi testy a kolik testovacích cyklů je zapotřebí provést [Black, 2002, s. 183]. Testovacím cyklem se dle [ISTQB, 2012c, s. 43] rozumí *„provedení procesu testování proti jedné identifikovatelné verzi testovacího objektu“*. Během testovacího cyklu je nasazena jedna stabilní verze testovaného objektu a nepovoluje se nasazení verze nové s výjimkou odstranění defektů, které blokují další testy [Doležel, 2013a].

Například pokud se předpokládá, že projít všemi testy potrvá jedné osobě celkem šest týdnů a jsou k dispozici tři testeři, tak testovacímu týmu potrvá provedení všech testů 2 týdny. Pokud jsou navíc zkušenosti z předchozích projektů, kdy k nalezení a opravení všech důležitých defektů bylo zapotřebí šest testovacích cyklů, harmonogram bude rozvržen do šesti jednotýdenních cyklů, ve kterých se provedou celkem tři průchody všemi testy. [Black, 2002, s. 183]

Ačkoliv počet testovacích cyklů, který vychází z minulých zkušeností, může být dobrým výchozím odhadem, tak skutečný počet cyklů závisí na mnoha faktorech, které test manažer nemůže ovlivnit. Může se stát, že bude potřeba provést více cyklů například z důvodu nedostatečné kvality softwaru, pomalých oprav defektů nebo pokud je tendence vývojářů opravách defektů zanášet defekty nové do již fungujících částí. [Black, 2002, s. 183]. Také se v průběhu provádění testování může stát, že systém či komponenta obsahuje kritické defekty, které blokují ostatní testy. Potom je nutné vybočit z původního harmonogramu, nasadit novou opravenou verzi k testování co nejdříve a spustit další testovací cyklus. Pro tyto případy by se při rozvrhování průběhu testů mělo počítat s časovou rezervou.

Při sestavování harmonogramu je také důležité brát v úvahu dostupnost testovaného systému či komponenty, protože není neobvyklé, když je kód nasazován po částech a provádění testů pak musí být rozvrženo v souladu s nasazováním objektů testování. Stejně tak je třeba zvážit i dostupnost osob, od nichž je při provádění testů vyžadována součinnost, dále dostupnost testovacího prostředí a nástrojů pro testování a dostupnost samotných testerů v daném čase. Například u kompetenčního centra může být právě dostupnost testerů dost podstatným faktorem, protože členové testovacího týmu jsou mimo aktivit v kompetenčním centru především studenty a musí skloubit aktivity pro kompetenční centrum s řadou školních povinností. Alokace testerů by se do harmonogramu provádění testů měla promítnout, aby měl každý člen testovacího týmu přehled, kdo a kdy bude testovat.

V nástroji IBM Rational Quality Manager je možnost vytvořit harmonogram provádění testů, kde jednotlivé úlohy mohou být prováděny postupně v naplánovaném čase nebo mohou být spouštěny automaticky ve stanovený čas, resp. být spouštěny nějakou událostí jako je například dokončení sestavení. [IBM, 2012]

#### *6. 2. 4. 5. Úloha ověření připravenosti infrastruktury pro testování*

Před zahájením provádění testů je potřeba ověřit, zda je veškerá infrastruktura pro testování připravena. Pod označením infrastruktura pro testování lze chápat vše, co je potřeba zabezpečit před prováděním testů, tzn. nejen software a podpůrné nástroje, ale řadí se sem i organizační aspekty jako je například zajištění místností, vybavení, personálu, komunikačního vybavení nebo zajištění oprávnění pro různé uživatele.

Je třeba ověřit stabilitu testovacího prostředí, tj. zda testovací prostředí umožňuje provádět alespoň základní operace. Pro tento účel mohou sloužit tzv. smoke testy, což jsou vybrané testovací případy, které pokrývají základní funkcionalitu komponenty nebo systému. Provedením smoke testů lze zjistit, zda je možné projít kritickými funkcemi programu bez ohledu na detailní funkcionalitu [ISTQB, 2012c, s. 39]. V případě, že smoke testy selžou, nemá smysl zahajovat provádění testů a je nutné nejdříve vyřešit základní problémy, jejichž příčinou může buď přímo nastavení testovacího prostředí, nebo závažný defekt v kódu. Testovací tým by měl rovněž zajistit součinnosti se správci testovacího prostředí v době provádění testů, aby případný výpadek testovacího prostředí v době provádění testů nezpozdil celé testování.

Dále je potřeba ověřit, zda jsou připraveny všechny nástroje pro podporu testování, což může zahrnovat nástroje pro správu procesu testování, logování výsledků testů a reportování defektů, nástroje pro spouštění automatizovaných skriptů, nebo jiné specializované nástroje např. pro výkonnostní nebo bezpečnostní testy.

Opomenuta by neměla být ani kontrola vytvořených testovacích případů, skriptů, sad a testovacích dat proti výstupním kritériím dané testovací úrovně.

### 6.2.5 Hlavní aktivita provádění testů

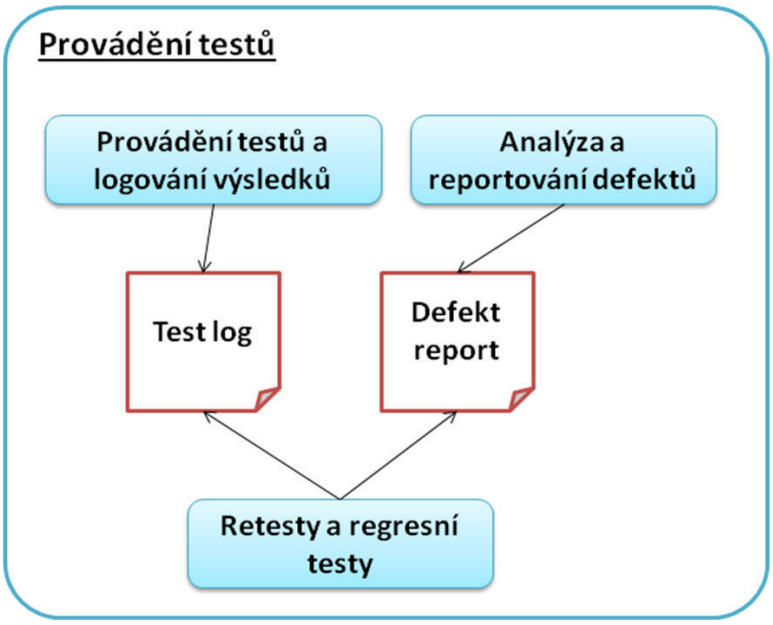
Provádění testů je ústřední aktivitou celého procesu testování. Bez provádění testů by všechny dosud popsané přípravné aktivity postrádaly smysl. Začíná, jakmile je systém nasazen pro testování a jsou splněna vstupní kritéria pro zahájení provádění testů. Testy jsou prováděny podle připravených testovacích sad, případů a skriptů, přičemž jsou logovány výsledky testů a reportovány nalezené defekty. Pro zvýšení efektivity testování je vhodné provádění skriptovaných testů doplnit ještě některou z neformálních technik jako je například průzkumné testování, viz kapitola 6.3.4. Při provádění testů se postupuje podle plánu a harmonogramu testování, výsledky testování jsou průběžně reportovány a monitorovány. Pokud se při plnění plánu a harmonogramu testování vyskytnou nějaké komplikace, je potřeba včas zasáhnout a usměrnit testování tak, aby spělo k naplnění definované mise, cílů a strategie.

Pracovními produkty, které vznikají v průběhu provádění testů, jsou test logy a defekt reporty, viz kapitoly 6. 5. 5. 1. a 6. 5. 5. 2.

Hlavní zodpovědnost za provádění testů nese role tester. Role test analytik přispívá svými nápady při provádění testů mimo připravené skripty. Jelikož testy připravoval, nejlépe ví, které testovací podmínky nejsou skripty zcela pokryty, a může neformální testování zaměřit právě na oblasti dosud nepokryté testy. Zde je třeba upozornit, že role není totéž, co osoba. Stejná osoba je často alokována jak na přípravu testování do role test analytika, tak na provádění testů do role testera. Test manažer má během provádění testů za úkol monitorovat a koordinovat průběh testování směrem k naplnění stanovených cílů.

Úlohami, které jsou vykonávány během provádění testů, jsou provádění testů a logování výsledků, analýza a reportování defektů a opakování testů v souvislosti se změnami v kódu a opravami defektů, tj. úloha retesty a regresní testy. Během úlohy provádění testů a logování výsledků testeři nacházejí defekty, které analyzují a reportují. Jakmile jsou defekty opraveny, je zapotřebí potvrdit, zda byl defekt korektně opraven a zda změna v kódu nezpůsobila jiné defekty v ostatních částech komponenty či systému.

Úlohy hlavní aktivity provádění testů ve vztahu k pracovním produktům zachycuje Obrázek č. 11.




**Obrázek č. 11 Vztah úloh a pracovních produktů hlavní aktivity provádění testů, zdroj: [autor]**

### *6. 2. 5. 1. Úloha provádění testů a logování výsledků*

Jádrem provádění testů je srovnávání aktuálního chování systému či komponenty s chováním očekávaným. Testy jsou prováděny podle připravených testovacích sad, případů a skriptů, přičemž výsledky jednotlivých testů jsou zaznamenávány do test logu, viz kapitola 6. 5. 5. 1.

Při provádění manuálních testů musí být testeři plně koncentrováni na srovnávání aktuálních a očekávaných výsledků, protože jinak by veškerá příprava testů byla marná. Ne-

pozornost může vést k přehlédnutí defektů a k nesprávnému vyhodnocení testu jako „prošlo“, nebo k nesprávnému vyhodnocení testu jako „neprošlo“ a následné reportování defektů, které ve skutečnosti defekty nejsou. Při spouštění automatizovaných testů se tester soustředí pouze na odchylky od očekávaného chování zalogované nástrojem, které prověřuje a kontroluje, zda příčinou odchylky je skutečně defekt v systému nebo jen chybně vytvořený testovací skript.

Smyslem logování výsledků testů je udržovat přehled o tom, které testy již byly provedeny a s jakými výsledky a které testy ještě zbývá dokončit. Na základě zalogovaných informací může test manažer monitorovat a reportovat průběh testování projektovému manažerovi, měřit výstupní kritéria a argumentovat případné zpoždění testování oproti původnímu harmonogramu. Testeři díky záznamům o provedení testu neopakují pořád stejné testy, které před nimi už provedl někdo jiný, ale soustředí se na testy, které ještě nikdo neprovedl.

Jakmile je v průběhu provádění testů nalezena nějaká odchylka od očekávaného chování systému, je třeba ji ihned prošetřit a pokud se potvrdí, že jde o defekt, je nutné založit defekt report, viz kapitola 6. 5. 5. 2.

Kromě provádění testů podle připravených testovacích sad, případů a skriptů je dobré vyhradit určitý čas i na otestování dalších nápadů, které vzniknou při pozorování chování systému během testování. Doplňkové použití reaktivních strategií, kde se využívá technik testování založených na zkušenostech a defektech, pomáhá obejít paradox pesticidů[38] a nalézt defekty v místech, která nejsou testovacími skripty pokryty. Příkladem často zmiňované techniky založené na zkušenostech je průzkumné testování, jemuž je věnována kapitola 6.3.4.

#### *6. 2. 5. 2. Úloha analýza a reportování defektů*

Účelem provádění testů je vyhledávání defektů. Tester by neměl předpokládat, že v systému vše funguje, jak má, ale naopak by se měl zaměřit na to, co je v rozporu s očekávaným chováním. Pokud tester na rozpor aktuálního a očekávaného chování narazí, je potřeba jej nejdříve důkladně prošetřit, určit, zda se skutečně jedná o defekt a v případě, že ano, je nutné jej nahlásit.

Často se stává, že testeři ihned po nalezení anomálie zakládají defekt report, aniž by prověřili, zda se jedná o defekt v systému, nebo jen o chybně provedený test. Takové defekt reporty potom zbytečně zatěžují vývojáře, kteří se marně snaží defekt navodit a když zjistí, že problém je v testu, označí defekt report jako neplatný. Ano, defekty je třeba oznamovat co nejdříve, ale ne ledabyle. Pokud si tester nepřeje, aby byl vývojáři vní-

[38] **Paradox pesticidů** se projevuje tak, že při opakování stále stejné množiny testů se zanedlouho přestávají nacházet nové defekty, protože systém se postupně vyladí tak, aby byl vůči těmto testům odolný. [ISTQB, 2011, s. 14]

mán jako nespolehlivý, měl by se zastavit a než založí defekt report, měl by si nejprve odpovědět na následující čtyři otázky [Software testing help, 2008]:

1. Co nefunguje?
2. Proč to nefunguje?
3. Jak zajistit, aby to fungovalo?
4. Jaké jsou možné důvody selhání?

Tester by se měl snažit vypátrat příčinu selhání a poskytnout vývojářům veškerá východiska pro opravu defektu. Pokud je vývojářům jasné, v čem defekt spočívá a ve kterých místech systému a jak se selhání projevuje, budou mnohem ochotnější defekt opravit, než když musí ještě zjišťovat, při jaké situaci přesně selhání nastává. A cílem testera je, aby defekty byly opraveny.

Jako první je třeba vyloučit defekt či omyl na straně testovacího týmu. To znamená zkontrolovat, zda je korektně nastaveno testovací prostředí, zda jsou korektní parametry samotného testu a zda kroky testu byly skutečně provedeny tak, jak je požadováno.

Často se stává, že se tester na provádění testů plně nekoncentruje a neprovede vše tak, jak je požadováno v testovacím skriptu. Dospěje k jinému výsledku testu než k očekávanému a neví, zda se jedná o defekt. Zde je třeba být ostražitý, selhání se pokusit znovu nasimulovat a ubezpečit se, že se jednalo jen o omyl při provádění testu a že při korektním provedení testu systém reaguje správně.

Jinými příčinami selhání, které ve skutečnosti nejsou defektem, mohou být například chybějící konfigurační soubory, chybějící tabulky v databázi, chyby v automatizovaných skriptech, neplatné přihlašovací údaje, neplatná verze systému nebo nesplnění požadavků na hardware a software. [Software testing help, 2008]

Aby testeři předešli nevalidním defekt reportům a ztrátě důvěryhodnosti ze strany vývojářů, je dobré na tyto možné příčiny selhání pamatovat a důkladně prošetřit, zda se nejedná o jednu z nich.

Dalším problémem, na kterém závisí oprava defektu, je reprodukovatelnost selhání. Reprodukovatelnost znamená, že je možné popsat, jak se program dostane do známého stavu a potom každý, kdo je obeznámen s programem může podle tohoto popisu postupovat a dostane se do daného stavu taktéž [Kaner, 1993, s. 79]. Tester by se měl vždy snažit reportovat defekty tak, aby byly reprodukovatelné. Pokud jsou pro vývojáře nereprodukovatelné, raději jejich opravu odloží na později nebo je ignorují úplně.

Někdy však tester neví, co způsobilo selhání a jak přesně selhání navodit. Nedokáže ho navodit znovu, aby popsal kroky, které k selhání vedly. Je třeba vycházet z předpokladu, že každé selhání je ve skutečnosti reprodukovatelné a i když se objevuje zřídkakdy, vždy nastává za stejných podmínek. Úkolem testera je odhalit tyto podmínky,

při kterých se selhání objevuje. V takové situaci je dobré si nejdříve sepsat, co všechno si tester pamatuje, že dělal, čím si je naprosto jistý a co jsou domněnky. Dále je třeba vzít do úvahy, co předcházelo sérii kroků vedoucích k selhání, včetně detailů. Pokud se stále nelze dopátrat příčiny selhání, je vhodné dále vzít v úvahu následující hypotézy [Kaner, 1993, s. 84-88]:

- Při prvotním navození selhání byly kroky testu prováděny rychle, kdežto při snaze o znovunasimulování selhání byl test prováděn pomaleji.
- Při provádění neskriptovaných testů se může snadno zapomenout na nějakou maličkost, kterou tester udělal a která je příčinou selhání.
- Tester nedělal to, co si myslí, že dělal a dokud nezopakuje stejný omyl, kterého se dopustil poprvé, selhání znovu nenastane.
- Selhání způsobí, že už není možné podruhé dospět do stejného stavu.
- Selhání je závislé na dostupnosti paměti.
- Selhání se může vyskytovat pouze před inicializací programu. Jakmile je program inicializován, selhání se přestane vyskytovat.
- Program mohou poškodit přímo jeho vlastní data na disku nebo v paměti, resp. chybná data může do programu zadat tester.
- Selhání může být vedlejším efektem ostatních problémů.
- Selhání může být způsobeno chvilkovým selháním hardwaru.
- Důvodem selhání mohou být časové závislosti jako například selhání pouze na přestupný rok nebo závislosti na zdrojích.
- Omyl nemá okamžitý dopad, ale projeví se až za nějaký čas.
- V kódu programu mohou být kritické podmínky, o kterých tester neví.
- Mezitím, co se tester na chvíli vzdálil od počítače, mohl někdo zadat do programu nová data, vypnout tiskárnu, apod.

Pokud ani tyto úvahy nestačí pro znovunavození selhání, potom je nutné defekt nahlásit jako nereprodukovatelný a snažit se ho přitom co nejpřesněji popsat. Nezbývá než doufat, že příčinu selhání vypátrá vývojář podle popisu.

Kromě potíží s reprodukovatelností defektů mohou testeři narazit také na komplikované defekty, kde k selhání vede dlouhá série kroků, nebo jejichž důsledky je těžké popsat. V těchto případech je třeba defekt report zjednodušit nebo ho rozdělit do více reportů. Cílem analýzy defektů je dle [Kaner, s. 79]:

- nalézt nejzávažnější důsledky problému, které vzbudí zájem defekt opravit,

- nalézt co možná nejjednodušší, nejkratší a nejobecnější podmínky, které vyvolají selhání,
- nalézt alternativní cesty k navození stejného problému,
- nalézt problémy s daným defektem související.

O tom, jak má vypadat defekt report, pojednává kapitola 6. 5. 5. 2. Zde uvedené postupy a náměty pro analýzu a reportování defektů je třeba brát jako užitečné rady a nikoliv jako dogma. Je samozřejmé, že čím precizněji jsou defekt reporty zapsány, tím lépe se budou vývojářům opravovat. Tester má ovšem na provádění testů vyhrazen často velmi úzký časový limit, ve kterém musí stihnout projít mnoho testů. Kdyby každé nalezené selhání do detailu analyzoval, nezbude mu čas na provedení ostatních testů. Je třeba se vyvarovat extrémům na jednu či druhou stranu a k analýze defektů přistupovat racionálně s ohledem na časová omezení testera. Pokud se například tester neorientuje dobře v logu aplikace, je efektivnější log pouze přiložit do defekt reportu, než aby tester trávil hodiny procházením logu a hledáním informace o selhání, kterou by vývojář našel během pěti minut.

#### *6. 2. 5. 3. Úloha retesty a regresní testování*

Nalezené defekty jsou v průběhu testování opravovány. Jakmile je k dispozici pro testování verze, kde jsou opravy defektů nasazeny, je třeba udělat retesty. Tester prověří, zda byly defekty správně opraveny a pokud je vše v pořádku, mohou defekt uzavřít jako vyřešený. V případě, že oprava není zcela v pořádku, předají defekt zpátky na vývojáře. Defekt je stále otevřen, dokud testeři nepotvrdí správnost opravy.

Kromě retestů se v souvislosti se změnami v systému provádí také regresní testy. Cílem regresních testů je ověřit, zda se v důsledku změn v kódu nezanesly defekty i do částí, které zasaženy být neměly. Vzhledem k tomu, že regresní testy by se měly opakovat při každém nasazení větších změn, je vhodné, aby byly automatizované.

### 6.2.6 Hlavní aktivita monitorování, reportování a řízení testování

Monitorování, reportování a řízení testování jsou aktivity, které jsou vykonávány v průběhu celého procesu testování. Ve fázi plánování testů je definována mise a cíle testování, je sestaven harmonogram testování a sada metrik, které budou v průběhu procesu testování monitorovány a pomocí kterých se bude kontrolovat, zda testování probíhá podle plánu.

V průběhu dalších fází testování se sleduje, měří a analyzuje, zda proces testování postupuje podle plánu. Výsledky sledování a měření pokroku testování jsou pravidelně reportovány uvnitř testovacího týmu i vně testovací tým projektovému manažerovi, vedoucímu vývoje a dalším na projektu zainteresovaným stranám. Pokud jsou zjištěny odchylky aktuálního průběhu testování proti původnímu plánu, je třeba provést korektivní akce a zajistit, aby testování vedlo k naplnění stanovených cílů. Informace o probíhajících akti-

vitách testování získané skrze monitorování a reportování testování mohou vést ke změnám původního plánu testování. Je zapotřebí, aby tyto změny byly průběžně do plánu testování zapracovávány, aby plán testování vždy udržoval aktuální informace.

Výsledky získané při monitorování testování jsou sumarizovány v pracovním produktu report o stavu testování, viz kapitola 6. 5. 6. 1.

Za monitorování a řízení testování je zodpovědný test manažer. Jeho úlohou je ve fázi plánování testů definovat rámec pro potřeby monitorování a následně sledovat vývoj procesu testování proti plánu testování a strategickým cílům. V případě zjištění odchylek testování oproti plánu provádí korektivní akce a v rámci své působnosti usiluje o to, aby testování naplnilo stanovené cíle. Za reportování aktuálního stavu testování uvnitř týmu testování jsou zodpovědní testeři a test analytici. Na jejich práci závisí veškerá měření. Získávají a zapisují data, ze kterých jsou počítány hodnoty metrik. Přesnost těchto dat je kritická, neboť z nepřesných dat jsou vytvářeny nepřesné trendové informace a ty mohou vést ke špatným závěrům a rozhodnutím ze strany test manažera. Za reportování mimo testovací tým je zodpovědný test manažer. Vychází z informací reportovaných členy testovacího týmu, které reviduje a sumarizuje pro potřeby projektového manažera, vedoucího vývoje a dalších zainteresovaných stran.

Úlohami, které jsou vykonávány v této fázi, jsou monitorování testování, průběžné reportování o stavu testování a řízení testování. Vazba mezi jednotlivými úlohami je taková, že během monitorování testování a průběžného reportování o stavu testování se získávají informace, které mohou dát impuls k provedení určitých řídících nebo korektivních akcí.

Úlohy hlavní aktivity monitorování, reportování a řízení testování ve vztahu k pracovním produktům zachycuje Obrázek č. 12.


**Obrázek č. 12 Vztah úloh a pracovních produktů hlavní aktivity monitorování, reportování a řízení testování, zdroj: [autor]**

#### *6. 2. 6. 1. Úloha monitorování testování*

Účelem monitorování testování je poskytnout pro řízení testování informace o probíhajících aktivitách testování. Monitorování prostupuje skrze celý proces testování od začátku do konce a získává aktuální hodnoty metrik, které slouží k hodnocení aktuálního pokroku testování oproti plánovanému.

Metriky mohou sloužit nejen pro monitorování průběhu aktivit samotného procesu testování, ale jsou často navázány i na milníky a vstupní a výstupní kritéria testování.

Typickou metrikou pro monitorování průběhu přípravy testovací dokumentace je například procento pokrytí požadavků testovacími případy [ISTQB, 2012a, s. 40]. Metriky pro monitorování provádění testů mohou zahrnovat například procento plánovaných testovacích případů, které byly spuštěny a kolik z nich prošlo úspěšně nebo neprošlo [ISTQB, 2012a, s. 40], informace o defektech jako je například hustota defektů, bug fix rate, nebo výsledky retestů, apod.

Příkladem metrik, které jsou mapovány na výstupní kritéria, může být počet požadovaných, akceptovaných, nasazených a otestovaných změn, plánované versus aktuální náklady, plánované versus aktuální trvání testování, plánovaná versus aktuální data milníků testování, atd. [ISTQB, 2012a, s. 41]

Základní metriky, které je vhodné monitorovat v každém projektu, jsou uvedeny v kapitole 6. 2. 1. 4.

Monitorování informací o testování by mělo být co nejvíce automatizováno, aby se ušetřil čas na měření a jeho zpracování. V nástrojích pro řízení testování jako je IBM Rational Quality Manager je monitorování usnadněno vzájemnou provázaností pracovních produktů. Plán testování může být navázán na požadavky a propojen testovacími případy a sadami. Testovací případy jsou propojeny s harmonogramem provádění testů, test logy a s defekty, které byly při provádění testů nalezeny. Při používání přístupu založeném na rizicích je vhodné také mapovat testovací případy na rizika, která pokrývají a následně zjišťovat pokrytí identifikovaných rizik testovacími případy. Díky vzájemnému provázání pracovních produktů je možné snadno sledovat většinu potřebných metrik a vytvářet různé trendy a grafy, které zachycují průběh testování a kvalitu produktu.

Automaticky naměřené hodnoty metrik musí být následně verifikovány, zda opravdu reflektují skutečný stav projektu. Někdy se může stát, že data zadaná do nástroje pro řízení testování neodpovídají skutečnému stavu, například pokud tester omylem zadá defekt s jinou závažností, než reálně odpovídá klasifikaci, viz kapitola 6.1.2. Měření potom mohou ukazovat příliš pozitivní nebo naopak negativní trend. Úkolem test manažera je analyzovat důvody rozdílů naměřených hodnot oproti očekávaným hodnotám ještě před tím, než stav testování prezentuje vně testovací tým nebo učiní nějaké rozhodnutí či opatření.

#### *6. 2. 6. 2. Úloha průběžné reportování o stavu testování*

Účelem reportování je předávání a zpřístupňování informací o stavu testování uvnitř týmu testování i vně testovací tým. Informace, které jsou reportovány, vychází z monitorování testování, kde jsou získávány a analyzovány naměřené hodnoty metrik. Cílem reportování je poskytnout klíčové informace o stavu testování pro cíle managementu ve srozumitelné podobě a na odpovídající úrovni detailu. Test manažer obvykle prezentuje stav testování na různých mítincích, kde se účastní mnoho zainteresovaných stran od technicky zaměřených pracovníků až po výkonný management, sponzora projektu a zákazníka. Pro prezentaci informací je vhodné využít především různé tabulky a grafy, kde je patrný vývoj metrik v čase. Při vytváření reportů je vhodné využít podporu nástrojů pro řízení testování, kde je obvykle možné automaticky vygenerovat různé trendové informace a grafy.

Pracovní produkt, kde jsou průběžně sumarizovány informace o stavu testování, se nazývá report o stavu testování a věnuje se mu kapitola 6. 5. 6. 1.

#### *6. 2. 6. 3. Úloha řízení testování*

Řízení testování je průběžná úloha, která zahrnuje veškeré řídící a korektivní akce, které jsou reakcí na získané informace a metriky. Smyslem řízení testování je zajistit, aby testování směřovalo k naplnění stanovené mise, strategie a cílů.

Pokud jsou zjištěny odchylky od původního plánu testování, je zapotřebí včas zasáhnout. Například vysoký počet defektů v jedné oblasti softwaru může indikovat potřebu dodatečného testování této oblasti. U zbylých oblastí testování se potom provedou pouze testy s nejvyšší prioritou určenou na základě pokrytí rizik a požadavků. Řízení testování musí reagovat jak na informace o průběhu samotného testování, tak i na měnící se podmínky projektu mimo testovací tým. Častou situací je například zkrácení časového limitu vyhrazeného na testování v důsledku zpoždění aktivit v úvodních fázích projektu. V těchto případech je testovací tým nucen realokovat své zdroje na testování a reprioritizovat testy tak, aby testování bylo co nejefektivnější. Řídící a korektivní akce mohou zasahovat nejen do testovacích aktivit v působnosti testovacího týmu, ale mohou zasahovat i do aktivit mimo testovací tým. V takových případech je zapotřebí, aby test manažer eskaloval nutnost provedení řídících zásahů na projektového manažera.

### 6.2.7 Hlavní aktivita vyhodnocení testování

V rámci plánování testů je definována mise a cíle testování ve vazbě na výstupní kritéria testování. Dále je sestaven harmonogram testování a rámec pro měření procesu testování. V průběhu procesu testování je třeba zajistit, aby členové testovacího týmu pravidelně poskytovali test manažerovi informace pro vyhodnocení míry splnění výstupních kritérií. Jakmile je ukončeno provádění testů, následuje vyhodnocení naplnění definovaných cílů a výstupních kritérií na základě získaných hodnot metrik a dalších informací

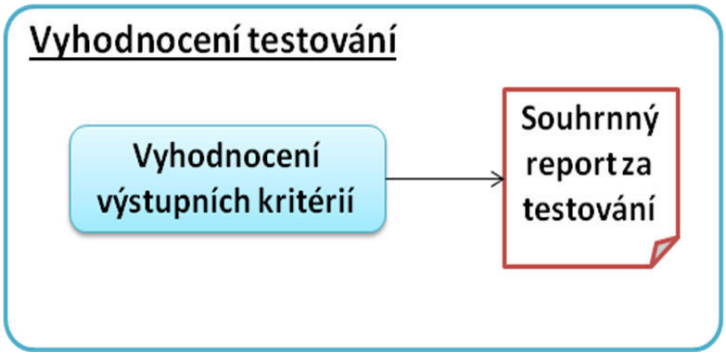
o výsledcích a průběhu testování. Po posouzení naplnění výstupních kritérií jsou souhrnné informace o aktivitách a výsledcích testování zpracovány do souhrnného reportu za testování a předány projektovému manažerovi a dalším na projektu zainteresovaným stranám.

Výstupním produktem z hlavní aktivity vyhodnocení testování je souhrnný report za testování, viz kapitola 6. 5. 7. 1.

Za vyhodnocení testování je zodpovědný test manažer. Jeho úlohou je ve fázi plánování testů definovat výstupní kritéria a rámec pro potřeby monitorování, následně sledovat vývoj procesu testování proti definovaným cílům a výstupním kritériím a po provedení testů vyhodnotit, zda byla výstupní kritéria splněna. Následně vytvoří souhrnný report za testování. Testeři a test analytici při vyhodnocování testování přispívají shromažďováním metrik z nástrojů pro řízení testování a jsou test manažerovi po ruce při posouzení celkového pokrytí a průběhu testů.

Součástí vyhodnocení testování je pouze úloha vyhodnocení výstupních kritérií, jejímž výstupem je souhrnný report za testování. Úloha vyhodnocení výstupních kritérií je popsána v kapitole 6. 2. 7. 1.

Úlohy hlavní aktivity vyhodnocení testování ve vztahu k pracovním produktům zachycuje Obrázek č. 13.




**Obrázek č. 13 Vztah úloh a pracovních produktů hlavní aktivity vyhodnocení testování, zdroj: [autor]**

### *6. 2. 7. 1. Úloha vyhodnocení výstupních kritérií*

Výstupní kritéria jsou definována v plánu testování. Mohou být rozdělena na kritéria, která „musí“ být splněna a kritéria, které „by měla“ být splněna. Může být například stanoveno, že pro postoupení do další úrovně testování „musí“ být otevřeno 0 defektů závažnosti blokující *(blocker)* a kritická *(critical)* a že 95 % testovacích případů by „mělo“ úspěšně projít testem. Pokud v tomto případě nebude otevřen žádný defekt závažnosti blokující a kritická, ale úspěšně projde pouze 93 % testovacích případů, projekt stále může postoupit do další úrovně. [ISTQB, 2012b, s. 18]

Důležité je definovat výstupní kritéria tak, aby mohla být objektivně vyhodnocena. Pro objektivní vyhodnocení je klíčové, aby test analytik zadával do nástroje pro řízení testů

přesná a konzistentní data. Pokud jsou například možné výsledky testovacích případů označeny s příznakem „prošlo“, „neprošlo“ nebo „prošlo s výhradou“, test analytik by měl jasně vědět, podle čeho určit, zda výsledek daného testovacího případu bude ohodnocen jako „neprošlo“ nebo prošlo s výhradou“. Ohodnocení výsledku testovacího případu je obzvláště kritickým faktorem, pokud je úspěšnost průchodu testovacími případy výstupním kritériem, které „musí“ být splněno. V případě, kdy si není test analytik jistý, jaký příznak k výsledku testovacího případu přiřadit, měl by se poradit s test manažerem.

Při vyhodnocování výstupních kritérií musí test manažer ověřit, zda naměřené hodnoty metrik skutečně reflektují realitu projektu. Kromě objektivně naměřených údajů by měl test manažer při vyhodnocování výstupních kritérií zvážit i subjektivní pocit testerů, kteří přišli reálně s produktem do styku. Subjektivní hodnocení produktu ze strany testerů může mnohé vypovídat o kvalitě produktu z pohledu uživatele.

Výsledky vyhodnocení výstupních kritérií jsou zachyceny do souhrnného reportu za testování, viz kapitola 6. 5. 7. 1. Při vytváření souhrnných reportů se postupuje obdobně jako při vytváření průběžných reportů v úloze průběžné reportování o stavu testování, viz kapitola 6. 2. 6. 2.

### 6.2.8 Hlavní aktivita uzavření testování

Jakmile je provádění testů vyhodnoceno a uzavřeno jako kompletní, měly by být zachyceny klíčové výstupy z testovacího procesu a předány relevantním osobám, nebo archivovány. Souhrnně jsou tyto aktivity označovány jako aktivity uzavírání testování.

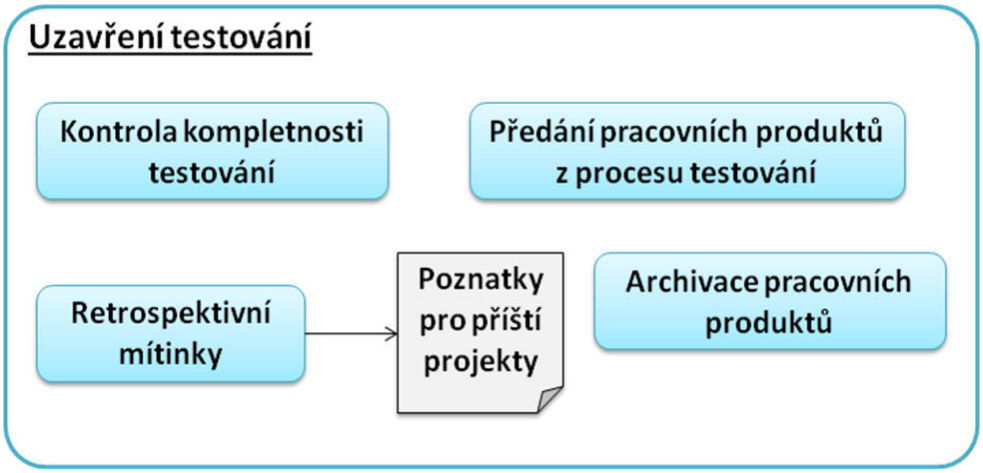
Výstupem z aktivit uzavírání testování jsou zdokumentované poznatky pro příští projekty. Vzhledem k tomu, že tento pracovní produkt je pouze zápisem důležitých poučení z retrospektivních mítinků, není striktně vyžadován, nemá specifickou formu a slouží jako znalostní báze pro následující projekty, není mu věnována samostatná kapitola, jako u ostatních pracovních produktů.

Za uzavření testování jako celek je zodpovědný test manažer. Kontroluje, zda je testování skutečně kompletní, je zodpovědný za předání pracovních produktů relevantním osobám, účastní se retrospektivních mítinků a rozhoduje o tom, jaké informace budou archivovány. Předání pracovních produktů za testování, stejně tak, jako archivaci dokumentů a pracovních produktů může test manažer delegovat na test analytiky a testery, kteří dokumenty a produkty vytvářeli. Test analytici a testeři, jakožto významné zdroje informací o projektu, se také spolu s test manažerem mohou účastnit retrospektivních mítinků, nebo alespoň zprostředkovávají relevantní informace test manažerovi.

Do uzavření testování se řadí čtyři hlavní úlohy. Jedná se o kontrolu kompletnosti testování, předání pracovních produktů z procesu testování, retrospektivní mítinky a Archivaci pracovních produktů. Popisu jednotlivých úloh se postupně věnují kapitoly 6. 2. 8. 1., 6. 2. 8. 2., 6. 2. 8. 3. a 6. 2. 8. 4. Na tyto úlohy procesu testování se často zapomíná.

Projekt je ukončen, členové týmu se musí věnovat následujícím projektům a na ukončený projekt již nejsou zdroje. Úlohy se vynechávají často z toho důvodu, že u nich není patrný přínos v krátkém období. V dlouhém období ovšem mohou ušetřit mnoho času a úsilí. Proto je vhodné na ně pamatovat a explicitně je zahrnout jako součást plánu testování, kdy proces testování neskončí v okamžiku vytvoření souhrnného reportu za testování, ale skončí, až bude ověřeno, že všechny testovací práce proběhly, pracovní produkty byly zaarchivovány nebo předány odpovídajícím osobám a byly zdokumentovány zkušenosti a poznatky z projektu pro následující projekty.

Úlohy hlavní aktivity uzavření testování ve vztahu k pracovním produktům zachycuje Obrázek č. 14. Šedivě je označen pracovní produkt poznatky pro příští projekty, který je vytvářen v návaznosti na retrospektivní mítinky, ale který metodika nespecifikuje.




**Obrázek č. 14 Vztah úloh a pracovních produktů hlavní aktivity uzavření testování, zdroj: [autor]**

#### *6. 2. 8. 1. Úloha kontrola kompletnosti testování*

Smyslem kontroly kompletnosti testování je ujištění se, že všechny testovací práce byly skutečně řádně dokončeny. Kontrolní akcí je například ověření, zda byly provedeny všechny plánované testy, nebo byly úmyslně vynechány. Podobně může být také ověřeno, že všechny nahlášené defekty byly opraveny a retestovány, nebo jejich oprava byla odložena na další release a nebo byly akceptovány jako permanentní omezení, které se dále řešit nebude. [ISTQB, 2012a, s. 15]

#### *6. 2. 8. 2. Úloha předání pracovních produktů z procesu testování*

Některé pracovní produkty z testování mohou být užitečné i osobám mimo testovací tým. Je třeba zajistit, aby pracovní produkty z testování byly předány těm, kteří je potřebují. Například známé defekty, které nebyly opraveny a jejich oprava byla buď zamítnuta, nebo byly akceptovány jako permanentní omezení, by měly být ohlášeny těm, kteří budou daný systém používat, nebo budou jeho použití technicky podporovat. Testy, testovací prostředí a regresní testovací sady by měly být předány týmu, který má na starosti údržbové

testování. Všechny informace o pracovních produktech z procesu testování musí být zdokumentovány, včetně případných odkazů a udělení přístupových práv. [ISTQB, 2012a, s. 15]

#### *6. 2. 8. 3. Úloha retrospektivní mítinky*

Testovací tým je jedním z článků celého projektu. Ovlivňuje a je ovlivňován děním projektu. Proto by se zástupci testovacího týmu měli spolu s ostatními členy projektového týmu účastnit retrospektivních mítinků, nazývaných jako „lessons learned“. Na retrospektivních mítincích se projednává, jaké praktiky se v projektu osvědčily a bude vhodné je využít i pro příští projekty a na druhé straně, jaké praktiky dobré nebyly, a v příštích projektech je třeba se jim vyvarovat. Tyto poznatky by měly být následně zdokumentovány a měly by být k dispozici při konání následujících projektů. [ISTQB, 2012a, s. 15]

#### *6. 2. 8. 4. Úloha archivace pracovních produktů*

Výsledky, logů, reporty, a ostatní dokumenty a pracovní produkty z testování se měly archivovat v configuration management systému[39]. Archivace je důležitá, obzvláště pokud informace o testování na projektu budou potřebné pro budoucí projekty. Zachování informací pro opětovné spuštění projektu v budoucnosti může ušetřit mnoho úsilí na začátku následujícího spuštění projektu. Pro tyto případy by v archívu měl být uložen alespoň plán testování a projektový plán, včetně provázání na systém a verzi, ve které byl daný produkt předán do produkčního prostředí, resp. posunut do další úrovně testování. [ISTQB, 2012a, s. 15]

## 6.3 Techniky

Technika dle [Chlapek, 2011, s. 8] *„určuje, jak se dobrat požadovaného výsledku. Zpravidla určuje přesný postup jednotlivých činností, způsob použití nástrojů, varianty rozhodnutí v určitých situacích a co z nich vyplývá, vymezuje obor své působnosti atd.“* V této metodice jsou pod pojmem technika chápány postupy a přístupy k tomu, jak dosáhnout určitého cíle, jako je snížení míry rizika, redukce množství testovacích případů nebo zvýšení účinnosti a efektivity testování. Metodika popisuje následující čtyři techniky:

- testování založené na rizicích,
- rozdělení tříd ekvivalencí,
- analýza hraničních hodnot,
- průzkumné testování.

[39] **Configuration management system** je systém, ve kterém jsou identifikovány a spravovány konfigurační položky, včetně řízení jejich změn a verzí. Konfigurační položky jsou seskupením hardwaru, softwaru nebo obojího. [ISTQB, 2012c, s. 14, 15]

Techniky, které budou v průběhu testování uplatňovány, jsou určeny v rámci plánování testů v úloze definice přístupu k testování, viz kapitola 6. 2. 1. 3.

### 6.3.1 Technika testování založené na rizicích

Smyslem přístupu k testování založenému na rizicích *(risk-based testing)* [ISTQB, 2012a, s. 23-26] je včasná identifikace a analýza rizik kvality produktu a následný návrh, implementace a provádění testů, které povedou ke snížení úrovně produktových rizik. Na kvalitu produktu je v tomto pojetí nahlíženo jako na souhrn vlastností, chování, charakteristik a atributů, které ovlivňují spokojenost zákazníka, uživatele nebo jiné zainteresované strany.

Testování založené na rizicích snižuje úroveň rizik kvality produktu, ať už odhalí defekty, nebo ne. Pokud testy odhalí defekty a testovací tým je ohlásí, snižuje se úroveň rizika už jen tím, že projektový tým o defektech ví a může je opravit. Pokud testy defekty neodhalí, snižuje se úroveň rizika potvrzením, že pod danými testovacími podmínkami systém pracuje korektně.

Hlavním cílem přístupu k testování založenému na rizicích je tedy maximálně využít testování ke snížení úrovně rizik kvality produktu a obzvláště ke snížení úrovně rizik na úrovni akceptačního testování.

Přístup k testování založený na rizicích může zjištěná rizika využít k různým účelům:

- k výběru testovacích podmínek, které budou předmětem testování,
- k výběru testovacích technik pro návrh a provádění testů,
- k určení rozsahu testování,
- k rozvržení testování v závislosti na prioritách testovacích případů tak, aby kritické defekty byly nalezeny pokud možno co nejdříve,
- k určení aktivit mimo oblast testování, které sníží úroveň rizik jako je například školení nezkušených analytiků.

Testování založené na rizicích zahrnuje čtyři hlavní aktivity:

- identifikace rizik,
- hodnocení rizik,
- snížení úrovně rizik,
- řízení rizik v životním cyklu.

Identifikace rizik kvality produktu není záležitost výhradně testovacího týmu, ale měla by probíhat ve spolupráci se zainteresovanými stranami, které očekávají určitou úro-

veň kvality produktu. Při identifikaci rizik mohou být využity různé techniky jako interview s experty, nezávislá hodnocení, použití šablon pro rizika, projektová retrospektiva, workshopy, brainstorming, seznamy, nebo minulé zkušenosti.

Jakmile jsou rizika identifikována, může se přistoupit k hodnocení rizik, kde se provede rozbor těchto identifikovaných rizik. Hodnocení rizik se sestává z kategorizace každého produktového rizika a určení jeho pravděpodobnosti a dopadu. Dále může také proběhnout vyhodnocení nebo přiřazení dalších vlastností rizika jako například vlastník rizika.

Ohodnocená rizika jsou poté zanesena do plánu testování a s ohledem na pokrytí rizik v plánu testování potom probíhá návrh, implementace a provádění testů. Testovací tým by si měl být vědom, že během projektu se může množina produktových rizik či jejich úroveň změnit. Měla by tedy probíhat soustavná identifikace a analýza rizik, jejíž výsledky by se měly promítnout i v odpovídajícím přizpůsobování testů a testovacích aktivit.

V organizacích na vyšších úrovních zralosti[40] řízení rizik prostupuje celým životním cyklem softwaru napříč projektovým týmem a ne jen testováním. Závažná rizika jsou podchycena již v rámci nižších úrovní testování vývojovým týmem a nedostanou se až na úroveň systémových testů k testovacímu týmu. Kromě potlačování důsledků potencionálních rizik je oblastí zájmu také analýza příčin rizik a implementace procesů, které rizikům předchází.

### 6.3.2 Technika rozdělení tříd ekvivalencí

Jedním ze základních principů testování je, že není možné otestovat úplně vše. Pokud je například vstupní hodnotou věk zákazníka, u něhož má aplikace akceptovat hodnoty od 0 do 120 let, nelze otestovat každou jednotlivou vstupní hodnotu, ale je zapotřebí počet testovaných hodnot zredukovat. Efektivním způsobem, jak počet testovacích případů snížit, je rozdělení hodnot do tříd ekvivalencí [ISTQB, 2012b, s. 27-29]. Rozdělení do tříd ekvivalencí spočívá v tom, že množina vstupních hodnot je rozdělena do skupin, od kterých se očekává, že se budou chovat stejným způsobem. Výběrem jedné reprezentativní hodnoty z této skupiny je potom zajištěno pokrytí i všech ostatních hodnot z dané skupiny. V případě zákazníka by vstupní hodnoty mohly být rozděleny například do následujících tří skupin:

- Zákazníci ve věku od 0 do 17 let (včetně), kteří budou mít slevu JUNIOR ve výši 30 % z plné ceny,
- zákazníci ve věku od 18 do 59 let (včetně), kteří budou platit plnou cenu,

[40] **Úroveň zralosti** je definována jako *„stupeň procesu zlepšování napříč předdefinovanou skupinou procesních oblastí, ve kterých je dosahováno všech cílů dané skupiny oblastí“* [ISTQB, 2012c, s. 28].

- zákazníci ve věku od 60 let, kteří budou mít slevu SENIOR ve výši 20 % z plné ceny.

Reprezentativními hodnotami z těchto skupin potom mohou být například hodnoty 15, 45 a 90.

Kromě těchto základních tří rozdělení, která testují pouze validní data, je vhodné přidat i třídy pro nevalidní data, která aplikace nemá akceptovat. V případě zákazníka mohou být přidány například následující třídy:

- Zákazníci ve věku méně než 0 let, které aplikace odmítne,
- zákazníci ve věku 120 a více let, které aplikace odmítne.

Reprezentativními hodnotami z těchto skupin potom mohou být například hodnoty -5 a 150. Další skupinou nevalidních dat potom mohou být typicky pokusy o zadání věku v nesprávném datovém formátu, ponechání prázdného pole, apod.

Přehledně tento příklad zachycuje Tabulka č. 8.

**Tabulka č. 8 Rozdělení tříd ekvivalencí, zdroj: [autor]**

| **Třída ekvivalence** | **věk < 0** | **0 ≤ věk > 18** | **18 ≤ věk > 60** | **60 ≤ věk > 120** | **120 ≤ věk** |
|---|---|---|---|---|---|
| **Očekávaný výsledek** | nepřijato | sleva JUNIOR | plná cena | sleva SENIOR | nepřijato |
| **Testovaná hodnota** | -5 | 15 | 45 | 90 | 150 |

Kromě vstupních hodnot se pomocí rozdělení do tříd ekvivalencí mohou roztřídit také výstupy, interní hodnoty, časově omezené hodnoty nebo rozhraní parametrů.

Ekvivalenční rozdělení je možné aplikovat na všech úrovních testování a je použitelné jak na lidské vstupy, tak i na vstupy skrze rozhraní systému nebo rozhraní parametrů při integračním testování. Pro test analytika je důležité zejména správně porozumět všem procesům, aby pak mohl sestavit co nejlepší třídy ekvivalence a z každé z nich vybrat jednoho zástupce.

Největší sílu má tato technika ve spojení s technikou analýza hraničních hodnot, která se zaměřuje speciálně na testování hodnot na okrajích jednotlivých rozdělení.

### 6.3.3 Technika analýza hraničních hodnot

Analýza hraničních hodnot [ISTQB, 2012b, s. 28-29] rozšiřuje techniku rozdělení tříd ekvivalencí. Používá se pro testování hodnot, které leží na okraji (hranici) ekvivalenčních rozdělení. Vychází z předpokladu, že mnoho selhání bývá způsobeno špatným ošetřením hodnot na hranicích intervalů nebo v jejich těsném okolí. Hraničními

hodnotami se rozumí maximální a minimální přípustné hodnoty daného ekvivalenčního rozdělení. Předmětem testování hraničních hodnot jsou hodnoty přesně na hranici rozdělení, těsně před hranicí a těsně za hranicí. U méně rizikových položek testování je také možné použít dvouhodnotový přístup, kde hodnoty těsně před hranicí se netestují a testují se pouze hodnoty na hranici a za hranicí daného rozdělení.

V příkladě se zákazníky uvedeném u techniky rozdělení tříd ekvivalencí, kde byli zákazníci rozděleni do několika intervalů podle věku, by například třída ekvivalence *„Zákazníci ve věku od 0 do 17 let (včetně)“* byla doplněna o testování následujících hodnot:

- Hranice minima: -1, 0, 1
- Hranice maxima: 17, 18, 19

Přehledně tento příklad zachycuje Tabulka č. 9.

**Tabulka č. 9 Hraniční hodnoty, zdroj: [autor]**

| **Třída ekvivalence** | **věk < 0** | **0 ≤ věk > 18** | **18 ≤ věk > 60** | **60 ≤ věk > 120** | **120 ≤ věk** |
|---|---|---|---|---|---|
| **Očekávaný výsledek** | nepřijato | sleva JUNIOR | plná cena | sleva SENIOR | nepřijato |
| **Testované hodnoty** | -1 | 0, 1, 17 | 18, 19, 59 | 60, 61, 119 | 120, 121 |

Při sestavování množiny testovaných hodnot je klíčové si ujasnit, jakých nejmenších možných přírůstků mohou hodnoty nabývat, protože ne vždy se musí jednat o celá čísla. Ve výši uvedeném příkladě je pro názornost věk považován za celé číslo, takže přírůstky jsou jednotlivé roky. Ve skutečnosti je věk ovšem spojitá veličina a přírůstkem by tak mohly být i jednotlivé dny. Hranice 18 let by potom musela být otestována hodnotami den před osmnáctými narozeninami, den osmnáctých narozenin a den následujícím po dni osmnáctých narozeninách.

Analýza hraničních hodnot je použitelná na všechny úrovně testování. Je to relativně jednoduchá technika na použití, ale přitom má poměrně vysokou úspěšnost při nacházení defektů. Obvykle se používá spolu s technikou rozdělení tříd ekvivalencí nebo s jinými technikami černé skříňky. Analýza hraničních hodnot nachází uplatnění nejen při testování uživatelských vstupů na obrazovce, ale např. i při testování časových rozmezí, rozsahů tabulek, kapacit paměti, apod.

### 6.3.4 Technika průzkumné testování

Průzkumné testování [ISTQB, 2012b, s. 38-39] je jednou z neformálních technik testování založených na zkušenostech a intuici testera. James Bach průzkumné testování defi-

nuje jako *"současné učení, návrh testů a provádění testů"* [2003, s. 1]. Testy nejsou tedy definovány v předem připravených testovacích případech a skriptech, ale jsou prováděny dynamicky na základě znalostí a zkušeností daného testera. Hlavním zdrojem znalostí testera jsou znalosti testovaného produktu. Pro testera je např. užitečné vědět, jak probíhaly předchozí testy produktu, jaké problémy se vyskytovaly, jaké jsou silné a slabé stránky produktu, jaké jsou obecné problémy a rizika platformy, na které je produkt nasazen, jaká rizika jsou spjatá s produktem, jaké byly poslední změny v produktu, co je podobné nebo odlišné oproti ostatním produktům, atd. [Bach, 2003, s. 4; IEEE, 2005, s. 5]

Správné průzkumné testování je interaktivní a kreativní. Přestože se během průzkumného testování nepostupuje podle předpřipravených skriptů, neznamená to, že by testování probíhalo chaoticky. Vhodným způsobem, jak řídit průběh průzkumného testování je rozvržení testování do několika mítinků, kde každý mítink je časově omezený a má určena pravidla, která vedou k naplnění poslání daného mítinku. Pravidla mítinku můžou zahrnovat například požadavky na otestování datových vstupů, kontrolu uživatelského rozhraní proti standardům platformy, integraci s externími aplikacemi, ověření výkonu a spolehlivosti produktu, nebo požadavky na otestování všech možných průchodů systémem. [Bach, 2003, s. 5] Pravidla také mohou definovat, co je a co není v rámci rozsahu mítinku, jaké zdroje jsou na mítink vyhrazeny nebo na jaký typ defektů by se testovací mítink měl zaměřit.

Testeři poté během určeného času interagují s produktem ve snaze naplnit předem definovanou misi a reportují nalezené výsledky. [Bach, 2003, s. 5] Na konci daného mítinku, resp. více mítinků, tester manažer svolá poradu, kde se shrnou výsledky testů a stanoví se pravidla pro další mítinky.

Problémem průzkumného testování je reprodukovatelnost nalezených defektů. V momentě, kdy k selhání vedl delší sled kroků, si tester musí vzpomenout, kterými kroky k selhání dospěl a znovu selhání nasimulovat, aby mohl založit reprodukovatelný defekt report. Pro tyto případy je vhodné využít automatizovaných nástrojů, které jsou schopny zaznamenat prováděné kroky a při dohledání cesty k selhání pak stačí spustit nahraný záznam kroků.

Průzkumné testování je vhodné používat jako doplněk ostatních formálních technik pro zvýšení efektivity skriptovaných testů. Často se využívá i v situacích, kde není k dispozici dokumentace dostatečná pro přípravu skriptovaných testů a průzkumné testování pak může posloužit, když je potřeba se seznámit vůbec s tím, jak produkt funguje. Dále může být průzkumné testování využito v situacích, kdy je časový tlak, není čas na přípravu skriptovaných testů a je vyžadováno, aby testovací tým poskytl rychlou odezvu na nový produkt nebo novou vlastnost, nebo když je potřeba prošetřit určité riziko či rychle prozkoumat a izolovat nějaký defekt. [Bach, 2003, s. 7]

## 6.4 Role

Role je abstraktním vyjádřením souboru zodpovědností za jednotlivé úlohy a pracovní produkty, které mohou být naplněny jednou nebo více osobami [IBM, 2005]. K jednotlivým rolím jsou přiřazeny kromě zodpovědností i schopnosti a dovednosti, které jsou pro řádné plnění zodpovědností vyžadovány. Konkrétním osobám je potom přiřazena jedna nebo více rolí a stejně tak jednu roli může vykonávat jedna nebo více osob.

V materiálech ISQTB rozdělení úloh a pracovních produktů testování podle jednotlivých rolí není dostatečně pokryto. V materiálu pro základní úroveň certifikace [ISTQB, s. 47-48] jsou úlohy testování přiřazeny pouze ke dvěma rolím, test leader a tester, přičemž role testera postihuje všechny výkonné činnosti od analýzy testů, návrhu testů, implementace testů až po provádění testů. Nejsou zde specifikovány znalosti a dovednosti, které jsou od rolí vyžadovány, ani způsob obsazení osob do jednotlivých rolí. Jelikož zodpovědnosti role tester podle ISTQB jsou dle mého názoru velmi rozsáhlé, inspirovala jsem se metodikou RUP [RUP, 2010] a rozdělila zodpovědnosti role tester do rolí tester a test analytik. Metodika RUP [RUP, 2010] je i z pohledu testování velmi robustní a definuje celkem čtyři role pro testování: Test manažer, test designer, test analytik a tester. V metodice RUP je pro role test designer a test analytik nadefinováno mnoho úloh, které tato metodika nepostihuje, protože jsem se rozhodla tyto dvě role sloučit do jedné. Zvolila jsem tedy střední cestu mezi ISTQB a RUP a nadefinovala pro testování celkem tři role: Test manažer, test analytik a tester. Odpovědnosti role test manažer odpovídají odpovědnostem role test leader v ISTQB a role test analytik a tester odpovídají dle ISTQB odpovědnostem role tester.

Popis těchto rolí má jednotnou strukturu. U každé role je uveden její hlavní účel, zodpovědnosti za jednotlivé úlohy a pracovní produkty, požadované schopnosti a dovednosti a způsob, jakým může být role na projektu obsazena.

### 6.4.1 Role test manažer

Test manažer má na starost vedení týmu testování v průběhu celého procesu testování. Je zodpovědný za kvalitu výsledného produktu, plánování a řízení zdrojů na projektu a řešení problémů, které stojí testovacímu týmu na cestě ke splnění cílů [RUP, 2010].

Na začátku každého projektu musí test manažer sladit zájmy testovacího týmu se zájmy a potřebami ostatních na projektu zainteresovaných stran, zejména projektového manažera, vrcholového vedení a zákazníků. S nimi se musí shodnout na misi testování. Jinak bude testování probíhat v případě systémů, na kterých závisí lidské životy a u kterých je nezbytné, aby byly perfektní, a jiný bude průběh testování v případě produktů, u kterých je prioritou být první na trhu před konkurencí.

Jakmile je odsouhlasena mise testování, může test manažer začít vytvářet plán testování. Smyslem plánování testování je zjednodušeně řečeno vydefinovat, co se bude testovat a jak, jakého výsledku má testování dosáhnout a na základě toho potom rozdělit práci

mezi jednotlivé členy týmu, sestavit předběžný harmonogram testování a stanovit metriky, pomocí kterých bude možné sledovat průběh testování. Dále by měl test manažer vzít do úvahy rizika testování, která mohou velmi významně přispět k definici přístupu k testování a k prioritizaci testovacích aktivit. Náležitosti plánu testování jsou popsány v kapitole 6. 5. 1. 1. Vzhledem k tomu, že plán testování se vytváří na počátku projektu, úskalím při vypracování plánu testování často bývá dostupnost informací. Test manažer by měl vycházet v první řadě z projektového plánu. Informace, které nejsou zapracovány do projektového plánu, je nutné explicitně vyžádat od ostatních členů projektového týmu.

Po odsouhlasení plánu testování projektovým manažerem následuje analýza, návrh, implementace a provádění testů. Úlohou test manažera v těchto fázích je sledovat průběh procesu testování a průběžně jej vyhodnocovat na základě aktuálních hodnot metrik. Naslouchat by měl ale i neformálním informacím od členů testovacího týmu. O průběžném stavu testování pravidelně informuje projektového manažera a ostatní na projektu zainteresované strany. Zastupuje testovací tým na projektových mítincích a usiluje o to, aby testování naplnilo stavené cíle. Pokud zjistí odchylky aktuálního stavu testování oproti plánovanému, pátrá po příčinách těchto odchylek. Následně provádí korektivní opatření. Příčiny zjištěných problémů mohou být uvnitř testovacího týmu, ale i mimo testovací tým. Pokud jde o problémy mimo působnost testovacího týmu, eskaluje je na projektového manažera, který provede odpovídající zásah.

V průběhu procesu testování je postupně k dispozici stále více informací o projektu a testování a během projektu může dojít ke změnám oproti původním předpokladům. Test manažer by neměl nechávat plán testování ležet v jeho výchozí podobě, ale měl by ho průběžně doplňovat a aktualizovat.

Po dokončení provádění testů test manažer vyhodnotí testování, zda naplnilo výstupní kritéria, vytvoří souhrnný report za testování a předá ho projektovému manažerovi a dalším zájemcům. Závěrečnými úlohami test manažera v procesu testování jsou kontrola kompletnosti testování, zajištění předání pracovních produktů odpovídajícím osobám, účast na retrospektivních mítincích a zajištění archivace pracovních produktů.

Úlohy, které test manažer vykonává a pracovní produkty, za které zodpovídá, znázorňuje Obrázek č. 15.


**Obrázek č. 15 Úlohy a pracovní produkty role test manažer, zdroj: [autor]**

Dle [RUP, 2010; Testování softwaru] by měl mít test manažer následující schopnosti a dovednosti:

- Všeobecnou znalost procesu vývoje softwaru,
- předchozí zkušenost s testováním, technikami testování a nástroji pro podporu testování,
- schopnost diplomatického vyjednávání a komunikační dovednosti,
- schopnost plánovat a řídit,
- nebát se učinit neoblíbená rozhodnutí,
- umění motivovat podřízené k precizní práci,
- znalost testované aplikace, systému nebo oblasti, která je předmětem testování (výhodou),
- základní znalost nějakého programovacího jazyka (výhodou).

Obvyklý přístup k přidělení role test manažera je přidělení této roli jedné osobě, aniž by daná osoba vykonávala další role. Tento přístup je vhodný jak u velkých týmů, tak u malých týmů, kde projektový manažer nemá příliš zkušeností s testováním. V malých týmech, kde projektový manažer má základní zkušenost s testováním je možné kumulovat roli test manažera s rolí projektového manažera. Častým modelem u malých týmů je kromě této kombinace také kumulace rolí test manažer a test analytik. V tomto případě se ovšem vyžaduje, aby daná osoba měla jak dobré schopnosti vedení a řízení týmu, tak zároveň i technické dovednosti. [RUP, 2010]

### 6.4.2 Role test analytik

Hlavní zodpovědností test analytika je příprava podkladů pro testery.

V rámci plánování testů je test analytik po ruce test manažerovi při specifikaci dílčích záležitostí. Pomáhá test manažerovi například se specifikací typů testů, nástrojů

a testovacích technik, reviduje odhady a pomáhá při identifikaci vazeb mezi podklady pro testování a pracovními produkty testování.

Jakmile je plán testování odsouhlasen, je zodpovědný za veškerou přípravu testování. Nejdříve analyzuje a reviduje všechny dostupné podklady pro testování. Pokud v nich narazí na nějaké defekty či nesrovnalosti, oznámí to autorům podkladů a ti ihned podklady opraví a upřesní. Na základě podkladů pro testování a odsouhlaseného plánu testování pokračuje identifikací testovacích podmínek, návrhem testovacích případů, vytvářením testovacích skriptů, přípravou testovacích dat a organizací a prioritizací testovacích případů. Při vytváření testovacích skriptů může být zapotřebí některé skripty automatizovat. K tomu je potřeba osob technického zaměření se znalostí skriptování a nástrojů pro automatizaci testů. Test analytik bez technického zaměření může požádat o pomoc s automatizací skriptů vývojáře, nebo povolat přímo na automatizaci specializované technické testery. Cílem test analytika je pochopit, jak má správně systém, resp. komponenta fungovat a efektivně jej celý pokrýt jednotlivými testovacími případy a skripty. Na cestě k optimálnímu pokrytí systému testy může využít různé techniky pro návrh testů, jejichž kategorizace byla uvedena v kapitole 6. 2. 1. 3.

Po kompletaci přípravy testování se proces testování posouvá do fáze provádění testů, která je v režii testerů. Test analytik ale může významně přispět při provádění testů mimo připravené skripty, protože má přehled o tom, co je připravenými testy pokryto a jaké oblasti dosud pokryty nejsou. Může testerům snadno poradit, na jaké oblasti se dále zaměřit, nebo otestovat tyto oblasti sám.

V průběhu celého procesu testování test analytik spolu s testery podává test manažerovi průběžná hlášení o stavu testování, což jsou v případě test analytika hlášení o stavu přípravy podkladů pro testery. Při přípravě jednotlivých podkladů pro testery musí dbát na pečlivé provázání jednotlivých podkladů mezi sebou a provázání podkladů na požadavky, aby se následně daly vysledovat metriky zaměřené na pokrytí.

Po dokončení provádění testů spolu s testery pomáhá test manažerovi se shromažďováním metrik z nástrojů pro řízení testování a s posouzením celkového pokrytí a průběhu testů. Na závěr testovacích aktivit může být pověřen předáním pracovních produktů testování relevantním osobám a týmům, archivací pracovních produktů a může být přizván jako důležitý zdroj informací na retrospektivní mítinky.


**Obrázek č. 16 Úlohy a pracovní produkty role test analytik, zdroj: [autor]**

Úlohy, které test analytik vykonává a pracovní produkty, za které zodpovídá, znázorňuje Obrázek č. 16.

Dle [RUP, 2010; Testování softwaru] by měl mít test analytik následující schopnosti a dovednosti:

- předchozí zkušenost s testováním, technikami testování a nástroji pro podporu testování,
- analytické schopnosti a schopnost diagnostikovat a řešit problémy,
- být pečlivý, vytrvalý a zvídavý a věnovat pozornost i detailům,
- znalost výhod a nevýhod automatizovaných testů, aby se mohl rozhodnout, které testy budou vytvořeny manuálně a které automatizovaně (vytvořením automatizovaných testů může následně pověřit specializované technické testery),
- znalost obvyklých softwarových chyb a selhání,
- znalost programovacího jazyka, ve kterém jsou psány testované aplikace (výhodou),
- znalost testované aplikace, systému nebo oblasti, která je předmětem testování (velkou výhodou).

Požadavky na specifické dovednosti se liší i dle typu testů, které se budou provádět. Pokud jsou požadovány kromě klasických funkcionálních testů také například testy výkonnosti a bezpečnosti, je zapotřebí osob, kteří mají s daným typem testů zkušenosti.

V malých testovacích týmech se zkušenými členy týmu je typicky role test analytika přidělována více osobám společně s rolí testera. Pokud jsou vytvářeny automatizované skripty, je dobrým zvykem, že techničtí testeři skripty nejen vytváří, ale následně i spouští a zaznamenávají jejich výsledky. Častým přístupem k přiřazení rolí v menších týmech je také přidělení role test analytika navíc k roli analytika. Nevýhodou však zde je, že podklady pro testování vytvářené analytikem neprojdou revizí od nezávislé osoby a defekty zanesené v analytických dokumentech se tak snadno zapracují i do testů. Naopak výhodou je, že analytik zná dokonale danou oblast a nemusí ji při přípravě testování explicitně studovat. Jak již bylo zmíněno u role test manažer, častým modelem u malých týmů je také kumulace rolí test manažer a test analytik. [RUP, 2010]

### 6.4.3 Role tester

Hlavní zodpovědností testera je provádění testů a zaznamenávání jejich výsledků. [RUP, 2010]

Role testera se do procesu testování zapojuje až ve fázi implementace testů, kde pomáhá připravovat testovací data a ověřuje, zda infrastruktura pro testování je připravena

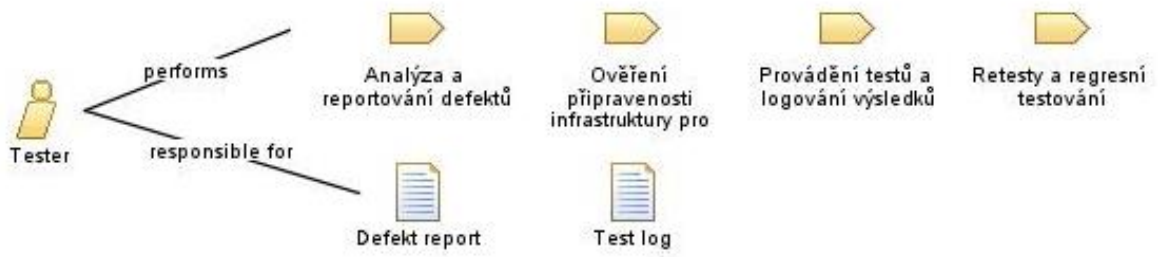
na zahájení provádění testů. Někteří testeři mohou být specializovaní na automatizaci testů. Do jejich zodpovědnosti pak přirozeně spadá nejen provádění testů, ale i vytváření automatizovaných skriptů.

Jakmile jsou splněna vstupní kritéria pro zahájení provádění testů, úlohou testerů je provádět testy podle připravených testovacích případů, sad a skriptů, zaznamenat jejich výsledky do test logu, ohlásit nalezené defekty a následně opakovat testy v souvislosti se změnami v kódu a opravami defektů. Při procházení testů podle připravených testovacích případů by tester měl zapojit kreativitu a nejen splnit povinnost „odklikat“ test podle návodu. Dobrý tester rád vyzkouší i věci navíc, nad rámec testovacího případu. Někdy je v rámci provádění testů přímo vyhrazen čas na provádění testů mimo připravené testovací případy, sady a skripty. Uplatňují se reaktivní strategie testování, kde se využívá technik testování založených na zkušenostech a defektech. Účinnost reaktivních strategií přímo závisí na intuici, zkušenostech a znalostech daného testera. Pokud jsou testeři schopni najít defekty v oblastech dosud nepokrytých připravenými testy, je to velkým přínosem pro rozhodování o dalším postupu testování.

V průběhu celého procesu testování tester, stejně jako test analytik, podává test manažerovi průběžná hlášení o stavu testování, což jsou v případě testera hlášení o průběhu a stavu provádění testů. Při zaznamenávání výsledků a reportování defektů musí tester vyhodnocovat, zda testovací případ označit jako „prošlo“,“neprošlo“, nebo „prošlo s výhradou“, jaký je stupeň závažnosti a priority nalezeného defektu a další informace. Je důležité, aby testeři zadávání těchto dat nepodceňovali a zadávali tato data přesně a konzistentně, protože na jejich základě jsou potom počítány hodnoty metrik a monitorován a řízen průběh testování.

Po dokončení provádění testů jsou úlohy testera obdobné jako úlohy test analytika. Pomáhá test manažerovi se shromažďováním metrik z nástrojů pro řízení testování a s posouzením celkového pokrytí a průběhu testů, může být pověřen předáním pracovních produktů testování relevantním osobám a týmům, archivací pracovních produktů a může být přizván jako důležitý zdroj informací na retrospektivní mítinky.

Úlohy, které tester vykonává a pracovní produkty, za které zodpovídá, znázorňuje Obrázek č. 17.




**Obrázek č. 17 Úlohy a pracovní produkty role tester, zdroj: [autor]**

Dle [RUP, 2010; Patton, 2002, s. 18] by měl mít tester následující schopnosti, vlastnosti a dovednosti:

- předchozí zkušenost s testováním, technikami testování a nástroji pro podporu testování,
- schopnost rozpoznat problémy a řešit je,
- být taktní, diplomatický a přesvědčivý při hlášení defektů,
- být pečlivý, vytrvalý, zvídavý a věnovat pozornost i detailům,
- znalost testované aplikace, systému nebo oblasti, která je předmětem testování (výhodou),
- znalosti webových aplikací a systémových architektur (výhodou).

U testera se zaměřením na automatizované testy je dále vyžadováno [RUP, 2010]:

- zkušenosti s nástroji pro automatizaci testů,
- programátorské dovednosti a zkušenosti s debuggováním[41].

Požadavky na dovednosti role testera se, stejně jako u test analytika, mohou lišit v závislosti na typu testů, které jsou prováděny.

V malých testovacích týmech je běžnou praxí přiřazení role testera společně s rolí test analytik. V testovacím týmu je potom v obou rolích zároveň více osob se stejnými znalostmi a zodpovědnostmi. Ve větších testovacích týmech mohou být tyto dvě role odděleny. Do role testera se obsazují obvykle méně zkušení členové týmu a mají na starost pouze povinnosti testera. Postupem času získávají četné zkušenosti v oboru testování a mohou přibírat k roli testera i úlohy test analytika. Do separátní role testera bývají také obsazováni testeři specializovaní na automatizaci testů. [RUP, 2010]

## 6.5 Pracovní produkty

Pracovní produkty jsou výstupy z procesu testování, které jsou v průběhu testování vytvářeny, modifikovány a používány [IBM, 2005]. V této metodice mají pracovní produkty v převážné většině podobu dokumentů spravovaných nástrojem pro řízení testování IBM Rational Quality Manager. Pracovní produkty mohou být v IBM Rational Quality Manager jednoduše mezi sebou provázány, což umožňuje komplexní pohled na průběžný stav testování. Náležitosti pracovních produktů byly nadefinovány na základě

[41] **Debuggování** je dle [ISTQB, 2012c, s. 17] definován jako *„proces nalézání, analyzování a odstraňování příčin selhání v softwaru“*.

standardu IEEE 829, který je charakterizován v kapitole 4.2.1. Pokud některé pracovní produkty nejsou navrženy v souladu s tímto standardem, je uveden důvod a navržena alternativní varianta. U pracovních produktů, které standard IEEE 829 nepokrývá, není uveden přesný seznam náležitostí, které by měl pracovní produkt obsahovat, ale jsou uvedena alespoň doporučení, jak by se k vytváření pracovního produktu mělo přistupovat.

Jednotlivé pracovní produkty jsou rozčleněny podle hlavních aktivit, které popisuje kapitola 6.2.

Podkapitoly, které pojednávají o pracovních produktech, mají jednotnou strukturu. V úvodu je pracovní produkt začleněn do kontextu hlavních aktivit a úloh procesu testování, následuje jeho definice a určení, kdo má za něj zodpovědnost. Potom je uveden seznam položek, které by měl pracovní produkt obsahovat, nebo alespoň doporučení pro vytváření pracovního produktu. Pro některé z pracovních produktů byly navíc nadefinovány i šablony v nástroji IBM Rational Quality Manager, popř. v MS Word. Šablony jsou k nahlédnutí v příloze B a při popisu pracovních produktů je na ně odkázáno. Dalším bodem popisu pracovních produktů je vazba na nástroj IBM Rational Quality Manager, kde je uvedeno, jak lze s daným pracovním produktem v nástroji pracovat.

Poslední částí popisu pracovního produktu je tabulka s informacemi o pracovním produktu, která je zabudována přímo v nástroji EPF Composer jako součást specifikace pracovního produktu. Jsou v ní uvedeny možné následky neexistence pracovního produktu, podmínky, kdy není pracovní produkt potřebný a doporučovaná forma pracovního produktu.

### 6.5.1 Hlavní aktivita plánování testů

#### *6. 5. 1. 1. Pracovní produkt plán testování*

Plán testování je komplexním výstupem z hlavní aktivity plánování testů. Tento dokument nastavuje proces testování na daném projektu a jsou podle něj řízeny veškeré aktivity testování. Vzniká již v době plánování projektu ve vazbě na projektový plán a postupně spolu s přibývajícími informacemi a dokumentacemi o projektu je zpřesňován a průběžně doplňován. Proces plánování testů ovšem nekončí ve chvíli, kdy je plán testování vytvořen. Plánování by mělo prostupovat celým procesem testování až po vyhodnocení výstupních kritérií a reportování. Plán testování by měl tedy být průběžně aktualizován v závislosti na změnách okolností a průběhu testovacích aktivit.

Dle slovníku standardu *IEEE Standard Glossary of Software Engineering Terminology (dále jen IEEE 610)* je plán testování definován jako dokument popisující rozsah, přístup, zdroje a harmonogram testovaných aktivit. Identifikuje položky k testování, testované vlastnosti, úlohy a zodpovědnosti za ně [s. 75] a podle [ISTQB, 2012c, s. 45] dále stupeň nezávislosti testerů, testovací prostředí, techniky návrhu testů, vstupní a výstupní kritéria a rizika vyžadující kontingenční plánování.

Na velkých projektech bývá zvlášť rozlišován hlavní testovací plán, který se zaobírá více úrovněmi testování a samostatné testovací plány pro jednotlivé úrovně testování. Tato metodika je určena především pro úroveň systémových testů, takže plán testování je možné chápat jako plán pro úroveň systémových testů.

Za vypracování a průběžné aktualizace plánu testování je zpravidla zodpovědný test manažer. Pomocnou rukou mu při specifikaci dílčích částí plánu mohou být i ostatní členové testovacího týmu, zejména v roli test analytik a v menší míře přispět mohou i testeři.

I když přesný obsah a struktura plánu testování závisí na organizaci, jejích standardech pro dokumentaci a úrovni formálnosti projektu, typicky plán testování obsahuje informace, které zachycuje Tabulka č. 10. Položky plánu testování v této tabulce vychází ze standardu IEEE 829 a jedná se o výběr a shrnutí těch nejdůležitějších z nich.

**Tabulka č. 10 Položky testovacího plánu, zdroj: [IEEE, 2008, s. 42-50; Systeme Evolutif Limited]**

| Položka plánu testování | Popis |
|---|---|
| **Manažerské shrnutí** | Krátké a výstižné shrnutí účelu plánu pro potřeby managementu a přehled záměrů a cílů testování. Zahrnuje identifikaci rozsahu plánu ve vztahu k projektovému plánu, rozpočtová omezení a omezení zdrojů, rozsah testování, vymezení vazby testování na ostatní aktivity a případně proces změnového řízení, pravidla pro komunikaci a koordinaci klíčových aktivit. |
| **Reference** | Seznam všech dokumentů, které plán testování podporují, nebo se k němu vztahují. Může se jednat o projektový plán, specifikace požadavků, designové specifikace, standardy pro proces vývoje software a testování metodologické příručky a příklady, podnikové standardy a směrnice, apod. |
| **Testované položky** | Seznam položek, které jsou předmětem testování. Může se jednat například o specifické vlastnosti systému, instalační příručky, uživatelské příručky, hardwarová rozhraní, apod. K jednotlivým položkám se obvykle váže i číslo verze a u kritických položek také termín předávky z ostatních prostředí do testovacího prostředí. |
| **Testované vlastnosti** | Seznam všech vlastností systému, které jsou předmětem testování. Jedná se o ty vlastnosti, které jsou viditelné z pohledu uživatele, nikoliv tedy o technický popis jako u testovaných položek. |

| | |
|---|---|
| **Netestované vlastnosti** | Seznam všech vlastností systému, které nejsou předmětem testování včetně zdůvodnění, proč nebudou testovány. |
| **Přístup k testování** | Celkový přístup k testování. Popisuje, kterou úroveň testování pokrývá, které charakteristiky kvality dle metody FURPS budou pokryty, techniky pro návrh testů (techniky černé a bílé skříňky) a nástroje pro automatizaci testů nebo případně jiné nástroje pro podporu přípravy a provádění testování. |
| **Vstupní a výstupní kritéria** | Vstupní kritéria definují, co musí být splněno před zahájením testování. Výstupní kritéria definují podmínky, za kterých může být testování zakončeno. Jsou obvykle založena na počtu nalezených defektů určitého stupně závažnosti. |
| **Kritéria pro přerušení a požadavky na následnou obnovu** | Definice podmínek, při kterých je nutné přerušit testování. Specifikují akceptovatelnou úroveň defektů, které ještě umožní pokračovat v testování po předchozích defektech. Dále specifikuje aktivity, které je nutné opakovat při obnově testování. |
| **Součásti dodávky za testování** | Seznam součástí dodávky ze strany testovacího týmu. Součástí dodávky mohou být kromě testovacího plánu testovací případy, výstupy z nástrojů pro podporu testování, simulátory, statické a dynamické generátory, test logy, reporty defektů, reporty o stavu testování a další. |
| **Součinnosti** | Identifikace vztahů a dodávaných pracovních produktů mezi testovacím týmem a ostatními osobami či odděleními. |
| **Plánované aktivity a úlohy** | Identifikace úkolů, které budou prováděny v rámci přípravy a provádění testů, včetně jejich vzájemných závislostí. Zároveň je zapotřebí určit všechny důležité faktory jako je například dostupnost testované položky, dostupnost zdrojů a termíny. |
| **Zodpovědnosti** | Identifikace jednotlivců a skupin, kteří jsou zodpovědní za řízení, návrh, přípravu, vykonávání, potvrzování a kontrolu výsledků testování a za řešení nalezených defektů. Dále jsou specifikovány veškeré další zodpovědnosti za jednotlivé položky definované v plánu testování (výběr vlastností, které budou nebo nebudou testovány, identifikace rizik, atd.). |

| | |
|---|---|
| **Požadavky na testovací prostředí a infrastrukturu pro testování** | Specifikace nezbytných a požadovaných vlastností testovacího prostředí a testovacích dat, což může zahrnovat požadavky na speciální hardware i software, podpůrné nástroje, databáze, platformy, lidské zdroje, nastavení prostředí před zahájením testování a další potřeby. |
| **Školení a požadované dovednosti** | Specifikace potřebných školení a nezbytných dovedností členů testovacího týmu, přičemž varianty školení mohou být různé od tradičních kurzů v podobě přednášek a seminářů, přes samostudium prostřednictvím e-learningu a internetu, až po mentoring od zkušenějších členů týmu. |
| **Harmonogram, odhady a náklady** | Veškeré milníky pro testování identifikované v projektovém plánu a vlastní milníky testovacího týmu, časové odhady jednotlivých úloh testování, harmonogram pro každou úlohu, rozpočet na testovací zdroje jako je vybavení, nástroje a personál. |
| **Rizika testování** | Identifikace produktových a projektových rizik testování, které mají dopad na úspěšně zakončení aktivit testování, včetně specifikace dopadu každého rizika a návrhu preventivních a nápravných opatření pro jejich eliminaci. |
| **Metriky** | Metriky hodnocení cílů, které budou v rámci projektu sledovány, analyzovány a reportovány, včetně stanovení jejich cílových hodnot s ohledem na výstupní kritéria. |
| **Terminologický slovník** | Seznam termínů a zkratek používaných v plánu testování a obecně v oboru testování spolu s vysvětlením jejich významů. |

Ukázková šablona plánu testování v nástroji IBM Rational Quality Manager, která je nadefinována podle standardu IEEE 829 [IEEE, 2008, s. 42-50; Systeme Evolutif Limited], je zařazena v příloze C. V nástroji IBM Rational Quality Manager je možnost kromě již předdefinovaných položek přidávat do šablony i své vlastní položky a vytvořit si tak šablonu dle potřeby. Nevýhodou však je, že tyto vlastní položky mají omezenou možnost formátování a naopak předdefinované položky jsou needitovatelné. Přestože záměrem bylo převést šablonu kompletně do češtiny, některé položky musely být v důsledku těchto nevýhod ponechány v angličtině ve formě předdefinované nástrojem IBM Ratinoal Quality Manager.

Další informace o tomto pracovním produktu uvádí Tabulka č. 11.

**Tabulka č. 11 Plán testování, zdroj: [autor]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Pokud neexistuje plán testování, řízení procesu testování probíhá pouze intuitivně bez vazby na konkrétní cíle a metriky. Příprava a provádění testů nejsou vázány na rizika produktu, nejsou schváleny vztahy a termíny dodávek pracovních produktů ze strany ostatních osob a týmů mimo tým testování, což může mít za důsledek nedodržení očekávaných povinností od těchto zainteresovaných stran a nevyhovění požadavkům testovacích týmu. |
| **Podmínky, kdy není pracovní produkt potřebný** | Plán testování je zapotřebí vždy. V závislosti na potřebách projektu je možné vynechat některé části, které test manažer v konkrétním případě uzná za nedůležité. |
| **Forma pracovního produktu** | Plán testů je vhodné udržovat přímo v nástroji IBM Quality Manager, kde ho lze v průběhu procesu testování snadno propojit s dalšími pracovními produkty, zejm. s testovacími případy (popř. testovacími sadami). |

## 6.5.2 Hlavní aktivita analýza testů

### *6. 5. 2. 1. Pracovní produkt seznam testovacích podmínek*

Seznam testovacích podmínek je výstupem z hlavní aktivity analýza testů, která předchází aktivitě návrh testů, nebo je s ní sloučena. Tento dokument v bodech definuje, co se bude testovat ve vazbě na podklady pro testování, produktová rizika, strategické cíle a cíle testování. Následně slouží jako podklad pro další pracovní produkty v aktivitách návrh a implementace testů, zejména pro testovací případy. Seznam testovacích podmínek by měl být zpracován, jakmile je k dispozici plán testování a podklady pro testování.

Dle slovníku ISTQB [ISTQB, 2012c, s. 43] je testovací podmínka definována jako „*položka či událost systému nebo komponenty, která má být ověřena jedním či více testovacími případy“*. Jinak řečeno je to tedy „něco“, co je možné otestovat, ale už neřeší způsob, jakým to bude otestováno. V některých jiných metodikách (RUP, OPENUP i MMSP) a v literatuře se namísto pojmu testovací podmínka používá pojem testovací nápad (test idea). Tyto dva pojmy označují totéž.

Za vypracování seznamu testovacích podmínek je zpravidla zodpovědný test analytik. V případě, že není dostatek podkladů pro testování, nebo jsou zastaralé, je nutné vydefinovat testovací podmínky ve spolupráci se zainteresovanými stranami na projektu.

Seznam pracovních podmínek není specifikován standardem IEEE 829. Ačkoliv forma, v jaké jsou testovací podmínky zpracovány, obvykle závisí na položce, která je předmětem testování, je užitečné se držet některých obecných doporučení [ISTQB, 2012b, s. 12]:

- Testovací podmínky je vhodné uspořádat do hierarchické struktury, kdy vyšší úroveň mapuje danou podmínku obecně (např. funkcionalita obrazovky X) a nižší úroveň potom tuto obecnou podmínku rozvádí do většího detailu (např. obrazovka X odmítne rodné číslo, které není ve specifickém formátu).
- Pokud jsou definována produktová rizika, tak testovací podmínky by měly být zaměřeny na jejich pokrytí.
- V rámci zpracování seznamu testovacích podmínek je kromě samotné specifikace podmínek vhodné zachytit i vazbu na podklady testování, ze kterých podmínka vychází, cíle testování, strategické cíle a produktová rizika. V rámci návrhu testů, které vychází z testovacích podmínek, je vhodné opět zachytit vazbu testovacích případů na dané testovací podmínky.

Ukázková šablona seznamu testovacích podmínek je zařazena v příloze C. Vzhledem k tomu, že tento pracovní produkt nemá svou interpretaci v nástroji IBM Rational Quality Manager, šablona seznamu testovacích podmínek byla zpracována v MS Word a vychází ze šablony testovacích nápadů z metodiky MMSP, přičemž přidává propojitelnost testovacích podmínek na podklady pro testování, cíle, produktová rizika, testovací případy a mění terminologii podle ISTQB. V nástroji IBM Rational Quality Manager může být tento produkt připojen jako příloha k testovacímu plánu, tzn. do části *„Attachments“*.

Další informace o tomto pracovním produktu uvádí Tabulka č. 12.

**Tabulka č. 12 Seznam testovacích podmínek, zdroj: [autor]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Bez seznamu testovacích podmínek je při přípravě testovacích případů nutné vycházet přímo z podkladů pro testování. Vzhledem k tomu, že seznam testovacích podmínek obvykle vzniká ještě před tím, než je zpracovaný detailní design pro vývoj softwaru, je možné při revizi podkladů pro testování zachytit případné defekty a vyvarovat se tak jejich odstraňování v pozdějších fázích návrhu. Jiným následkem neexistence je absence přímého propojení cílů a produktových rizik na jednoduché testovací podmínky, kterým na rozdíl od testovacích případů rozumí všechny zainteresované strany a mohou včas zareagovat, pokud zjistí nedostatečné pokrytí rizik a cílů. |

| | |
|---|---|
| **Podmínky, kdy není pracovní produkt potřebný** | Seznam testovacích podmínek není nutné vytvářet, pokud lze podklady pro testování jednoduše vztáhnout přímo na testovací případy nebo je možné využít již existující seznam testovacích podmínek z podobného projektu, který stačí poupravit. |
| **Forma pracovního produktu** | Seznam testovacích podmínek může být ve formě textového dokumentu, který se připojí k plánu testování v IBM Rational Quality Manager. Vhodná forma zápisu testovacích podmínek je „akce a reakce“ (např. Provedeme X → Zobrazí se X). Jednotlivé testovací podmínky mohou být hierarchizovány a propojeny s podklady pro testování, cíly, produktovými riziky a testovacími případy. |

### 6.5.3 Hlavní aktivita návrh testů

#### *6. 5. 3. 1. Pracovní produkt testovací případ*

Testovací případy jsou výstupem z hlavní aktivity návrh testů, která buď následuje po analýze testů jako separátní aktivita, nebo je s ní sloučena. Testovací případy konkretizují abstraktní testovací podmínky specifikací vstupů pro testování a očekávaných výsledků testů. V rámci implementace testů jsou potom do testovacích případů začleněny testovací skripty, což je konkrétní posloupnost kroků pro vykonání testu. V praxi jsou pod pojmem testovací případ chápány nejen specifikace vstupů a výsledků testů, ale jsou pod tímto pojmem často zahrnuty i testovací skripty, tedy jednotlivé akce testera a očekávané výsledky těchto akcí ze strany systému. V tomto pojetí se tedy tvorba testovacích případů, která spadá do aktivity návrh testů, slučuje s tvorbou testovacích skriptů, která patří již do aktivity implementace testů.

Dle slovníku ISTQB [ISTQB, 2012c, s. 43] je testovací případ definován jako *„sada vstupů pro testování, podmínek před provedením, očekávaných výsledků a podmínek po provedení, vytvořená za určitým cílem jako například otestovat určitou větev kódu či ověřit stupeň splnění určitého požadavku“*.

Tvorba testovacích případů by měla být zahájena, jakmile jsou identifikovány testovací podmínky a je k dispozici dostatek informací a podkladů pro vytvoření testovacích případů. Testovací případy jsou, stejně jako testovací podmínky, vztaženy na podklady pro testování, produktová rizika, strategické cíle a cíle testování. V nástroji IBM Rational Quality Manager je propojení zajištěno na úrovni vytváření testovacího případu skrze sekci *„Requirement links“*, kde se testovací případ propojí s požadavky, které pokrývá. S plánem testování jsou testovací případy propojeny v sekci *„Test cases“* na úrovni vytváření

testovacího plánu. Vazba testovacích případů na testovací podmínky je potom zachycena v dokumentu Seznam testovacích podmínek v sekci Testovací podmínky.

Za vytváření testovacích případů je zodpovědný Test analytik. Při tvorbě testovacích případů postupuje v rámci rozsahu a v souladu s přístupem, stanovenými v plánu testování.

Dle standardu IEEE 829 testovací případ zahrnuje položky, které zachycuje Tabulka č. 13.

**Tabulka č. 13 Položky testovacího případu, zdroj: [IEEE, 2008, s. 52-55; SQE, 2001a]**

| Položka testovacího případu | Popis |
|---|---|
| **Cíl** | Identifikace a stručný popis záměru nebo cíle testovacího případu. Specifikuje, co testovací případ testuje, jakou testovací podmínku a jakou položku testování. Dále může zahrnovat i riziko a prioritu testovacího případu. |
| **Vstupy** | Specifikace vstupů, které jsou zapotřebí pro spuštění testovacího případu. Může se jednat o konkrétní hodnoty proměnných, nebo o tabulky, databáze a soubory. |
| **Výsledky** | Specifikace výsledků a očekávaného chování testovaných položek (např. doba odezvy). |
| **Potřeby prostředí** | Popis testovacího prostředí, které je zapotřebí pro spuštění a provádění testovacího případu a pro záznam výsledků testovacího případu. |
| **Speciální požadavky na skripty** | Popisují speciální omezení testovacích skriptů, jako jsou vstupní a výstupní podmínky při provádění skriptů. |
| **Vazba na ostatní testovací případy** | Seznam testovacích případů, které musí být spuštěny před tímto testovacím případem. |

Ukázková šablona testovacího případu v nástroji IBM Ratinal Quality Manager je zařazena v příloze C. Všechny výše uvedené položky případu užití jsou v šabloně zahrnuty, ovšem jejich struktura byla pozměněna. Vstupy jsou rozlišeny na vstupní data a vstupní podmínky, přičemž ve vstupních podmínkách mohou být zároveň zahrnuty speciální požadavky na skripty a vazby na ostatní testovací případy. Analogicky ke vstupním podmínkám jsou nadefinovány samostatně i výstupní podmínky, které specifikují kroky, které je zapotřebí provést po provedení testovacího případu, opět včetně případných vazeb na ostatní testovací případy. Zvláštní sekce je vyhrazena také na rizika, položky k testování a vazbu na požadavky, které testovací případ pokrývá. Stejně jako v případě plánu testování, je

většina položek v češtině, ale některé položky bylo nutné ponechat v angličtině v původní formě předdefinované nástrojem IBM Rational Quality Manager.

Další informace o tomto pracovním produktu zachybuje Tabulka č. 14.

**Tabulka č. 14 Testovací případ, zdroj: [autor]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Bez testovacích případů chybí při provádění testů kontrola nad tím, co už bylo otestováno a co ještě zbývá otestovat a informace o tom, kolik testovacích případů prošlo nebo selhalo. Chybí tak důležitý podklad pro rozhodnutí o tom, zda daný software splňuje očekávání zainteresovaných stran. Při testech hrozí, že se zapomene řádně otestovat nějaká důležitá oblast nebo pokrýt testy významné riziko. Zároveň může dojít k chybné interpretaci pozorovaných výsledků testů, protože nejsou definovány očekávané výsledky a je nutné vycházet buď přímo z podkladů pro testování, nebo postupovat intuitivně. Obtížnějším se stává i reportování defektů, protože je nutné přesně popsat, jakým způsobem k selhání došlo, namísto aby defekt byl navázán přímo na testovací případ, při kterém k selhání došlo. Absence testovacích případů také znemožňuje opakování stejných testů s časovým odstupem. |
| **Podmínky, kdy není pracovní produkt potřebný** | Testovací případy by měly být vytvářeny vždy. Ve výjimečných případech ovšem mohou být definovány pouze testovací podmínky (např. daná položka pro testování nevyžaduje podrobný popis testovacích případů, požadavky na systém se neustále mění, poklady pro testování nejsou dostatečné pro definici očekávaných výsledků, nebo projektový manažer zkrátil čas původně vymezený na testování). |
| **Forma pracovního produktu** | Testovací případy je vhodné udržovat přímo v nástroji IBM Quality Management. Tam je lze přímo propojit s plánem testování a požadavky, s testovacími skripty a následně je zařadit do testovacích sad. Pro testovací případy je klíčová jejich výstižnost a srozumitelnost. |

## 6.5.4 Hlavní aktivita implementace testů

### *6. 5. 4. 1. Pracovní produkt testovací skript*

Testovací skripty jsou jedním z výstupů z hlavní aktivity implementace testů, která následuje po analýze a návrhu testů a předchází samotnému provádění testů. Testovací skripty specifikují jednotlivé kroky, které je zapotřebí vykonat při provádění testovacích případů. V praxi bývají testovací skripty často přímou součástí testovacích případů a vytváření testovacích skriptů je tak spojeno s vytvářením testovacích případů a spadá do aktivity návrh testů. V terminologii ISTQB a standardů IEEE se ovšem jedná o dva samostatné pracovní produkty a testovací skripty se do testovacích případů začleňují až v rámci implementace testů. Nástroj IBM Rational Quality Manager je v tomto ohledu v souladu s ISTQB a IEEE a tyto dva pracovní produkty také rozlišuje. Z tohoto důvodu byl v této metodice zvolen přístup, kdy testovací skripty jsou samostatným produktem testování.

V terminologii standardu IEEE 610 je pojem testovací skript synonymem s pojmem testovací procedura, přičemž testovací procedura je zde definována jako *„detailní instrukce pro nastavení, provedení a vyhodnocení výsledků daného testovacího případu“*, nebo také jako *„dokumentace specifikující posloupnost akcí pro vykonání testu“* [1990, s. 76]. Standard ISTQB se od IEEE 610 mírně odlišuje a přiklání se k často používanému rozlišení testovacího skriptu na manuální a automatizovaný. Testovací proceduru [ISTQB, 2012c, s. 46] definuje obecně jako *„posloupnost akcí pro vykonání testu“* s tím, že *„je známá také jako testovací skript nebo jako manuální testovací skript“*. Testovací skript potom [ISTQB, 2012c, s. 47] definuje jako *„běžně používaný jako specifikace testovací procedury, ale je zejména automatizovaný“*. Vzhledem k tomu, že nástroj IBM Rational Quality Manager tento produkt nazývá jako testovací skript bez ohledu na to, zda je manuální nebo automatický, z důvodu zachování jednotné terminologie v nástroji i v této metodice není dále používán pojem testovací procedura ale pouze pojem testovací skript, který může být buď manuální, nebo automatický.

Za vytváření testovacích skriptů, stejně jako za vytváření testovacích případů, je zodpovědný Test analytik.

Dle standardu IEEE 829 testovací skript zahrnuje položky, které zachycuje Tabulka č. 15.

**Tabulka č. 15 Položky testovacího skriptu, zdroj: [IEEE, 2008, s. 55-57; SQE, 2001b]**

| Položky testovacího skriptu | Popis |
|---|---|
| **Vstupy, výstupy a speciální požadavky** | Identifikace všeho, co je zapotřebí pro provedení testů jako např. nástroje pro automatizaci testů, požadavky na testovací prostředí, speciální požadavky na dovednosti nebo školení, jiné skripty, které musí být provedeny před daným skriptem, apod. |

| | |
|---|---|
| **Uspořádaný popis kroků** | Specifikace jednotlivých na sebe navazujících aktivit a jejich očekávaných výsledků, což může zahrnovat např. přípravné kroky pro provedení skriptu (setup), úvodní kroky nezbytné pro zahájení skriptu (start), kroky vykonávané během provádění skriptu (proceed), kroky nezbytné pro dokončení provedeného skriptu (wrap-up), atd. |

Výchozí šablona testovacího skriptu v nástroji IBM Rational Quality Manager je zařazena v příloze C. V nástroji není možnost úpravy šablony testovacího skriptu dle specifických potřeb, tudíž je nutné zachovat výchozí strukturu. Velkou výhodou testovacího skriptu v IBM Rational Quality Manager je, že kromě vytváření klasických manuálních skriptů, poskytuje možnost přímého připojení automatizovaných skriptů. Automatizované skripty se pak následně mohou spouštět přímo z IBM Rational Quality Manager.

Další informace o tomto pracovním produktu zachycuje Tabulka č. 16.

**Tabulka č. 16 Testovací skript, zdroj: [autor]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Bez testovacích skriptů je velmi obtížné zajistit konzistentní provádění testů při opakování stejných testů vícekrát, nebo při provádění testů různými osobami. Může se snadno stát, že tester narazí na selhání, ale není schopen jej znovu nasimulovat, protože během provádění testu zapomene, jaké kroky mu předcházely. Při vytváření defekt reportu je potom nutné znovu hledat cestu, která selhání zapříčinila, aby defekt report nebyl vývojovým týmem zamítnut jako nereprodukovatelný. |
| **Podmínky, kdy není pracovní produkt potřebný** | Testovací skripty není nutné vytvářet, pokud jsou kroky testu jednoduché a zřejmé již z testovacích případů a testovacích podmínek. Při použití automatizovaných nástrojů pro provádění testů je vhodné nastavit automatické zachycení záznamu testování, aby při případném vzniku selhání bylo možné nejen identifikovat místo vzniku selhání, ale také kroky, které selhání předcházely. |
| **Forma pracovního produktu** | Testovací skripty je vhodné udržovat přímo v nástroji IBM Rational Quality Manager, kde je lze přímo propojit s testovacími případy. Pro testovací skripty je stejně jako pro testovací případy klíčová jejich výstižnost a srozumitelnost. |

#### *6. 5. 4. 2. Pracovní produkt testovací data*

Během implementace testů při vytváření testovacích skriptů bývá zvykem zvlášť vytvořit data, která jsou potřebná pro testování. Testovací skripty potom obsahují odkazy na testovací data, která jsou zadávána do systému při provádění testů anebo jsou očekávána na výstupu testu.

Dle [ISTQB, 2012c, s. 43] jsou testovací data definována jako „*data, která existují (například v databázi) ještě před provedením testu a která ovlivňují nebo jsou ovlivňována testovanou komponentou nebo systémem"*.

Za vytváření testovacích dat je zodpovědný primárně test analytik, ale část úkolů může delegovat i na testery, kteří mají s vytvářením testovacích dat zkušenosti. Při testování specifických oblastí je často nutná i spolupráce testovacího týmu se zadavateli ze strany byznysu, kteří se orientují v problematice, kterou systém podporuje, a znají specifickou vnitřní logiku dat.

Testovací data nemají předepsanou strukturu, protože se pokaždé může jednat o data jiného typu. [Šplíchalová, 2008, s. 43] Data mohou mít podobu sešitu v Excelu [Software Testing Mentor], který může testerům sloužit pro manuální zadávání dat do systému, nebo může být napojen na automatizované nástroje. Dále mohou být data organizována v databázích, v textových souborech, v souborech XML nebo v podobě bitmapových obrázků a šifrovaného textu [Šplíchalová, 2008, s 43].

V některých případech mohou být sady dat velmi rozsáhlé a jejich vytváření a aplikování při provádění testů není v lidských silách. Pro tyto případy je vhodné použít automatické generátory dat, které jsou schopné v krátkém čase vytvořit velké množství dat podle zadaných kritérií a následně je načíst do požadovaného datového zdroje, což může být např. tabulka, soubor nebo přímo databáze. Zástupem generátorů dat jsou například nástroje DTM Data Generator, Tnsgen, forSQL Data Generator, Test Data Generator TDG nebo DB Data Generator [Fiurášek, 2010, s. 66-67]. Při provádění testů je potom možné vytvořenou datovou základnu připojit na automatizovaný nástroj, který si v požadovaném sledu bere jednotlivá data.

V nástroji IBM Rational Quality Manager je možné nahrát testovací data ze sešitu v Excelu uložená ve formátu csv. Při vytváření testovacích skriptů lze potom provázat skripty s testovacími daty a do popisu jednotlivých kroků zařadit odkazy na jednotlivá data. Další informace o tomto pracovním produktu zachycuje Tabulka č. 17.

**Tabulka č. 17 Testovací data, zdroj: [autor]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Validní testovací data jsou jednou z podmínek pro úspěšné vykonání testu. Pokud nejsou připravena, některé testovací skripty není možné úspěšně dokončit, protože nevalidní testovací data mohou zapříčinit rozdílné chování systému oproti očekávanému. Logika chování systému často závisí právě na zadávaných datech. Pokud tester nezná dopady na chování systému při zadání určitých dat, nemůže korektně vyhodnotit, zda test prošel nebo selhal. Testeři potom reportují defekty, které jsou vývojovým týmem vráceny z důvodu nevalidních dat. |
| **Podmínky, kdy není pracovní produkt potřebný** | Testovací data musí existovat vždy. Ne vždy však musí být vytvářena samostatně. Pokud jsou menšího rozsahu, lze je zapsat přímo do popisu kroku skriptu (např. „Vyplň pole rok narození ve formátu RRRR“.) |
| **Forma pracovního produktu** | Forma testovacích dat může být různá. Mohou být ve formě sešitu v Excelu, nebo mohou být organizována v databázích, v textových souborech, v souborech XML, v podobě bitmapových obrázků, šifrovaného textu, atd. Pokud jsou vytvářena manuálně v Excelu, je vhodné je převést do formátu csv, nahrát přímo do nástroje IBM Rational Quality Manager a následně je propojit s testovacími skripty. |

#### *6. 5. 4. 3. Pracovní produkt testovací sada*

V rámci implementace testů mohou být testovací případy organizovány do tzv. testovacích sad, které umožňují kompletně otestovat určitou část vyvíjené komponenty nebo systému. Vytváření testovací sady spočívá ve výběru a uspořádání jednotlivých testovacích případů do logické posloupnosti. [Rejnková, 2012, s. 80]

Dle [ISTQB, 2012c, s. 47] je testovací sada definována jako *„sada několika testovacích případů pro komponentu nebo systém pod testem, kde výstupní podmínky jednoho testu jsou často použity jako vstupní podmínky testu následujícího“*.

Uspořádání testovacích případů do testovací sady lze použít například v případě, kdy jednotlivé testovací případy představují části procesu a testovací sada sestavená z těchto testovacích případů představuje komplexní proces, kterým uživatel prochází při založení nového účtu do systému. [IBM, 2012] Taková testovací sada by potom obsahovala například sled následujících testovacích případů:

- Testovací případ č. 1 – Registrace uživatele,
- Testovací případ č. 2 – Aktivace uživatele,
- Testovací případ č. 3 – Inicializace uživatele,
- Testovací případ č. 4 – Přihlášení uživatele,
- Testovací případ č. 5 – Odhlášení uživatele.

V tomto případě je tedy nutné vykonat testovací případy právě v tomto pořadí určeným testovací sadou, protože např. samotné odhlášení uživatele by nebylo možné provést bez předchozího přihlášení uživatele.

Tento způsob použití testovací sady ovšem není jediným. Testovací sady lze použít obecně pro organizaci testovacích případů za nějakým účelem. Výstupní podmínky jednoho testu nemusí být nutně vstupními podmínkami testu následujícího. Testovací sady v tomto pojetí mohou být použity např. pro kolekci testovacích případů za účelem ověření sestavení nebo ověření stability produktu, nebo pro sdružení testovacích případů zaměřených na specifickou funkcionalitu produktu, nebo k vytvoření regresního průchodu přes funkční oblasti produktu [IBM, 2012].

Organizace testovacích případů do testovacích sad je v zodpovědnosti test analytika, který nejlépe ví, které testovací případy je zapotřebí uspořádat či sloučit a které naopak mohou být prováděny samostatně, bez vazby na ostatní testovací případy.

Ukázková šablona testovací sady v nástroji IBM Rational Quality Manager je zařazena v příloze C. Standard IEEE 829 nepředepisuje žádnou strukturu testovací sady, proto jsem vycházela z výchozí struktury testovací sady v IBM Rational Quality Manager, kterou jsem upravila obdobně jako šablonu testovacího případu.

Další informace o tomto pracovním produktu zachycuje Tabulka č. 18.

**Tabulka č. 18 Testovací sada, zdroj: [autor; Rejnkonvá, 2011, s. 80]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Pokud nejsou vytvářeny testovací sady, může se stát, že identifikovaná selhání nejsou způsobena špatnou funkcionalitou komponenty nebo systému, ale špatně provedenými testy, které probíhají neorganizovaně a chaoticky. Jiné defekty zase nemusí být odhaleny. [Rejnková, 2011, s. 80] |
| **Podmínky, kdy není pracovní produkt potřebný** | Testovací sady nemusí být vytvářeny, pokud jsou testovací případy na sobě nezávislé a lze je provádět samostatně bez vazby na ostatní testovací případy. |

| | |
|---|---|
| **Forma pracovního produktu** | Testovací sady je vhodné udržovat přímo v nástroji IBM Rational Quality Manager. Tam je lze přímo propojit s plánem testování a uspořádat v nich předem vytvořené testovací případy. Forma testovací sady by měla být obdobná jako forma testovacích případů. |

### 6.5.5 Hlavní aktivita provádění testů

#### *6. 5. 5. 1. Pracovní produkt test log*

Při provádění testů jsou výsledky jednotlivých testů zaznamenávány do tzv. test logu. Účelem logování výsledků testů je získat přehled o tom, jaké testy byly provedeny, jak probíhaly jednotlivé kroky testů a jak testy dopadly celkově. Zároveň test log slouží i jako důkaz o tom, že dané testy provedeny byly. [Šplíchalová, 2008, s. 44] Díky informacím o průběhu testů zaznamenaným v test logu je možné reportovat aktuální stav testování a kvality produktu, řídit průběh testování, měřit dosažené pokrytí testy a srovnávat aktuální stav proti výstupním kritériím.

Dle [ISTQB, 2012c, s. 45] je test log definován jako „*chronologický záznam všech relevantních detailů souvisejících s prováděním testů*“.

Za zaznamenávání výsledků testů a detailů o testech do test logu je zodpovědný tester. Při provádění manuálních testovacích skriptů musí tester psát záznamy o jednotlivých krocích ručně. Pokud jsou spouštěny automatizované skripty, všechny potřebné informace může zachytit automatizovaný nástroj a tester jen prověří případná selhání. [Čermák, 2009]

Dle standardu IEEE 829 test log zahrnuje informace, které zachycuje Tabulka č. 19.

**Tabulka č. 19 Položky test logu, zdroj: [IEEE, 2008, s. 58-69; SQE, 2001c]**

| Položka Test logu | Popis a dílčí položky |
|---|---|
| **Popis** | Obsahuje obecné informace vztahující se ke všem položkám daného test logu: |
| | • Identifikace testovaných položek včetně jejich verze, |
| | • identifikace případných rozdílů skutečných podmínek testu oproti plánovaným, |
| | • datum a čas zahájení a ukončení testu, |
| | • osoba, která daný test provedla. |

| | |
|---|---|
| **Aktivity a vstupy událostí** | Záznam aktivit nebo událostí pro každý relevantní detail: |
| | • Datum a čas začátku a konce každé důležité aktivity, resp. výskytu události, |
| | • identifikace prováděného testovacího skriptu, |
| | • identifikace všech osob, které participují na provádění testu včetně jejich rolí, |
| | • výsledky jednotlivých kroků, skriptů a celých testovacích případů (prošlo / neprošlo), |
| | • informace o změnách v testovacím prostředí, které se odchylují od plánů, |
| | • záznamy o anomáliích, tzn., co se stalo před a po neočekávané události a jaké byly její další okolnosti[42], |
| | • odkazy na identifikátory defekt reportů vztahujících se k běhu testu. |

V nástrojích pro řízení testování obvykle proces logování výsledků testů probíhá tak, že tester vybere danou testovací sadu nebo testovací případ, spustí jej, vybere verzi a zadá další potřebné informace. Následně provádí test krok za krokem, přičemž pozoruje a zapisuje skutečné chování testované komponenty či systému. Po dokončení posledního kroku testu se automaticky vygeneruje test log, kde jsou zaznamenány všechny potřebné informace.

Na tomto principu funguje i zaznamenávání výsledků v IBM Rational Quality Manager. Po spuštění testovacího případu se objeví formulář pro výběr plánu testování, ke kterému se daný testovací případ váže, volba iterace a testovacího prostředí, výběr konkrétního testovacího skriptu a verze, na které test probíhá. Následně se přejde do režimu záznamu výsledků jednotlivých kroků, kde je možné kromě textových informací připojit přílohy, screenshoty a také přímo založit nový defekt report. Tím je zajištěna provázanost mezi testovacími skripty a defekt reporty. Usnadňuje se tak založení defektu, protože přesné kroky, které vedly k výskytu defektu a okolnosti výskytu defektu jsou již jednou zaznamenány v test logu a do defekt reportu se automaticky předvyplní. Po ukončení posledního kroku testu se vygeneruje test log, kde jsou všechny zadané informace uloženy. K nim se automaticky doplní datum a čas zahájení a ukončení běhu testu, celkový čas běhu testu, vazba na testovací případ, testovací skript a testovací data, osoba, která prováděla test a celkový výsledek daného běhu testu.

[42] Pokud jsou tyto informace zaznamenány přímo v defekt reportu, tato položka se nevyplňuje.

Další informace o tomto pracovním produktu zachycuje Tabulka č. 20.

**Tabulka č. 20 Test log, zdroj: [autor; Rejnková, 2011, s. 81]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Bez zaznamenávání výsledků do test logu není možné sledovat průběžný stav testování. Testeři nemají přehled o tom, kdo kdy provedl jaké testy, a tak se může snadno stát, že některé z připravených testů provedou vícekrát a na některé zapomenou. Test manažer není schopen říci, kolik z připravených testů již bylo provedeno a kolik ještě zbývá provést, porovnat aktuální stav testování proti výstupním kritériím a kvalitu produktu může hodnotit pouze podle zalogovaných defektů. |
| **Podmínky, kdy není pracovní produkt potřebný** | Test log je nutné vytvářet vždy, když se postupuje podle testovacích sad, případů a skriptů. Nicméně, u automatizovaných testů může mít podobu automaticky generovaného test logu. [Rejnková, 2011, s. 81] Výjimkou je, pokud se testovací tým rozhodne zefektivnit provádění testů využitím některé z technik založené na zkušenostech. V případě testování založeného na zkušenostech testeři nepostupují podle připravených skriptů, tudíž nelogují výsledky testů krok za krokem, ale pouze reportují nalezené defekty. |
| **Forma pracovního produktu** | Test logy je vhodné vytvářet přímo v nástroji IBM Quality Manager. Tam je lze přímo propojit s plánem testování, testovacími skripty, testovacími případy, testovacími sadami a defekt reporty. |

#### *6. 5. 5. 2. Pracovní produkt defekt report*

Jádrem provádění testů je srovnávání aktuálního chování systému s očekávaným. Pokud se skutečné chování od očekávaného liší, je třeba neočekávaný výskyt (tzv. anomálii) blíže prozkoumat. Když se prokáže, že anomálie je selhání způsobené defektem, je třeba založit defekt report[43] a nalezený defekt vyřešit.

Dle [ISTQB, 2012c, s. 18] je defekt report definován jako *„dokument, který reportuje nějaký nedostatek v komponentě nebo v systému, který může způsobit, že daná komponenta či systém selže při vykonávání požadované funkce“*.

[43] Použití pojmu **report** je v oboru testování běžnější, než synonymní český výraz hlášení. Testeři při hlášení defektů často využívají nástroje pro podporu řízení testů a správu defektů, ve kterých je funkce založení defekt reportu (popř. incident reportu). Pojem defekt report je tedy v souladu se zvyklostmi testerů.

Účelem defekt reportu je

- upozornit vývojáře a ostatní osoby zainteresované na projektu o defektu a poskytnout jim dostatečné informace pro identifikaci, izolaci a opravu defektu,
- poskytnout test manažerovi prostředky pro hodnocení kvality testovaného systému a kontrolu pokroku testování,
- poskytnout výchozí bod pro další release [Ramdeo, 2011],
- poskytnout podněty pro zlepšení procesu testování.

Dle Pattona je cílem softwarového testera vyhledávat chyby (v této práci nazývané termínem defekty), vyhledávat je co nejdříve a zajistit jejich nápravu [2002, s. 17]. Defekty nemusí být nalézány a reportovány až při samotném provádění testů, ale již mnohem dříve. Test analytik může objevit a nahlásit defekty už během přípravy testování při revizi a analýze podkladů pro testování. Čím dříve jsou defekty objeveny a nahlášeny, tím nižší náklady vyžaduje jejich oprava. Pokud je například během analýzy testů identifikován nekorektní požadavek a je opraven ještě předtím, než ho vývojáři implementují do kódu, předejde se ztrátě času na implementaci a otestování nekorektního požadavku v pozdějších fázích projektu. Nehledě na to, že pokud by byl požadavek zachycen až během akceptačních testů uživatelem, hrozí i narušení důvěry uživatele v daný systém.

Hlavní zodpovědnost za zakládání defekt reportů mají testeři, kteří objevují nejvíce defektů při provádění testů. Pokud ovšem defekty naleznou i jiní členové projektového týmu, mohou defekt report buď sami založit, nebo defekt předat testerům, kteří jej prověří a následně defekt report založí. Standard IEEE 829 definuje strukturu pro report o anomálii. Tato struktura rovněž platí i pro defekt report. Uspořádání polí reportu o anomálii ve standardu IEEE je příliš obecné a v praxi používaným šablonám defekt reportu vzdálené. Místo typických položek defekt reportu závažnost a priorita jsou zde položky s názvy dopad *(impact)* a hodnocení naléhavosti defektu z pohledu zakladatele defekt reportu *(originator´s assessment of urgency)*. Do položky dopad jsou mimo jiné zahrnuty i odhady času, úsilí a rizik na opravu defektu, což obvykle bývá samostatná položka. Také se neztotožňuji se zde navrženou položkou kontext *(context)*, která slučuje mnoho informací na jednom místě (identifikaci softwaru, proces životního cyklu softwaru, verzi testované položky, úroveň testování a odkazy na testovací skript, testovací případ a test log). Z těchto důvodů zde uvedená Tabulka č. 21 popisuje strukturu defekt reportu dle zdrojů [Kaner, 1993, s. 68-76; Šplíchalová, 2008, s. 55-56; Ramdeo, 2011; Software testing fundamentals, 2011], která je dle mého názoru praxi bližší a názornější.

**Tabulka č. 21 Položky defekt reportu, zdroj: [Kaner, 1993, s. 68-76; Šplíchalová, 2008, s. 55-56; Ramdeo, 2011; Software testing fundamentals, 2011]**

| Položka defekt reportu | Popis |
|---|---|
| **Identifikace defektu** | Jedinečný identifikátor daného defektu, který je obvykle automaticky přidělen systémem pro správu defektů. |
| **Shrnutí** | Stručný a výstižný popis defektu. Ze shrnutí musí být jasné, k čemu se defekt vztahuje a jak je kritický. Často se jedná spolu s prioritou a závažností defektu o jedinou část, která zajímá manažery. Ke shrnutí je proto třeba přistupovat jako k titulku v novinách. |
| **Produkt** | Specifikace produktu, na kterém byl defekt nalezen. |
| **Komponenta** | Specifikace komponenty nebo modulu produktu, kde byl defekt nalezen. U komplexních produktů, kde je mnoho komponent či modulů, mohou manažeři podle určení konkrétní komponenty snadno předat defekt na odpovědnou osobu. |
| **Identifikace verze a release** | Identifikace kódu, který je na testu. Například identifikátor verze může být 1.01m a produkt může být nasazen jako release 1.01. Písmeno verze m značí, že se jedná o třináctý draft release 1.01 uvolněné pro testování. |
| **Datum nálezu** | Datum nalezení defektu (většinou se shoduje s datem zadání defektu reportu do systému). Datum nálezu je důležitý, zejména když nejsou dostupné informace o verzi produktu. |
| **Typ reportu** | Určení, zda se jedná skutečně o defekt jako je chyba v kódu, chyba návrhu, chyba ve specifikaci, hardwarová chyba, anebo se nejedná přímo o defekt, ale např. o návrh na vylepšení nebo o dotaz, pokud si tester není jistý, zda je chování systému korektní či nikoliv. |
| **Závažnost** | Dopad defektu na produkt vyjádřený v pětistupňové škále s hodnotami méně významná *(minor)*, středně významná *(normal)*, velmi významná *(major)*, kritická *(critical)*, blokující *(blocker)*. Podrobnější popis je uveden v kapitole 6.1.2. |
| **Priorita** | Dopad defektu na byznys uživatele, podle kterého se určuje priorita prací. Je určena hodnotami v třístupňové škále nízká *(low)*, střední *(medium)*, vysoká *(high)*. Podrobnější popis je uveden v kapitole 6.1.2. |

| | |
|---|---|
| **Popis defektu** | Podrobný popis defektu. Uvádí, zda je defekt reprodukovatelný a poskytuje dostatek informací k tomu, aby jej vývojáři i testeři mohli znovu nasimulovat. Součástí popisu defektu by měl být přesný popis kroků, které vedou k chybovému stavu včetně vstupních údajů, aktuální výsledky po provedení popsaných kroků, očekávané výsledky a prostředí, na kterém byl daný defekt nalezen (konfigurace počítače, databáze, operační systém, webový prohlížeč). |
| **Autor reportu** | Jméno osoby, která založila defekt report. |
| **Aktuálně přiděleno na** | Jméno osoby, na kterou je defekt momentálně přiřazen. Nejčastěji se jedná o vývojáře, který má na starosti opravu defektu, nebo o testera, který má za úkol defekt retestovat. |
| **Status defektu** | Identifikace současného stavu defektu. Základní sekvence stavů defektu může být následující: Nový, otevřený, přidělený, opravený, uzavřený. Podrobnější popis je uveden v kapitole 6.1.3. |
| **Přílohy** | Jakákoliv dodatečná informace, která pomáhá developerům při pochopení problému a nasimulování selhání. Může se jednat o logy aplikace, testovací data, zachycení stisku kláves, printscreeny, výpisy z databáze, apod. |
| **Komentáře** | Prostor pro další komentáře vztahující se k defektu. Obvykle se zde k defektu vyjadřují vývojáři, testeři, technická podpora, produktový manažer, popř. další osoby. Řeší se zde, zda bude defekt opraven a jak, resp. důvody zamítnutí opravy defektu nebo důvody pro opětovné otevření defektu. |

Dobrý defekt report by dle [Kaner, 1993, str. 76-78] měl být v psané podobě, očíslovaný, jednoduchý (neslučovat více defektů do jednoho defekt reportu), srozumitelný, reprodukovatelný (viz kapitola 6. 2. 5. 2.), čitelný a nekritický (držet se pouze objektivních faktů a vypustit emoce).

V nástroji IBM Rational Quality Manager implicitně není k dispozici záložka s defekt reporty. Poskytovatelem defekt reportů je aplikace Change and Configuration Management (CCM). Z aplikace Quality Management je možné pouze defekt report založit a nikoliv jej dále spravovat. Pro správu defektů přímo z Quality Management je nejprve nutné integrovat Quality Management s Change and Configuration Management. Poté je v Quality Management k dispozici záložka *„Defects“*, kde lze defekty spravovat a propojovat s ostatními pracovními položkami stejně jako v Change and Configuration Management. [IBM, 2012]

Struktura polí defekt reportu v IBM Ratinal Quality Manager je neměnná, ale je obdobná jako výše uvedená vzorová struktura. V příloze C je zařazen jednak formulář pro vytváření nového defekt reportu, který nabízí aplikace Quality Management a jednak již vytvořený defekt report, který je umístěn v aplikaci Change and Configuration Management. Oproti struktuře defekt reportu navržené zde jsou v založeném defekt reportu navíc odhady času potřebného k opravě a předpokládané datum nasazení opravy (doplňuje se až dodatečně, po vytvoření defekt reportu). Pole pro identifikaci verze je rozděleno na verzi, ve které byl defekt nalezen a verzi, ve které bude, resp. je nasazena oprava. Také zde není přímo kolonka na specifikaci produktu a jeho komponenty, ale naopak je zde kolonka na určení projektové oblasti, ke které se defekt vztahuje a možnost defekt oštítkovat.

Další informace o tomto pracovním produktu zachycuje Tabulka č. 22.

**Tabulka č. 22 Defekt report, zdroj: [autor; Kaner, 1993, s. 76]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Pokud nejsou evidovány defekt reporty, hrozí, že se na defekt zapomene, nebo že se opraví a tester už si přesně nevzpomene, jaké kroky k selhání vedly, takže není schopen potvrdit správnost opravy defektu. Tester a programátor však nejsou jediní, kdo potřebuje mít informace o defektu. O řešených defektech potřebují mít přehled i ostatní testeři, manažeři i pracovníci na help-desku. [Kaner, 1993, s. 76] |
| **Podmínky, kdy není pracovní produkt potřebný** | Defekt reporty jsou zakládány vždy, když je nalezen nový defekt. |
| **Forma pracovního produktu** | Defekt reporty je vhodné zakládat a spravovat přímo v nástroji IBM Rational Quality Manager, kde je lze snadno navázat na konkrétní testovací skript či případ. V případě, že nastane selhání při provádění testovacího skriptu, je možné založit defekt přímo ze skriptu, což má tu výhodu, že kroky, které vedou k selhání, se do defekt reportu automaticky předvyplní. |

## 6.5.6 Hlavní aktivita monitorování, reportování a řízení testování

### *6. 5. 6. 1. Pracovní produkt report o stavu testování*

Proces testování je průběžně monitorován. Výsledky sledování a měření průběhu testovacích aktivit jsou pravidelně sumarizovány do reportu o stavu testování.

V terminologickém slovníku ISTQB je report o stavu testování, na rozdíl od standardu IEEE 829, nazýván jako report o průběhu testování *(test progress report)*, ale jedná se o tentýž pracovní produkt. Report o průběhu testování je dle [ISTQB, 2012c, s. 46] slov-

níku definován jako dokument, který sumarizuje aktivity testování a jejich výsledky a který je vytvářen v pravidelných intervalech pro účely reportování průběhu testovacích aktivit ve srovnání s původním plánem testování.

Za vytváření průběžných neformálních reportů o stavu testování uvnitř testovacího týmu jsou zodpovědní testeři a test analytici, kteří dávají test manažerovi průběžná hlášení o aktuálním stavu testování na projektu. Test manažer poté hlášení dodaná členy testovacího týmu reviduje a sumarizuje do formálního reportu o stavu testování, který předává mimo testovací tým projektovému manažerovi a případně dalším na projektu zainteresovaným stranám.

Dle standardu IEEE 829 report o stavu testování zahrnuje informace, které zachycuje Tabulka č. 23.

**Tabulka č. 23 Položky reportu o stavu testování, zdroj: [IEEE, 2008, s. 63-64]**

| Položky reportu o stavu testování | Popis |
|---|---|
| **Shrnutí stavu testování** | Shrnuje aktuální výsledky provedených testů ve srovnání s plánovanými testy a sumarizuje aktuální versus očekávaný stav defektů. |
| **Změny oproti plánům** | Souhrn testů, které měly být podle plánu provedeny, ale dosud provedeny nebyly, včetně důvodů, proč nemohly být provedeny. Dále jsou zde uvedeny informace o změnách ve zbylých testech, jako jsou změny v harmonogramu provádění testů, seznam testů, které již nebudou prováděny a seznam testů, které musí být provedeny opětovně. |
| **Stav metrik testování** | Poskytuje průběžný stav metrik, jejichž sledování je vyžadováno v plánu testování a popisuje všechny rozdíly aktuálních hodnot metrik proti deklarovaným cílům. |

Tato struktura je obecným rámcem pro představu, jaké informace by měl report o stavu testování obsahovat. Konkrétní informace zahrnuté do reportu o stavu testování závisí na informačních potřebách zainteresovaných stran, kterým je report dodáván. Pro projektového manažera budou pravděpodobně důležité detailní informace o defektech, kdežto pro byznys manažera budou důležitější informace o pokrytí produktových rizik.

Pro podporu vytváření reportů o stavu testování nabízí nástroj IBM Rational Quality Manager celou řadu předdefinovaných reportů. Je možné vygenerovat například seznam defektů na základě vybraných parametrů, trend vytváření a uzavírání defektů, přehled provedených testů a jejich výsledků, trend aktuálního průběhu testů proti plánovanému průběhu, aktuální pokrytí požadavků testovacími případy a další. Kromě přednastavených

reportů je možné si také definovat vlastní šablony reportů a vytvářet tak reporty pro specifické potřeby projektu. [IBM, 2012]

Další informace o tomto pracovním produktu zachycuje Tabulka č. 24.

**Tabulka č. 24 Report o stavu testování, zdroj: [autor]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Pokud nejsou vytvářeny reporty o stavu testování, projektový manažer nemá žádný podklad pro rozhodování a korektivní akce a je závislý na stanovisku test manažera. Bez reportů o stavu testování má test manažer problémy při obhajování zpoždění testování oproti harmonogramu, protože členové projektového týmu neznalí testování často nechápou důvody zpoždění testování a testovací tým je vnímán jako brzda projektu, i když vina nemusí být nutně na straně testovacího týmu. Naopak pokud test manažer dodává informace o průběhu testování ve srozumitelné podobě, kde je jasně vidět, proč testování neprobíhá podle plánu a kdo na tom nese svou vinu, projektový manažer se na základě dodaných podkladů může rozhodnout, jak se bude postupovat dále a jestli je nutná např. redukce rozsahu testování nebo zastavení testování a předání release aplikace zpět vývojovému týmu k opravám. |
| **Podmínky, kdy není pracovní produkt potřebný** | Reporty o stavu testování je třeba vytvářet vždy. |
| **Forma pracovního produktu** | Forma reportu o stavu testování není striktně dána a závisí na potřebách na projektu zainteresovaných stran a na specifikách projektu a organizace. Pro podporu vytváření reportů o stavu testování je vhodné využít předdefinovaných reportů v IBM Rational Quality Manager, nebo si nadefinovat vlastní šablonu pro report. |

## 6.5.7 Hlavní aktivita vyhodnocení testů

### *6. 5. 7. 1. Pracovní produkt souhrnný report za testování*

Monitorování testování prostupuje skrze celý proces testování a shromažďuje informace o probíhajících aktivitách testování a o stupni naplnění výstupních kritérií testování. Po ukončení provádění testů je třeba výsledky testování zhodnotit proti cílům testování a výstupním kritériím definovaným plánem testování a následně je shrnout do souhrnného reportu za testování.

Souhrnný report za testování je dle [ISTQB, 2012c, s. 47] definován jako „*dokument, který shrnuje aktivity a výsledky testování. Zároveň zahrnuje vyhodnocení odpovídajících položek testování proti výstupním kritériím*“. Dle standardu IEEE 829 ještě také může obsahovat různá doporučení založená na zjištěných výsledcích [2008, s. 64].

Za vytváření souhrnného reportu za testování je primárně zodpovědný test manažer. Test analytici a testeři přispívají shromažďováním informací o vývoji metrik z nástrojů pro řízení testování a pomáhají při vyhodnocování celkového pokrytí požadavků a rizik testy. Na základě informací dodaných test analytiky a testery vytváří test manažer souhrnný report, který následně předává mimo testovací tým projektovému manažerovi a případně dalším na projektu zainteresovaným stranám.

Dle standardu IEEE 829 souhrnný report za testování zahrnuje informace, které zachycuje Tabulka č. 25.

**Tabulka č. 25 Položky souhrnného reportu za testování, zdroj: [IEEE, 2008, 64-66; SQE, 2001d]**

| Položky souhrnného reportu za testování | Popis |
|---|---|
| **Souhrnný přehled o testování** | Souhrnné vyhodnocení testovaných položek. Identifikuje otestované položky a případné rozdíly oproti položkám specifikovaným v plánu testování, včetně informací o verzi, resp. release dané položky. Dále uvádí informace o prostředí, na kterém testování probíhalo, a o jeho přesném nastavení a vlivu na testování, opět včetně rozdílů oproti testovacímu prostředí specifikovaném v plánu testování. Také zde mohou být uvedeny odkazy na další dokumenty a materiály, které podporují tento report. |
| **Shrnutí výsledků testování** | Identifikuje všechny vyřešené defekty a shrnuje způsob jejich řešení, popř. se odkazuje na místo, kde jsou tyto informace k dispozici. Dále identifikuje všechny nevyřešené defekty. Pokud je řešení některých defektů odloženo, vysvětluje nebo odkazuje na postup, jak s nimi bude naloženo. Kromě rozlišení defektu podle stavu je vhodné defekty také kategorizovat dle závažnosti a priority. |
| **Shrnutí aktivit testování** | Shrnuje hlavní události a aktivity testování a k nim odpovídající získané metriky. Reportuje změny proběhlých aktivit oproti plánovaným aktivitám a obzvláště tam, kde objem skutečného úsilí a času na testování přesáhl plánované úsilí a čas, uvádí důvody pro tyto změny, včetně možných dopadů na následující projekty a aktivity testovacího týmu. |

| | |
|---|---|
| **Odchylky** | Odchylky testovacích položek oproti jejich specifikacím a odchylky proti testovací dokumentaci (například změny v testech proti plánu testování, nebo testy, které nebyly provedeny). Soustředí se především na oblasti, které mohou vyvolat znepokojení při akceptaci výsledků testování. Zároveň jsou specifikovány důvody pro každou odchylku nebo skupinu odchylek nebo je uveden odkaz na místo, kde jsou tyto důvody zaznamenány. |
| **Komplexní vyhodnocení** | Vyhodnocuje úplnost procesu testování směrem k definovaným cílům testování a výstupním kritériím specifikovaným v plánu testování. Cíle a výstupní kritéria testování jsou vyhodnocovány na základě získaných hodnot metrik, jako je například počet plánovaných testovacích případů, které skončily úspěšně či neúspěšně nebo pokrytí požadavků a rizik provedenými testy. Dále uvádí, jak bylo testování účinné, jaké byly v procesu testování slabiny a zejména co bylo příčinou odchylek a neočekávaných trendů. |
| **Odůvodnění rozhodnutí** | Specifikuje problémy, které byly projednávány před učiněním rozhodnutí a zdůvodňuje, jak se došlo k výsledku daného rozhodnutí. |
| **Závěry a doporučení** | Specifikují celkové zhodnocení každé testované položky, včetně jejich omezení. Hodnocení by mělo být objektivní, založené na výsledcích testování a na jasně definovaných výstupních kritériích pro danou položku testování. Ve vyhodnocení mohou být zahrnuty také odhady pravděpodobnosti selhání a seznamy vypuštěných, nedokončených nebo jen částečně dokončených funkcí či vlastností dané položky. Navrhuje, zda testované položky mohou být nasazeny do produkce ihned, nebo jakmile nudou vyřešeny určité defekty, nebo až po kompletní revizi. |

Pro podporu vytváření souhrnných reportů za testování je možné, podobně jako při vytváření průběžných reportů, využít možnosti automatického generování reportů v nástroji IBM Rational Quality Manager.

Další informace o tomto pracovním produktu zachycuje Tabulka č. 26.

**Tabulka č. 26 Souhrnný report za testování, zdroj: [autor]**

| | |
|---|---|
| **Možné následky neexistence pracovního produktu** | Pokud nejsou vytvářeny souhrnné reporty za testování, test manažer nemá žádný důkaz o naplnění cílů a výstupních kritérií testování. Testování se z pohledu ostatních členů projektového týmu a na projektu zainteresovaných stran jeví jako černá díra. Neví se, co bylo otestováno, jaké defekty byly či nebyly vyřešeny a proč, jaké jsou příčiny případného překročení časového limitu na testování, atd. Souhrnné reporty za testování jsou důležité i pro testovací tým. V případě, kdy projekt bude v budoucnu předmětem změnových požadavků a nebyl by k dispozici souhrnný report z posledního testování, testovací tým nemůže plynule navázat na výsledky předchozího testování a musí nejdříve složitě zjišťovat, v jakém stavu byl daný produkt nasazen na produkci. |
| **Podmínky, kdy není pracovní produkt potřebný** | Souhrnné reporty za testování je třeba vytvářet vždy. Míra detailu souhrnného reportu závisí na projektu a organizaci. Nezbytné je, aby souhrnný report prokázal naplnění cílů a výstupních kritérií testování. Ostatní informace mohou být zredukovány nebo na ně může být z reportu odkázáno. |
| **Forma pracovního produktu** | Forma souhrnného reportu závisí na potřebách zainteresovaných stran projektu a na specifikách projektu a organizace. Pro podporu vytváření souhrnných reportů za testování je vhodné využít předdefinovaných reportů v IBM Rational Quality Manager, nebo si nadefinovat vlastní šablonu pro report. |

## 6.6 Shrnutí kapitoly

V této kapitole byla navržena metodika testování podle mezinárodních praktik a standardů, která je stěžejní částí této diplomové práce. V úvodu metodiky je uvedeno, z jakých zdrojů metodika čerpá, přičemž stěžejním zdrojem je jsou především materiály pro přípravu na certifikace ISTQB a standard IEEE 829. Následuje popis prvků metodiky, vymezení zaměření metodiky především na úroveň systémových testů a dynamické pojetí testování a objasnění vazby mezi metodikou a nástrojem IBM Rational Quality Manager. Dále jsou diskutovány možnosti využití metodiky, přičemž je zdůrazněno, že se jedná o metodiku, která nebyla vytvořena na míru konkrétní společnosti a je možné ji potřebám společnosti přizpůsobit. Na konci úvodní části je odkázáno také na interaktivní podobu

metodiky, která byla vytvořena a publikována na webových stránkách pomocí nástroje EPF Composer.

V dílčích podkapitolách byly podrobně popsány jednotlivé prvky metodiky. Po úvodní charakteristice metodika specifikuje koncepty, které doplňují ostatní prvky metodiky o potřebnou terminologii a další specifikace. Dále byly uvedeny hlavní aktivity testování a zasazeny do kontextu procesu testování. Následně byly jednotlivé hlavní aktivity a jejich úlohy popsány. Dále byly specifikovány čtyři vybrané techniky, které je možné při vykonávání úloh využít. Následně byly specifikovány tři role, kterými jsou určeny zodpovědnosti za jednotlivé úlohy a pracovní produkty metodiky. Posledním prvkem metodiky jsou pracovní produkty, které jsou vytvářeny v rámci hlavních aktivit a jejich dílčích úloh, přičemž je respektován standard IEEE 829.

V následující kapitole je navržená metodika posouzena proti druhé úrovni zralosti dle modelu TMMi, přičemž je zhodnoceno, které oblasti metodika v této podobě pokrývá a kde je ještě prostor pro možná rozšíření.

# 7 Zhodnocení metodiky testování proti druhé úrovni zralosti dle modelu TMMi

V předchozí kapitole byla navržena metodika testování podle mezinárodních praktik a standardů. Za účelem prokázání, že v navržené metodice jsou skutečně zapracovány praktiky charakteristické pro standardizovaný proces testování, jsem se rozhodla návrh metodiky uzavřít zhodnocením metodiky podle modelu pro zlepšování procesů testování Test Maturity Model integration (TMMi). Důvodem výběru modelu TMMi pro účely hodnocení metodiky je soulad TMMi s materiály pro ISTQB certifikace, ze kterých navržená metodika vychází a fakt, že TMMi se snaží procesy testování standardizovat.

Vzhledem k tomu, že zlepšování procesů testování a charakteristika modelu TMMi je velmi široká problematika a přesahuje rámec této diplomové práce, ale zároveň není možné na tomto místě pracovat s modelem TMMi zcela bez kontextu, zařadila jsem zpracování této problematiky do přílohy D. V příloze D je vysvětlen účel a princip modelů pro zlepšování procesů a následně je charakterizován model TMMi, přičemž je vysvětlena struktura modelu, úrovně zralosti a jakým způsobem probíhá hodnocení zralosti procesů testování dle tohoto modelu. V této kapitole se proto samotným modelem TMMi nezabývám a soustředím se pouze na hodnocení navržené metodiky dle modelu TMMi.

Model TMMi se mimo jiné využívá pro určení aktuální úrovně zralosti procesů testování na celopodnikové úrovni, což je předpokladem pro jejich následné zlepšování. Aby organizace dosáhla určité úrovně zralosti, je nutné splnit širokou škálu požadavků na procesy testování, které model TMMi pro danou úroveň zralosti definuje, přičemž pro dosažení vyšších úrovní zralosti je nutné splnit požadavky všech předchozích nižších úrovní. Již na druhé úrovni zralosti (řízená), která je prvním stupněm zralosti, kterého by se organizace měla snažit dosáhnout, model definuje 5 procesních oblastí zlepšení a vyžaduje splnění 17 specifických cílů a 2 generických cílů. K jednotlivým cílům je specifikováno široké množství praktik, které by organizace typicky měla implementovat, aby požadovaného cíle dosáhla.

Navržená metodika testování nepokrývá kompletně všechny požadavky TMMi na druhou úroveň zralosti procesů testování. Jelikož se jedná o metodiku, která není vázána na konkrétní organizaci, nezaobírá se **politikou testování**, za kterou je odpovědné vedení organizace. Stejně tak metodika nespecifikuje, jak má vypadat strategie testování, ve které jsou popsány jednotlivé úrovně testování v organizaci a jak má probíhat testování v rámci jednotlivých úrovní. Metodika se soustředí primárně na úroveň systémových testů a nikoliv na všechny úrovně testování, tudíž se nejedná přímo o **strategii testování**, ale může sloužit při vytváření strategie testování pro konkrétní organizaci jako opěrný bod. V rámci hlavní aktivity plánování testů metodika zmiňuje, že při plánování testů by se z politiky a strategie testování pro danou organizaci mělo vycházet, nebo zdůvodnit případné odchylky.

Ani ostatní procesní oblasti nejsou metodikou pokryty plně, aby byly splněny všechny cíle, které TMMi vyžaduje. Pokud by metodika měla pokrývat všechny cíle dle TMMi, její rozsah by byl mnohem větší. Nicméně metodikou jsou rámcově pokryty hlavní charakteristiky procesů testování na úrovni 2, a pokud se organizace rozhodne tuto metodiku využít na cestě k dosažení druhé úrovně zralosti, je možné ji rozšířit tak, aby zahrnovala praktiky, které povedou k naplnění cílů vyžadovaných na druhé úrovni zralosti.

Organizace TMMi Foundation, která je tvůrcem metodiky TMMi, dosud nevydala žádnou oficiální metodu pro hodnocení úrovní zralosti, podle které by mohly podniky při hodnocení úrovně zralosti postupovat. K dispozici je pouze dokument s požadavky na hodnocení úrovně zralosti procesů testování TAMAR a předpokládá se, že si organizace vyvinou vlastní metodu hodnocení. Tento nedostatek odhalil Tomáš Došek a ve své diplomové práci na téma *„Modely zlepšování procesů testování softwaru a zajištění kvality“* [2012] navrhl metodu určenou pro neformální hodnocení. Ačkoliv tato metoda může organizacím pomoci při hodnocení úrovně zralosti procesů testování, pro účely hodnocení zde navržené metodiky není vhodná. Bylo by nutné porovnávat množství jednotlivých cílů a praktik definovaných modelem TMMi se stavem procesů, které definuje navržená metodika. Takto detailní hodnocení je nad rámec této práce. V této kapitole hodnocení nejde do detailu na úroveň jednotlivých cílů a praktik, které model TMMi definuje. Jsou zhodnoceny pouze hlavní charakteristiky druhé zralostní úrovně, uvedené v příloze D.1.2.

Cílem hodnocení je zjistit, jak jsou charakteristiky druhé zralostní úrovně pokryty navrženou metodikou a upozornit na oblasti, které metodikou pokryty nejsou, nebo jsou pokryty nedostatečně a je zde prostor pro další rozšíření. Pokrytí charakteristik druhé úrovně zralosti modelu TMMi navrženou metodikou zachycuje Tabulka č. 27.

**Tabulka č. 27 Pokrytí charakteristik druhé úrovně zralosti TMMi navrženou metodikou testování, zdroj: [TMMi Foundation, 2012a, s. 23]**

| Charakteristiky druhé úrovně zralosti dle TMMi | Pokrytí charakteristik navrženou metodikou testování |
|---|---|
| Testování je jasně oddělené od debuggování. | Je splněno. Za testování a analýzu příčin defektů je zodpovědný testovací tým, kdežto za lokalizaci defektů na úrovni kódu a jejich opravu jsou zodpovědní vývojáři. |
| Existuje politika a strategie testování na úrovni organizace, nebo alespoň programová strategie pro skupinu několika projektů. | Politika testování je mimo rámec této metodiky, jelikož se jedná o dokument, za který je odpovědné vedení organizace. Za strategii testování v dané organizaci zodpovídá test manažer nebo quality manažer. Tato metodika, která je zaměřená na proces testování uvnitř úrovně systémových testů může sloužit jako východisko pro strategii testování. |

| | |
|---|---|
| Jsou vytvářeny plány testování. | Je splněno. Metodika obsahuje aktivitu plánování testů a specifikaci pracovního produktu plán testování. |
| Přístup k testování, který je součástí plánu testování, je založen na výsledcích hodnocení produktových rizik. | Je téměř splněno. Metodika doporučuje přístup k testování založený na rizicích a součástí plánování testů je úloha analýza rizik testování, na základě které je definován přístup k testování. Při plánování testů jsou rizika zohledněna, nicméně přístup k testování založený na rizicích není metodikou přímo vyžadován. |
| Plán testování definuje, co je cílem testování, kdy bude testování probíhat, jak a kdo za něj bude odpovědný | Je splněno. Položky plánu testování jsou metodikou specifikovány v pracovním produktu plán testování. Součástí plánu testování je specifikace položek testování, testovaných vlastností, přístupu k testování, vstupních a výstupních kritérií, harmonogram testování, součinnosti a zodpovědnosti za testování a další položky. |
| Jsou vymezeny závazky jednotlivých stran zainteresovaných na testování. | Je splněno. Plán testování definuje položky součinnosti a zodpovědnosti. |
| Testování je monitorováno a řízeno směrem k naplnění stanoveného plánu testování. Pokud jsou v průběhu testování zjištěny nějaké odchylky, jsou provedena nápravná opatření. | Je splněno. Metodika zahrnuje úlohy monitorování a řízení testování. Tyto úlohy prostupují celým procesem testování. Monitorování poskytuje informace o probíhajících aktivitách testování a pokud jsou zjištěny odchylky od stanoveného plánu testování, jsou provedeny řídící a korektivní akce, jejichž účelem je nasměrovat testování k naplnění stanovené mise, strategie a cílů. |
| Stav pracovních produktů testování a testovacích prací je dáván na vědomí managementu. | Je splněno. Součástí metodiky je úloha průběžné reportování o stavu testování a pracovní produkty report o stavu testování a souhrnný report za testování, prostřednictvím kterých jsou managementu dávány na vědomí informace o stavu testovacích prací a výsledcích testování. |
| Při vytváření testovacích případů se využívají techniky návrhu testů. | Metodika specifikuje dvě techniky návrhu testů a doporučuje jejich využití při návrhu testovacích případů. Konkrétně se jedná o techniky rozdělení tříd ekvivalencí a analýza hraničních hodnot. Specifikace dalších technik je námětem pro rozšíření metodiky. |

| | |
|---|---|
| Testování se do životního cyklu vývoje softwaru zapojuje relativně pozdě, nejčastěji až během fáze návrhu nebo implementace software. | Tímto problémem se metodika zabývá. Metodika uvádí, že plánování testů by mělo začínat co nejdříve, ideálně současně s plánováním projektu. V okamžiku, kdy je vytvořen plán testování, navazuje analýza testů, která začíná úlohou analýza podkladů pro testování. Účelem této úlohy je odhalit defekty již v podkladech pro testování a implementaci a upozornit na ně ještě před jejich implementací v kódu. Podklady jsou ovšem revidovány ad hoc a nejedná se o organizované revize. |
| Testování je rozděleno do jednotlivých úrovní (testy komponent, integrační testy, systémové testy a akceptační testy). Pro každou úroveň testů jsou v rámci strategie testování specifikovány určité cíle. | Metodika pokrývá pouze úroveň systémových testů, která oproti ostatním úrovním vyžaduje nejvyšší stupeň formálnosti a největší množství prvků, které je třeba organizovat. Pro účely ostatních úrovní testů je možné metodiku přizpůsobit. |
| Hlavním cílem testování je ověřit, zda produkt uspokojuje definované požadavky. | Je splněno. V úloze definice cílů a rozsahu testování je vymezen základní cíl testování jako ověření, *„že se systém chová při svém použití cílovými zákazníky a uživateli tak, jak se od něj očekává“* [Doležel, 2013c]. V konceptu definice defektu a příbuzných termínů je potom ověření chování systému rozlišeno na proces verifikace a validace. |
| Defekty, které vznikly při definování požadavků a při specifikaci návrhu se dostávají až do kódu. Nejsou zde žádné formální revize pracovních produktů a kódu, které by se na tento problém soustředily. | Metodika tento problém částečně řeší. Ještě před vytvářením pracovních produktů testování je v rámci analýzy testů zařazena úloha analýza podkladů pro testování, viz výše v této tabulce. O formální revize pracovních produktů a kódu se ovšem nejedná. |
| Testování je považováno za fázi, která následuje až po implementaci. | Po implementaci následuje provádění testů, které, jak uvádí metodika, je sice ústřední a nejviditelnější aktivitou procesu testování, ale na druhou stranu je v metodice zmíněno, že bez předchozího plánování, analýzy, návrhu a implementace testů by samotné provádění testů bylo méně účinné a efektivní. |

V procesní oblasti **plánování testů** se metodika nezabývá plánováním aktivit testování, jejich časovými odhady, milníky a rozpočtem. Z **generických cílů** napříč všemi procesními oblastmi se metodika nesoustředí na přidělování zodpovědností za jednotlivé aktivity, řízení konfigurací, školení lidí a přidělování zdrojů.

V procesní oblasti **monitorování a řízení testování** se metodika nesoustředí na detaily, jako je monitorování testovacího prostředí, monitorování projektových rizik testování a monitorování odpovědností za testování nebo monitorování interakcí se zainteresovanými stranami.

Procesní oblast **návrh a provádění testů** je touto metodikou pokryta beze sporu nejlépe. Možností rozšíření je především doplnění technik pro návrh testů, jako jsou například rozhodovací tabulky, testování přechodů mezi stavy, testování případů užití, testování uživatelských příběhů, doménová analýza a další techniky založené na defektech a zkušenostech.

V procesní oblasti **testovací prostředí** není pokryta implementace, správa a řízení testovacího prostředí, ani řízení testovacích dat. V rámci plánování testů jsou pouze definovány požadavky na testovací prostředí a v rámci implementace testů jsou vytvářena testovací data a probíhá ověření připravenosti infrastruktury pro testování.

## 7.1 Shrnutí kapitoly

V této kapitole bylo ověřeno, že navržená metodika splňuje alespoň rámcově všechny charakteristiky pro druhou úroveň zralosti procesů testování dle modelu TMMi. Model TMMi je určen pro posuzování zralosti v organizaci. Jelikož tato metodika není určena pro konkrétní organizaci, nepokrývá procesní oblast politika a strategie testování, což jsou oblasti, které závisí na potřebách organizací. Proces testování navržený v této metodice lze politice testování přizpůsobit a pro vytváření strategie testování v organizaci může být tato metodika dobrým podkladem. Metodika nejlépe pokrývá zejména oblast návrh a provádění testů, kde je možným směrem rozšíření přidání technik. Ostatními oblastmi se metodika zabývá v menším detailu a konkrétní příklady nepokrytých aspektů byly shrnuty na konci této kapitoly a mohou být podnětem pro rozšíření metodiky do budoucna.

# 8 Závěr

Hlavním cílem této diplomové práce bylo vytvořit metodiku pro organizaci procesu testování z hlediska řízení, přípravy i provádění testů, která vychází z mezinárodních praktik a standardů a není vytvořena na míru konkrétní organizaci. V praktické části diplomové práce byla navržena celá metodika podle mezinárodních praktik a standardů, která je dále rozšířitelná a přizpůsobitelná. Podle navržené metodiky budou organizace moci postupovat při definování procesu testování, a to zejména na středních a větších projektech, kde za testování zodpovídá testovací tým tvořený specializovanými testery. Reakcí na navrženou metodiku je zhodnocení této metodiky podle standardizovaného procesního rámce pro zlepšování procesů testování, z něhož vychází podněty pro možná rozšíření této metodiky. Kromě samotného návrhu metodiky je součástí práce dále hodnocení metodik vývoje softwaru z pohledu testování, standardy se vztahem k testování a certifikace v oblasti testování.

Cílů, které byly definovány v úvodu práce, se podařilo dosáhnout.

Teoretické zaměření má druhá, třetí, čtvrtá a pátá kapitola této práce.

**Ve druhé kapitole** práce, která následuje po úvodní části, byly vysvětleny základní termíny, které práce používá a jejich vzájemné vztahy. Vymezení terminologie je předpokladem pro porozumění obsahu této práce. Konkrétně se jedná o termíny informační systém, software, programový systém, životní cyklus softwaru, model životního cyklu, model životního cyklu softwaru, V-model, kvalita, zajišťování kvality softwaru, testování softwaru a úrovně testování.

**Ve třetí kapitole** práce byla uvedena definice metodiky a definice metodiky vývoje softwaru, dále byly metodiky rozděleny podle váhy na rigorózní a agilní, byly zhodnoceny základní odlišnosti rigorózních a agilních metodik z pohledu testování softwaru a následně byly z pohledu testování zhodnoceny metodiky RUP, OpenUP a MMSP. Pro hodnocení metodik OpenUP a MMSP byla použita sada kritérií definovaná v článku [Buchalcevová, 2008], ale vzhledem k velkému rozsahu práce bylo detailní hodnocení podle stanovených kritérií zařazeno do přílohy A. V samotném textu práce byla uvedena pouze základní charakteristika jednolitých metodik a souhrnné zhodnocení metodik z hlediska využitelnosti pro testování softwaru. V závěru kapitoly byly tyto tři metodiky porovnány podle nejdůležitějších rysů pro jejich využitelnost pro testování.

**Ve čtvrté kapitole** práce byl vysvětlen účel použití standardů, které se vztahují k testování softwaru a zajišťování kvality softwaru, byla uvedena definice standardu a následně byly standardy rozděleny do tří skupin podle původu na mezinárodní, národní a doménově specifické. Zvláštní pozornost byla věnována pouze mezinárodním standardům ISO/IEC a IEEE, které jsou relevantní pro testování a zajišťování kvality softwaru, přičemž bylo poznamenáno, které z nich existují mimo jiné i v podobě českých technic-

kých norem. Zvlášť charakterizovány byly standardy řady ISO 9000, ISO/IEC 12207, ISO/IEC SQuaRE, ISO/IEC/IEEE 29119 a IEEE 829. V závěru kapitoly bylo upozorněno na možný nesoulad mezi různými standardy.

**V páté kapitole** práce bylo objasněno, co to je certifikát, certifikace a proč může být získání certifikátu užitečné pro softwarové testery. Jelikož světově nejrozšířenější a nejuznávanější je systém pro certifikaci organizace ISTQB, dále byla věnována pozornost ISTQB certifikacím. V rámci podkapitoly o certifikacích ISTQB byla charakterizována organizace ISTQB, vysvětleno kvalifikační schéma, podle kterého probíhají ISTQB certifikace, přičemž bylo odkázáno na materiály pro přípravu na jednotlivé stupně certifikace. V závěru kapitoly byly diskutovány i jiné možnosti využití znalostí v materiálech pro přípravu na certifikace, než je samotné dosažení certifikátu. Důraz byl kladen především na hodnotu, kterou ISTQB přináší samotné profesi testování. V materiálech pro přípravu na certifikace jsou sumarizovány nejlepší praktiky a aktuality z oboru testování, a proto byly vybrány jako hlavní zdroj pro praktickou část práce, vytvoření metodiky testování podle mezinárodních praktik a standardů.

Námětem pro další **rozšíření teoretické části** práce může být například zhodnocení dalších metodik vývoje softwaru z pohledu testování, doplnění charakteristiky standardů ve vztahu k testování o IEEE standardy, které jsou pouze vyjmenovány a případně i doplnění komplexního přehledu a porovnání dalších možností certifikace pro softwarové testery vedle certifikace ISTQB.

V teoretické části byla rozebrána problematika, na kterou navazuje praktická část práce. Praktické zaměření má šestá a sedmá kapitola této práce.

**V šesté kapitole** práce, která je stěžejní částí této práce, byla navržena metodika testování podle mezinárodních praktik a standardů. V úvodu metodiky byl nejprve uveden důvod vytvoření nové metodiky. Následně byly shrnuty materiály, podle kterých byla metodika vytvářena, přičemž klíčovými materiály byly materiály pro přípravu na certifikace ISTQB a standard IEEE 829. Dále byly představeny prvky metodiky a vymezeno zaměření metodiky. Metodika se zaměřuje na testování softwaru v dynamickém pojetí a jejím účelem je pokrytí úrovně systémových testů. Zároveň metodika ukazuje, jakým způsobem je možné metodiku podpořit nástrojem pro řízení testů IBM Rational Quality Manager. Následně byla diskutována využitelnost metodiky, přičemž bylo zdůrazněno, že se jedná o metodiku, která není vytvořena na míru konkrétní organizaci, ani testovacímu týmu. Je vhodná zejména pro střední a větší projekty a pro testování aplikací, které nejsou vysoce kritické. Cílem práce bylo kromě vytvoření dokumentové verze metodiky také vytvoření interaktivní verze metodiky v podobě webové prezentace. Metodika byla vytvořena v nástroji pro správu metodik EPF Composer a v úvodní části šesté kapitoly je odkázáno na webové stránky, kde je vystavena aktuální verze metodiky. Jádrem šesté kapitoly byla specifikace jednotlivých metodických prvků: Koncepty, hlavní aktivity a úlohy, techniky, role a pracovní produkty (včetně šablon).

**V sedmé kapitole** praktické části práce byla navržená metodika podrobena hodnocení proti druhé úrovni zralosti dle rámce pro zlepšování procesů testování TMMi. Bylo zjištěno, že metodika rámcově pokrývá všechny základní charakteristiky druhé úrovně zralosti kromě politiky a strategie testování. Metodika se soustředí na proces testování uvnitř organizace. V případě nasazení metodiky v konkrétní organizaci je třeba proces testování přizpůsobit dle konkrétních potřeb. Při vytváření strategie testování v organizacích, je možné z této metodiky vycházet.

Metodika dobře pokrývá zejména procesní oblast návrh a provádění testů. Oblast plánování testů v této metodice se zaměřuje na čtyři vybrané úlohy zabývající se cíli, rozsahem, přístupem k testování, riziky a metrikami, ale nejsou rozpracovány úlohy zabývající se plánováním aktivit testování, jejich časovými odhady, milníky a rozpočtem. Oblast monitorování a řízení testování je pokryta v rámci hlavní aktivity monitorování, reportování a řízení testování, nicméně metodika doporučuje sledovat především základní metriky a neklade si za cíl monitorovat testovací prostředí, projektová rizika testování, odpovědnosti za testování nebo monitorovat interakce se zainteresovanými stranami. V oblasti testovacího prostředí navržená metodika nepokrývá implementaci, správu a řízení testovacího prostředí. Pokrývá pouze definování požadavků na úrovni plánování testů a ověření infrastruktury pro provádění testů před zahájením provádění testů.

Na základě zjištěných výsledků hodnocení je námětem pro **rozšíření praktické části** práce doplnění navržené metodiky o další techniky pro návrh testů, jako jsou například rozhodovací tabulky, testování přechodů mezi stavy, testování případů užití, testování uživatelských příběhů, doménová analýza a další techniky založené na defektech a zkušenostech. Dále by bylo vhodné rozšířit hlavní aktivitu plánování testů o další úlohy zaměřující se na plánování aktivit testování, jejich časové odhady, milníky a rozpočet. Dalším možným směrem rozšíření je metodické pokrytí implementace a správy testovacího prostředí. Možností rozšíření je i rozpracování sady metrik, jelikož metodikou jsou definovány pouze čtyři základní metriky. Doplnit je možné i specifikace prvků metodiky, o kterých se metodika zmiňuje, ale podrobněji je nerozpracovává. Jedná se o pracovní produkt harmonogram provádění testů a pracovní produkt poznatky pro příští projekty.

Možnost rozšíření práce vidím i v provedeném hodnocení navržené metodiky. Metodika byla zhodnocena pouze podle základních charakteristik druhé úrovně zralosti. Pro zjištění přesnějších výsledků hodnocení je možné využít neformální metodu hodnocení úrovně zralosti dle modelu TMMi, navrženou v diplomové práci [Došek, 2012], a zhodnotit navrženou metodiku na úrovni jednotlivých cílů a praktik modelu TMMi. V neposlední řadě je námětem pro rozpracování diplomové práce také praktická aplikace metodiky na konkrétním projektu v praxi a přizpůsobení metodiky pro potřeby konkrétní organizace nebo týmu.

Za hlavní **přínos práce** pokládám vytvoření nové metodiky testování, která není závislá na potřebách konkrétní společnosti a je postavena na procesu testování, který definuje světově uznávaná organizace ISTQB, jejíž misí je sumarizovat nejlepší praktiky v oboru

testování a posouvat tak profesi testování směrem kupředu. Procesu testování definovaným v materiálech ISTQB dodává navržená metodika oproti ISTQB materiálům metodický rámec. Z hlavních aktivit, které ISTQB popisuje, jsou vyčleněny a rozpracovány jednotlivé úlohy. Na úlohy jsou potom navázány role, techniky a pracovní produkty, kde většina z nich byla vytvořena podle standardu IEEE 829, nebo jsou zdůvodněny případné odchylky od standardu. Metodika může být využitelná jako výchozí bod při tvorbě testovací strategie v českých organizacích, které se zabývají vývojem softwaru a mají zájem k oblasti testování přistupovat podle mezinárodních praktik. Kromě dokumentové podoby byla metodika vytvořena a publikována pomocí nástroje pro správu metodik EPF Composer, kde je možné ji dále přizpůsobovat a rozšiřovat.

Kromě praktické části práce má dle mého názoru přínos i část teoretická, kde bylo provedeno hodnocení metodik vývoje softwaru RUP, OpenUP a MMSP z pohledu testování, přičemž metodiky OpenUP a MMSP byly zhodnoceny podle dílčích kritérií, čímž rozšiřují hodnocení metodik RUP, XP a MSF for Agile Software Development v článku [Buchalcevová, 2008]. Zároveň je přínosem teoretické části práce i zpracování standardů se vztahem k testování, které doposud nebyly předmětem zpracování žádné z dostupných českých diplomových prací ani knih.

# Terminologický slovník

| Termín (zkratka) | Význam [zdroj] |
|---|---|
| Agilní metodiky / Agile methodologies | Metodiky, které definují jen generativní pravidla, principy a praktiky. Předpokladem pro jejich zavedení je neschopnost popsat softwarové procesy a možnost předem definovat jen hrubé požadavky. [Buchalcevová, 2009, s. 70] |
| Akceptační testování / Acceptance testing | Čtvrtá úroveň testování. Jedná se o formální testování, při kterém jsou respektovány potřeby a požadavky uživatelů a podnikové procesy za účelem určení, zda systém splňuje akceptační kritéria a umožňuje uživatelům, zákazníkům nebo jiným oprávněným subjektům systém akceptovat. [ISTQB, 2012c, s. 8] |
| Analýza hraničních hodnot / Boundary value analysis | Technika černé skříňky, kdy testovací případy jsou navrženy tak, aby pokryly hodnoty na okraji tříd ekvivalencí. [ISTQB, 2012c, s. 11] |
| Analýza testů / Test analysis | Hlavní aktivita, ve které je definováno, co se bude testovat ve formě testovacích podmínek. [ISTQB, 2012a, s. 11] |
| Anomálie / Anomaly | Každý rozdíl aktuálního chování systému oproti očekáváním založeným na základě specifikací požadavků, designových dokumentů, uživatelských příruček, standardů atd. nebo na základě vnímání a zkušeností testera. [ISTQB, 2012c, s. 9] |
| Bezporuchovost / Reliability | Viz spolehlivost |
| Bug fix rate / Bud fix rate | Metrika, která vyjadřuje, kolik procent defektů bylo opraveno a vypočítá se jako podíl počtu opravených defektů a počtu nalezených defektů [Hutcheson, 2003] |
| Case nástroj / Computer-aided Software Engineering (Case) | Nástroje pro podporu procesu vývoje softwaru. Zahrnují nástroje pro podporu návrhu softwaru, správu požadavků, vytváření kódu, testování, dokumentaci a jiné aktivity softwarového inženýrství. [IEEE, 1990, s. 19] |
| Certifikace / Certification | Proces potvrzení, že komponenta, systém nebo osoba splňují specifikované požadavky. [ISTQB, 2012c, s. 12] |

| | |
|---|---|
| Certifikát / Certificate | Listina vydaná certifikační autoritou, která dokazuje určité teoretické nebo (i) praktické dovednosti a znalosti jejího držitele. [Košař, 2011, s. 9] |
| Debuggování / Debugging | Proces nalézání, analyzování a odstraňování příčin selhání v softwaru [ISTQB, 2012c, s. 17]. |
| Defekt / Defect | Chyba v komponentě nebo v systému, která může způsobit, že systém selže při vykonávání požadované funkce, jako například nekorektní příkaz nebo definice dat. [ISTQB, 2012c, s. 18] |
| Defekt report / Defect Report | Dokument, který reportuje nějaký nedostatek v komponentě nebo v systému, který může způsobit, že daná komponenta či systém selže při vykonávání požadované funkce. [ISTQB, 2012c, s. 18] |
| Dynamické testování / Dynamic testing | Testování, při kterém je vyžadováno spuštění komponenty nebo systému. [ISTQB, 2012c, s.19] |
| Efektivita / Effectiveness | Schopnost produkovat očekávaný výsledek. [ISTQB, 2012c, s. 19] |
| Funkcionalita / Functionality | Schopnost softwaru zabezpečit za předem stanovených podmínek určité funkce. [Faustová, 2009, s. 11] |
| Generické cíle / Generic goals | Povinné komponenty modelu TMMi, objevují se skrze více procesních oblastí a se používají pro vyhodnocování, zda daná procesní oblast splnila požadavky na určitou úroveň zralosti. [TMMi Foundation, 2012a, s. 14] |
| Gernerické praktiky / Generic practices | Očekávané komponenty modelu TMMi, objevují napříč několika procesními oblastmi a popisují aktivity, kterými lze nejlépe dospět k naplnění generických cílů. [TMMi Foundation, 2012a, s. 15] |
| Hlavní aktivita / Main activity | Seskupení úloh, které se vztahují k určité oblasti zájmu a jsou vykonávány ve stejném časovém období. [Autor] |
| Hustota defektů / Defect density | Počet defektů identifikovaný v komponentě nebo v systému dělený velikostí komponenty nebo systému (vyjádřené ve standardních výrazech pro měření jako jsou například řádky kódu, počet tříd nebo funkčních položek). [ISTQB, 2012c, s. 18] |
| Hustota metodiky | Vyjadřuje míru podrobnosti a těsnost tolerance metodiky, požadovanou podrobnost a konzistenci prvků. [Buchalcevová, 2009, s. 62] |

| | |
|---|---|
| Chyba / Bug | Viz defekt |
| Implementace testů / Test implementation | Hlavní aktivita, během níž jsou testy organizovány a prioritizovány a na základě navržených testovacích případů jsou připravovány testovací data a jsou implementovány manuální a případně i automatizované testovací skripty. [Autor] |
| Informační systém / Information systém (IS; IS/ICT) | Systém pro sběr, přenos, uchování, zpracování a poskytování dat (informací, znalostí) využívaných při činnosti podniku. Jedná se o systém informačních a komunikačních technologií, dat a lidí, jehož cílem je efektivně podporovat informační, rozhodovací a řídící procesy na všech úrovních řízení podniku. [Voříšek, 2008, s. 18] |
| Integrační testování / Integration testing | Druhá úroveň testování. Jedná se o testování vykonávané za účelem odhalení defektů na rozhraní a při interakci integrovaných komponent nebo systémů. [ISQTB, 2012c, s. 26] |
| Jednotkové testování / Unit testing | Viz Testování komponent |
| Koncept | Prvek metodiky poskytující informace, na které se odkazují ostatní prvky metodiky a které je vhodné vyčlenit zvlášť. [Autor] |
| Kritéria pro přerušení a požadavky na následnou obnovu / Suspension criteria and resumption requirements | Podmínky, při kterých je nutné přerušit testování. Specifikují akceptovatelnou úroveň defektů, které ještě umožní pokračovat v testování po předchozích defektech. Dále specifikují aktivity, které je nutné opakovat při obnově testování. [IEEE, 2008, s. 46] |

| | |
|---|---|
| Kvalita / Quality | Stupeň naplnění specifikovaných požadavků; stupeň naplnění potřeb a očekávání zákazníka nebo uživatele [IEEE, 1990, s 60]; Kromě naplnění požadavků uživatelů, které nejsou u všech skupin uživatelů jednotné (user based), je kvalita dále posuzována<br>• z hlediska splnění požadavků interních standardů a praktik pro vytváření produktu (manufacturing based),<br>• z hlediska splnění objektivně stanovených, kvantifikovatelných a měřitelných charakteristických atributů daného produktu (product based),<br>• z hlediska ochoty zákazníka produkt za danou cenu zakoupit (value based),<br>• z hlediska intuice, což vystihují slova „Neumím to definovat, ale vím to, když to vidím.“ (transcendental view). [David Garvin: citován v Ross, 2009] |
| Lehké metodiky | Viz agilní metodiky |
| Metodika / Methodology | Uznávané postupy a návody, které popisují činnosti při návrhu, vývoji, nasazování software, řízení projektu, atd. Cílem metodiky je formalizovat postupy, definovat zodpovědnosti a pravidla komunikace. [Buchalcevová, 2002, s. 173] |
| Metodika vývoje softwaru | Principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení. [Buchalcevová, 2009, s. 17] |
| Metrika / Metric | Rozsah měření a metoda používaná pro měření. [ISTQB, 2012c, s. 28] |
| Model životního cyklu | Rámec procesů a aktivit spojených s životním cyklem, které mohou být organizovány do stupňů. Model životního cyklu slouží také jako referenční rámec pro komunikaci a vzájemné dorozumívání. [ISO/IEC, 2008, s. 4] |
| Model životního cyklu softwaru | Rámec realizace procesů životního cyklu v časové posloupnosti. [Buchalcevová, 2009, s. 47] |
| Modely zlepšování procesů testování / Test Maturity Models | Procesní rámce, které slouží pro systematické zlepšování procesů testování s využitím v praxi ověřených postupů a praktik. Jejich účelem je optimalizování kvality, nákladů a doby realizace testování ve vztahu k celkovým informačním službám. [Andersin, 31, s. 3] |

| | |
|---|---|
| Monitorování testování / Test monitoring | Jedna z úloh ze skupiny úloh hlavní aktivity monitorování, reportování a řízení testování, jejímž úkolem je průběžná kontrola stavu testování v průběhu projektu. V rámci této úlohy jsou vytvářeny reporty, kde jsou porovnávány skutečné hodnoty s hodnotami plánovanými. [ISTQB, 2012c, s. 45] |
| Návrh testů / Test design | Hlavní aktivita, ve které je definováno, jak se bude něco testovat. Zahrnuje identifikace testovacích případů na základě definovaných testovacích podmínek nebo podkladů pro testování a využívá techniky identifikované ve strategii testování nebo v plánu testování. [ISTQB, 2012a, s. 13] |
| Omyl / Error | Akce člověka, která způsobuje nekorektní výsledek. [ISTQB, 2012c, s. 20] |
| Paradox pesticidů / Pesticid paradox | Jev, kdy při opakování stále stejné množiny testů se postupně přestávají nacházet nové defekty, protože se systém postupně vyladí tak, aby byl vůči těmto testům odolný. [ISTQB, 2011, s. 14] |
| Plán testování / Test Plan | Dokument popisující rozsah, přístup, zdroje a harmonogram testovaných aktivit. Identifikuje položky k testování, testované vlastnosti, úlohy a zodpovědnosti za ně, stupeň nezávislosti testerů, testovací prostředí, techniky návrhu testů, vstupní a výstupní kritéria a rizika vyžadující kontingenční plánování. [ISTQB, 2012c, s. 45] |
| Plánování testů / Test planning | Hlavní aktivita, při které je vytvářen a aktualizován plán testování [ISTQB, 2012c, s. 45]. Cílem plánování testů, je vymezit záměry a cíle testování a identifikovat a naplánovat všechny aktivity a zdroje, které povedou k naplnění stanovených cílů a mise definované ve strategii testování [ISTQB, 2012a, s. 9]. |
| Podklady pro testování / Test basis | Veškeré dokumenty, ze kterých je možné vyčíst požadavky na komponentu nebo systém. Jedná se o dokumentaci, na základě které jsou vytvářeny testovací případy. [ISTQB, 2012c, s. 42] |
| Podporovatelnost / Supportability | Přenositelnost vyvíjeného softwaru do různých prostředí. [Faustová, 2009, s. 12] |
| Politika testování / Test policy | Dokument zaměřený na dlouhodobé cíle testování, za který odpovídá vrcholový management organizace (popř. vedení útvaru informatiky) [Doležel, 2013b] a popisuje principy, přístup a hlavní cíle organizace z pohledu testování. [ISTQB, 2012, s. 46] |

| | |
|---|---|
| Použitelnost / Usability | Míra úsilí, které musí uživatel vynaložit, aby mohl software používat ke stanoveným cílům za stanovených podmínek. [Faustová, 2009, s. 11] |
| Pracovní produkt / Work Product | Výstup z procesu testování, který je v průběhu testování vytvářen, modifikován a používán. [IBM, 2005] |
| Priorita / Priority | Úroveň významnosti defektu z pohledu byznysu uživatele [ISTQB, 2012c, s. 33] |
| Procesní oblast / Process area | Komponenta modelu TMMi, která identifikuje, kam má organizace zaměřit zlepšování svých procesů, aby dosáhla dané úrovně zralosti. [TMMi Foudation, 2012a, s. 13] |
| Programový systém | Softwarový produkt, který je tvořen množinou programových jednotek (modulů, komponent, objektů) a jejich vzájemných vazeb. [Buchalcevová, 2002, s. 135] |
| Provádění testů / Test execution | Hlavní aktivita, ve které je spuštěn vyvíjený systém či komponenta a podroben testům za účelem porovnání aktuálních výsledků s očekávanými. [Autor] |
| Průzkumné testování / Exploratory testing | Neformální technika návrhu testů, kdy tester současně navrhuje i provádí testy a získané informace využívá pro navrhování nových a lepších testů. [ISQTB, 2012c, s. 21] |
| Přístup k testování / Test approach | Implementace strategie testování pro určitý projekt. Typicky zahrnuje rozhodnutí provedená na základě cílů projektu a testování a hodnocení rizik, východiska pro proces testování, techniky pro návrh testů, které budou aplikovány, výstupní kritéria a typy testů, které budou prováděny. [ISTQB, 2012c, s. 42] |
| Regresní testování / Regression testing | Testování systému či komponenty, která již byla otestována, ale některé její části byly předmětem změn. Cílem je ověřit, zda se v důsledku změn v softwaru nezanesly defekty i do neměněných částí softwaru. Regresní testy se provádějí při změnách v softwaru nebo v jeho prostředí. [ISTQB, 2012c, s. 35] |
| Report o stavu (průběhu) testování / Interim Test Status Report | Dokument, který sumarizuje aktivity testování a jejich výsledky a který je vytvářen v pravidelných intervalech pro účely reportování průběhu testovacích aktivit ve srovnání s původním plánem testování. [ISTQB, 2012c, s. 46] |

| | |
|---|---|
| Reportování testování / Test reporting | Jedna z úloh ze skupiny úloh hlavní aktivity monitorování, reportování a řízení testování, jejímž účelem je předávání a zpřístupňování informací o stavu testování uvnitř týmu testování i vně testovací tým. [Autor] |
| Retestování / Retesting | Provádění testovacích případů, které v minulosti nedopadly úspěšně za účelem ověření úspěšnosti korektivních akcí. [ISTQB, 2012c, s. 36] |
| Revize / Review | Proces nebo mítink, během kterého je určitý pracovní produkt prezentován členům projektového týmu, manažerům, uživatelům, zákazníkům, nebo jiným zájmovým skupinám s cílem získat jejich připomínky či souhlas. [IEEE, 1990, s. 64] |
| rigorózní metodiky | Metodiky s přesně definovanými procesy, činnostmi a artefakty. Předpokladem pro jejich zavedení je schopnost popsat softwarové procesy a předem definovat požadavky. [Buchalcevová, 2009, s. 70] |
| Riziko / Risk | Možnost, že nastane událost, nebezpečí, hrozba nebo situace, která má nechtěný dopad nebo vyústí v potencionální problém. [ISTQB, 2011, s. 53] |
| Role | Abstraktní vyjádření souboru zodpovědností za jednotlivé úlohy a pracovní produkty, které mohou být naplněny jednou nebo více osobami [IBM, 2005]. |
| Rozdělení tříd ekvivalencí / Ekvivalence partitioning | Technika černé skříňky,kdy testovací případy jsou navrženy tak, aby pokryly jednotlivé třídy ekvivalence a z každé třídy ekvivalence otestovaly alespoň jednoho zástupce. [ISTQB, 2012, s. 20] |
| Řízení testování / Test control | Jedna z úloh ze skupiny úloh hlavní aktivity monitorování, reportování a řízení testování, jejímž cílem je provádět nápravná opatření a nasměrovat projekt testování do správné roviny, pokud se při monitorování zjistí odchylky oproti plánu. [ISTQB, 2012c, s. 43] |
| Selhání / Failure | Pokud se při běhu komponenty či systému narazí na odchylku oproti očekávané dodávce, službě nebo výsledku, jedná se o selhání, které zhoršuje či znemožňuje používání systému [ISTQB, 2012c, s. 22]. |

| | |
|---|---|
| Smoke test | Podmnožina všech definovaných/plánovaných testovacích případů, které pokrývají hlavní funkcionalitu komponenty nebo systému. Jejich účelem je ověřit, zda fungují nejkritičtější funkce programu a nezaměřují se na detailní funkcionalitu. [ISTQB, 2012c, s. 39] |
| Software (SW) | Viz programový systém |
| Souhrnný report za testování / Summary Test Report | Dokument, který shrnuje aktivity a výsledky testování. Zároveň zahrnuje vyhodnocení odpovídajících položek testování proti výstupním kritériím [ISTQB, 2012c, s. 47] Může obsahovat také různá doporučení založená na zjištěných výsledcích [IEEE, 2008, s. 64]. |
| Specifické cíle / Specific goals | Povinné komponenty modelu TMMi, popisují jedinečné charakteristiky, které jsou nutné pro uspokojení dané procesní oblasti. [TMMi Foundation, 2012a, s. 14] |
| Specifické praktiky / Specific practices | Očekávané komponenty modelu TMMi, popisují aktivity, kterými lze nejlépe dospět k naplnění specifických cílů. [TMMi Foundation, 2012a, s. 15] |
| Spolehlivost / Reliability | Schopnost softwaru být za daných podmínek používán, bez toho, aby došlo k jeho výpadku či chybě. [Faustová, 2009, s. 11] |
| Standardy / Standards | Publikované dokumenty na základě dohod. Zahrnují technické specifikace nebo jiná přesná kritéria, která se důsledně uplatňují jako pravidla, směrnice nebo charakteristické definice a zaručují, že materiály, produkty, procesy a služby splní své poslání. [Buchalcevová, 2009, s. 57] |
| Statické testování / Static testing | Testování pracovních produktů vývoje softwaru, například požadavků, návrhu nebo kódu, bez spouštění těchto pracovních produktů. [ISTQB, 2012c, 41] |
| Strategie testování / Test strategy | Dokument zaměřený na střednědobé cíle testování, za který odpovídá quality manažer nebo test manažer (popř. metodik testování) [Doležel, 2013b]. Popisuje jednotlivé úrovně testování, ve kterých má testování v organizaci probíhat a jak má vypadat testování uvnitř jednotlivých úrovní. [ISTQB, 2012, s. 47] |

| | |
|---|---|
| Systémové testování / System testing | Třetí úroveň testování. Jedná se o proces testování celého integrovaného systému za účelem ověření, zda splňuje specifikované požadavky. [ISTQB, 2012c, s. 42] |
| Šablona / Template | Pomůcka pro opakované vytváření pracovních produktů. Definuje náležitosti, které má pracovní produkt mít a při vytváření vlastního pracovního produktu stačí tyto náležitosti doplnit bez potřeby řešit jejich formu. [Autor] |
| Technika / Technique | Postupy a přístupy k tomu, jak dosáhnout určitého cíle, jako je snížení míry rizika, redukce množství testovacích případů nebo zvýšení účinnosti a efektivity testování. [Autor] |
| Test analytik / Test Analyst | Role, jejíž hlavní zodpovědností je příprava podkladů pro testery. [Autor] |
| Test log / Test log | Chronologický záznam všech relevantních detailů souvisejících s prováděním testů. [ISTQB, 2012c, s. 45] |
| Test manažer / Test Manager | Role, která má na starost vedení týmu testování v průběhu celého procesu testování. Je zodpovědný za kvalitu výsledného produktu, plánování a řízení zdrojů na projektu a řešení problémů, které stojí testovacímu týmu na cestě ke splnění cílů. [RUP, 2010] |
| Tester / Tester | Role, jejíž hlavní zodpovědností je provádění testů a zaznamenávání jejich výsledků. [RUP, 2010] |
| Testovací cyklus / Test cycle | Provedení procesu testování proti jedné identifikovatelné verzi testovacího objektu. [ISTQB, 2012c, s. 43] |
| Testovací data / Test Data | Data, která existují (například v databázi) ještě před provedením testu a která ovlivňují nebo jsou ovlivňována testovanou komponentou nebo systémem [ISTQB, 2012c, s. 43]. |
| Testovací podmínka / Test Condition | Položka či událost systému nebo komponenty, která má být ověřena jedním či více testovacími případy [ISTQB, 2012c, s. 43] |
| Testovací případ / Test Case | Sada vstupů pro testování, podmínek před provedením, očekávaných výsledků a podmínek po provedení, vytvořená za určitým cílem jako například otestovat určitou větev kódu či ověřit stupeň splnění určitého požadavku. [ISTQB, 2012c, s. 43] |

| | |
|---|---|
| Testovací sada / Test Suite | Sada několika testovacích případů pro komponentu nebo systém pod testem, kde výstupní podmínky jednoho testu jsou často použity jako vstupní podmínky testu následujícího. [ISTQB, 2012c, s. 47] |
| Testovací skript / Test Script | Posloupnost akcí pro vykonání testu [ISTQB, 2012c, s. 47]; Detailní instrukce pro nastavení, provedení a vyhodnocení výsledků daného testovacího případu [IEEE, 1990, s. 75]; Může se jednat o manuální i automatizovaný testovací skript. |
| Testovaná položka / Test Item | Samostatný element, který bude předmětem testování. [ISTQB, 1012c, s. 44] |
| Testování komponent / Component testing | První úroveň testování. Jedná se o testování samostatných softwarových komponent. [ISTQB, 2012c, s. 13] |
| Testování softwaru / Software testing | Proces ověřování kvality softwaru (a případně i dalších součástí výsledného softwarového produktu) za účelem zjištění rozdílů mezi aktuálním a očekávaným chováním (resp. stavem), detekování případných defektů a poskytnutí informací o kvalitě softwaru a jeho součástí zainteresovaným stranám. [Autor] |
| Testování založené na rizicích / Risk based testing | Přístup k testování, jehož účelem je snížit úroveň produktových rizik a informovat zainteresované strany projektu o stavu těchto rizik v průběhu projektu. Začíná v již v počátečních fázích projektu a zahrnuje identifikaci produktových rizik a využívání úrovní rizik pro řízení procesu testování. [ISTQB, 2012c, s. 37] |
| Těžké metodiky | Viz rigorózní metodiky |
| Třída ekvivalence / Equivalence partition | Část vstupů nebo výstupů domény, pro kterou se na základě specifikace předpokládá, že bude vykazovat stejné chování. [ISTQB, 2012c, s. 20] |
| Účinnost softwarového produktu / Software product efficiency | Schopnost softwarového produktu poskytovat odpovídající výkon vzhledem k objemu zdrojů využívaných za určitých podmínek. [ISTQB, 2012c, s. 20]. |
| Účinnost procesu / Process efficiency | Schopnost procesu produkovat očekávaný výsledek vzhledem k objemu využívaných zdrojů. [ISTQB, 2012c, s. 20] |

| | |
|---|---|
| Úloha / Task | Nejmenší jednotky práce, které má smysl definovat jako samostatný proces. Jsou vykoná-vány rolemi a jedná se o popis jak pracovat, aby bylo dosaženo stanovených cílů či byly vytvořeny určité pracovní produkty. [MMSP, 2011] |
| Úroveň testování / Test level | Skupina testovacích aktivit, které jsou organizovány a řízeny dohromady a je vázána na zodpovědnosti v projektu. [ISTQB, 2012c, s. 45] |
| Úroveň zralosti / Maturity level | Stupeň procesu zlepšování napříč předdefinovanou skupinou procesních oblastí, ve kterých je dosahováno všech cílů dané skupiny oblastí. [ISTQB, 2012c, s. 28]. |
| Uzavření testování / Test closure activities | Hlavní aktivita, jejímž cílem je ověřit, zda byly kompletně dokončeny všechny testovací činnosti, zkompletovat pracovní produkty z testování a předat je relevantním osobám nebo týmům, které je využijí, zaarchivovat pracovní produkty a poučit se z proběhlého procesu testování do budoucna. [Autor] |
| Vada / Fault | Viz defekt |
| Váha metodiky | Je dána součinem velikosti a hustoty metodiky. [Buchalcevová, 2009, s. 62] |
| Validace / Validation | Kontrola, zda software vyhovuje požadavkům uživatele (je vytvořen správný systém). [Patton, 2002, s. 42] |
| Velikost metodiky | Vyjadřuje počet kontrolních prvků obsažených v metodice. [Buchalcevová, 2009, s. 62] |
| Verifikace / Verification | Potvrzení, že software vyhovuje zadané specifikaci (systém je vytvořen správně). [Patton, 2002, s. 42] |
| V-model | Model, životního cyklu, který ilustruje, jak mohou být aktivity testování integrovány do každé fáze životního cyklu vývoje softwaru. [ISTQB, 2012c, s. 49] |
| Vstupní kritéria / Entry criteria | Sada obecných a specifických podmínek, které je třeba splnit, aby mohl se proces mohl posunout k další definované úloze, například do další fáze testování. Účelem vstupních kritérií je zabránit zbytečnému úsilí na začátku dané úlohy, které by bylo větší ve srovnání s úsilím na odstranění nesplněného vstupního kritéria. [ISTQB, 2012c, s. 9] |

| | |
|---|---|
| Vyhodnocení testování / Test Evaluation | Hlavní aktivita, jejímž cílem je vyhodnotit, zda testování naplnilo výstupní kritéria a je možné postoupit do další úrovně testování. [Autor] |
| Výkonnost / Performance | Schopnost softwaru poskytovat požadovaný výkon vzhledem k použití stanovených zdrojů za stanovených podmínek. [Faustová, 2009, s. 12] |
| Výstupní kritéria / Exit criteria | Sada obecných a specifických podmínek, odsouhlasených se zainteresovanými stranami, které je potřeba splnit, aby mohl být proces oficiálně ukončen. Účelem výstupních kritérií je předcházet předčasnému ukončení úloh, kdy stále přetrvávají úlohy, které ještě nebyly kompletně dokončeny. Výstupní kritéria jsou předmětem reportování, jejich hodnoty jsou porovnávány proti plánu a podle hodnot výstupních kritérií se rozhoduje o tom, kdy testování ukončit. [ISTQB, 2012c, s. 21] |
| Zajišťování kvality softwaru/ Quality Assurance (QA) | Plánované a systematické procesy, jejichž cílem je zajistit, aby byl software vhodný k jeho zamýšlenému účelu. [Havlíčková, 2009] Náplní zajišťování kvality softwaru je zkoumání a měření současného procesu vývoje softwaru a nalezení způsobů jeho zdokonalení za účelem zabránění vzniku defektů. [Patton, 2002, s. 70] |
| Závažnost / Severity | Dopad defektu na vývoj nebo provoz komponenty či systému. [ISTQB, 2012, s. 38] |
| Životní cyklus defektu / Defect life cycle | Stavy defekt reportu, kterých může defekt report nabývat od svého odhalení až po uzavření, a přechody mezi nimi. [Autor] |
| Životní cyklus softwaru | Časové období, které začíná úmyslem vytvořit softwarový produkt a končí, když se software přestane nadále používat. [ISTQB, 2012c, s. 39] |
| Životní cyklus vývoje softwaru | Časové období, které začíná rozhodnutím vyvinout softwarový produkt a končí, jakmile je software dodán. [IEEE, 1990, s. 67] |

# Seznam literatury


[Andresin, 2004] ANDRESIN, Jari. TPI – a model for test process improvement [online]. Department of Computes Science. University of Heslinki. 2004-11-05 [cit. 2013-05-13] Dostupné z: http://www.cs.helsinki.fi/u/paakki/Andersin.pdf

[Bach, 2003] BACH, James. Exploratory Testing Explained [online]. 2003, v.1.3 [cit. 2013-06-10]. Dostupné z: http://people.eecs.ku.edu/~saiedian/teaching/Fa07/814/Resources/exploratory-testing.pdf

[Balduino, 2007] BALDUINO, Ricardo. Introduction to OpenUp (Open Unified Process) [online]. 2007 [cit. 2013-06-10]. Dostupné z WWW: http://www.eclipse.org/epf/general/OpenUP.pdf

[Beck, 2001] BECK, Kent et al. Manifest Agilního vývoje softwaru [online]. 2001 [cit. 2013-06-10]. Dostupné z: http://www.agilemanifesto.org/iso/cs

[Black, 2002] BLACK, Rex. Managing the testing process: practical tools and techniques for managing hardware and software testing. 2. ed. New York [u.a.]: Wiley, 2002. ISBN 04-712-2398-0.

[Black, 2008] BLACK, Rex. ISTQB Certification: Why You Need It and How to Get It. Testing experience: The Magazine for Professional Testers [online]. 2008, 01/08, s. 26-30 [cit. 2013-06-10]. Dostupné z: http://www.istqb.org/images/Articles/black-ISTQB%20Certification%20Why%20You%20Need%20It%20and%20How%20to%20Get%20It.pdf

[Borovcová, 2008] BOROVCOVÁ, Anna. Testování webových aplikací. Praha, 2008. Diplomová práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra teoretické informatiky a matematické logiky.

[Borůvka, 2009] BORŮVKA, Zdeněk. Způsoby ověření aplikací a systémů (metodika, nástroje). Praha, 2009. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Buchalcevová, 2002] BUCHALCEVOVÁ, Alena, Milan ŠIMŮNEK a Iva STANOVSKÁ. Základy softwarového inženýrství - základní témata. 1. vyd. Praha: Oeconomica, 2002, 198 s. ISBN 80-245-0346-8.

[Buchalcevová, 2008] BUCHALCEVOVÁ, Alena, KUČERA, Jan. Hodnocení metodik vývoje informačních systémů z pohledu testování. Systémová integrace, 2008, roč. 15, č. 2, s. 42–54. ISSN 1210-9479. Dostupné z: http://nb.vse.cz/~buchalc/clanky/testovani.pdf

[Buchalcevová, 2009] BUCHALCEVOVÁ, Alena. Metodiky budování informačních systémů. Vyd. 1. Praha: Oeconomica, 2009, 205 s. Vysokoškolská učebnice (Oeconomica). ISBN 978-80-245-1540-3.

[Čermák, 2009] ČERMÁK, Miroslav. Testovací log. Clever And Smart [online]. 2009 [cit. 2013-06-10]. Dostupné z: http://www.cleverandsmart.cz/testovaci-log

[ČNI, 2005] Český normalizační institut: ČSN ISO/IEC 90003. Softwarové inženýrství - Směrnice pro použití ISO 9001:2000 na počítačový software. Praha, 2005.

[Doležel, 2013a] DOLEŽEL, Michal. Proces testování. 2013. Prezentace k předmětu Řízení kvality softwaru. [cit. 2013-06-09].

[Doležel, 2013b] DOLEŽEL, Michal. Dokumentace testování. 2013. Prezentace k předmětu Řízení kvality softwaru. [cit. 2013-06-09].

[Doležel, 2013c] DOLEŽEL, Michal. Základy testování. 2013. Prezentace k předmětu Řízení kvality softwaru. [cit. 2013-06-09].

[Doležel, 2013d] DOLEŽEL, Michal. Kategorizace testů. 2013. Prezentace k předmětu Řízení kvality softwaru. [cit. 2013-06-09].

[Došek, 2012] DOŠEK, Tomáš. Modely zlepšování procesů testování softwaru a zajištění kvality. Praha, 2012. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Faustová, 2009] FAUSTOVÁ, Tereza. Nástroje na podporu testování. Praha, 2009. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Fiurášek, 2010] FIURÁŠEK, Tomáš. Návrh metodiky testování webových aplikací. Praha, 2010. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Galin, 2004] GALIN, Daniel. Software quality assurance: From theory to implementation. New York: Pearson Education Limited, 2004, xxvi, 590 s. ISBN 02-017-0945-7.

[Havlíčková, 2009] HAVLÍČKOVÁ, Anna. Proč bychom měli testovat?. *Testování softwaru* [online]. 2009 [cit. 2013-06-15]. Dostupné z: http://testovanisoftwaru.blogspot.cz/2009/12/proc-bychom-meli-testovat.html

[Hutcheson, 2003] HUTCHESON, Marnie L. Software testing fundamentals: methods and metrics. Indianapolis, Ind.: Wiley Pub., 2003, xxiii, 408 s. ISBN 04-714-3020-X.

[Chen, 2004] CHEN, Yanping, Robert L. PROBERT a Kyle ROBESON. Effective test metrics for test strategy evolution. Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research [online]. 2004, CASCON' 04 [cit. 2013-06-09]. Dostupné z: http://dl.acm.org/citation.cfm?id=1034923

[Chlapek, 2011] CHLAPEK, Dušan, Václav ŘEPA a Iva STANOVSKÁ. Analýza a návrh informačních systémů. 1. vydání. Praha: Oeconomica, 2011, 158 s. ISBN 978-80-245-1782-7.

[Chodura, 2010] CHODURA, Ondřej. Testování softwaru v metodice OpenUP. Praha, 2010. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[IBM, 2005] IBM, Rational software: TST170 Principles of Software Testing for Testers, v2002.05.00, Instructor Guide

[IBM, 2012] IBM Corporation. IBM Rational Help [online]. 2012 [cit. 2013-06-10]. Dostupné z: https://jazz-swg.gtslab.cz/clmhelp/index.jsp

[IBM, 2013] About Jazz. IBM DEVELOPERWORKS. Jazz Community Site [online]. 2013 [cit. 2013-06-10]. Dostupné z: https://jazz.net/story/about

[IEEE, 1990] The Institute of Electrical and Electronics Engineers: IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology. New York, USA, 1990. Dostupné z: http://standards.ieee.org/findstds/standard/610.12-1990.html

[IEEE, 2005] Chapter 5 Software Testing. THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. Guide to the software engineering body of knowledge: 2004 version. Los Alamitos, CA: IEEE Computer Society Press, 2005. ISBN 0769523307.

[IEEE, 2008] The Institute of Electrical and Electronics Engineers: IEEE Std 829™-2008. IEEE Standard for Software and System Test Documentation. New York, USA, 2008. Dostupné z: http://standards.ieee.org/findstds/standard/829-2008.html

[IEEE, 2010] The Institute of Electrical and Electronics Engineers: IEEE Std 1044TM-2009. IEEE Standard Classification for Software Anomalies. New York, USA, 2010. Dostupné z: http://standards.ieee.org/findstds/standard/1044-2009.html

[IEEE, 2013a] THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. IEEE: Advancing Technology for Humanity [online]. 2013 [cit. 2013-06-10]. Dostupné z: http://www.ieee.org/index.html

[IEEE, 2013b] THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. IEEE: Czechoslovakia Section [online]. 2013 [cit. 2013-06-10]. Dostupné z: www.ieee.cz

[Imperva, 2013] Cross-Site Scripting. IMPERVA. Imperva [online]. 2013 [cit. 2013-06-10]. Dostupné z: http://www.imperva.com/resources/glossary/cross_site_scripting.html

[ISO, 2013] THE INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO [online]. 2013 [cit. 2013-06-10]. Dostupné z: http://www.iso.org/iso/home.htm

[ISO/IEC, 2008] ISO/IEC, IEEE: ISO/IEC 12207: 2008, IEEE Std 12207TM-2008. Systems and software engineering – Software life cycle processes. USA, 2008. Dostupné z: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43447

[ISTQB, 2011] International Software Testing Qualification Board: Certified Tester Foundation Level Syllabus. Version 2011, 31. 3. 2011, 78 s. Dostupné z: http://www.istqb.org/downloads/viewcategory/16.html

[ISTQB, 2012a] International Software Testing Qualification Board: Certified Tester Advanced Level Syllabus Test Manager. Version 2012,1 9. 10. 2012, 82 s. Dostupné z: http://www.istqb.org/downloads/viewcategory/46.html

[ISTQB, 2012b] International Software Testing Qualification Board: Certified Tester Advanced Level Syllabus Test Analyst. Version 2012, 19. 10. 2012, 64 s. Dostupné z: http://www.istqb.org/downloads/viewcategory/46.html

[ISTQB, 2012c] International Software Testing Qualification Board – Glossary working Party, ed. Veenendaal, Erik van: Standard glossary of terms used in Software Testing. Version 2.2, 19. 10. 2012, 52 s. Dostupné z: http://www.istqb.org/downloads/viewcategory/20.html

[ISTQB, 2013a] INTERNATIONAL SOFTWARE TESTING QUALIFICATIONS BOARD. ISQTB: International Software Testing Qualifications Board [online]. 2013 [cit. 2013-06-10]. Dostupné z: www.istqb.org

[ISTQB, 2013b] INTERNATIONAL SOFTWARE TESTING QUALIFICATION BOARD. ISTQB in a Nutshell. 2013. Dostupné z: http://www.istqb.org/documents/ISTQB_Nutshell_201305_v15.pdf

[Jha, 2012] JHA, M.: Software Testing Maturity Models. Software Testing & QA [online]. 2012 [cit. 2013-05-13]. Dostupné z: http://www.rishabhsoft.com/blog/software-testing-maturity-models

[Julinek, 2008] JULINEK, Pavel. Použití RUP pro malé SW projekty [online]. Brno, 2008. Dostupné z: http://is.muni.cz/th/72639/fi_m/diplomova_prace.pdf. Diplomová práce. Masarykova univerzita, Fakulta informatiky.

[Kaner, 1993] KANER, Cem, Jack L FALK a Hung Quoc NGUYEN. Testing computer software. 2nd ed. New York: Van Nostrand Reinhold, c1993, xv, 480 s. ISBN 04-420-1361-2.

[Kocura 2010] KOCURA, Petr. Metodika OpenUP. Praha, 2010. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Kosek, 2011] KOSEK, Jiří. Bezpečnost webových aplikací. 2011. Prezentace k předmětu Tvorba webových stránek a aplikací. [cit. 2013-06-09]. Dostupné z: http://www.kosek.cz/vyuka/4iz228/prednasky/bezpecnost/frames.html

[Košař, 2011] KOŠAŘ, Stanislav. Vzdělávání a certifikace pracovníků v oblasti IT. Praha, 2011. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Králová, 2013] KRÁLOVÁ, Iveta. Zlepšování procesů testování v modelu TMMi. Praha, 2013. Seminární práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Kroll, 2007] KROLL, Per. IBM CORPORATION. Using RUP Effectively: How to drive success and avoid common pitfalls [Prezentace PowerPoint]. 2007 [cit. 2013-06-10]. Dostupné z: https://www.ibm.com/search/csass/search/?sn=dw&lang=en&cc=US&en=utf&hpp=20&dws=dw&q=using+rup+effectively&Search=Search

[Kučera, 2006] KUČERA, Martin. Testování softwaru a metodika RUP. Praha, 2006. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Kučera, 2009] KUČERA, Martin. Zvyšování zralosti testovacích procesů v IT společnosti. Praha, 2009. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[MMSP, 2011] Rejnková, Petra. MMSP. 2011. [cit. 2013-06-09]. Dostupné z: http://mmsp.czweb.org

[Myers, 2004] MYERS, Glenford J. The art of software testing. 2nd ed. Hoboken: Wiley, c2004, xv, 234 s. ISBN 978-0-471-46912-4.

[OpenUP, 2012] THE ECLIPSE FOUNDATION. OpenUP [online]. 2012 [cit. 2013-06-10]. 1.5.1.4, Version 7. Dostupné z: http://epf.eclipse.org/wikis/openup/

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

[Pospíšil, 2009] POSPÍŠIL, Marek. Eclipse Process Framework Composer - *nástroj na správu metodiky*. Praha : 2009. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Ramdeo, 2011] RAMDEO, Anand. Defect Report Template. TestingGeek [online]. 2011 [cit. 2013-06-10]. Dostupné z: http://www.testinggeek.com/defect-report-template

[Randová, 2007] RANDOVÁ, Libuše. Metodika RUP a testování. Praha, 2007. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Rational, 2011] RATIONAL SOFTWARE CORPORATION. Rational Unified Process: Best Practices for Software Development Teams [online]. 2011 [cit. 2013-06-10]. Dostupné z: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[Rejnková, 2011] REJNKOVÁ, Petra. Lokalizace a přizpůsobení metodiky OpenUP. Praha, 2011. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Ross, 2009] ROSS, Joel. E. Definition of Quality. Total Quality Management [online]. 2009 [cit. 2013-05-13]. Dostupné z: http://totalqualitymanagement.wordpress.com/2009/08/27/definition-of-quality/

[RUP, 2010] Rational Unified Process, version 7.5.1. IBM Corporation 2010.

[Singh, 2011] SINGH, Yogesh. Software testing. New York: Cambridge University Press, 2011, xxii, 626 s. ISBN 978-110-7012-967

[Software Testing Fundamentals, 2011] Defect Report. SOFTWARE TESTING FUNDAMENTALS. Software Testing Fundamentals [online]. 2011 [cit. 2013-06-10]. Dostupné z: http://softwaretestingfundamentals.com/defect-report

[Software Testing Help, 2008] How to get your all bugs resolved without any 'Invalid bug' label?. Software Testing Help [online]. 2008 [cit. 2013-06-10]. Dostupné z: http://www.softwaretestinghelp.com/how-to-get-your-all-bugs-resolved

[Software Testing Mentor] SOFTWARE TESTING MENTOR. What is Test Data. Software Testing Mentor [online]. [cit. 2013-06-10]. Dostupné z: http://www.softwaretestingmentor.com/test-deliverables/what-is-test-data

[SoftwareTestingStandard, 2012] SOFTWARETESTINGSTANDARD. ISO/IEC 29119 Software Testing: The new international software testing standard [online]. 2012 [cit. 2013-06-10]. Dostupné z: http://www.softwaretestingstandard.org/index.php

[Soundararajan, 2008] SOUNDARARAJAN, Pradeep. Automation replaces humans - The truth about what kind of humans it replaces. Tester Tested! [online]. 2008 [cit. 2013-06-10]. Dostupné z: http://testertested.blogspot.cz/2008_03_01_archive.html

[SQE, 2001a] SOFTWARE QUALITY ENGINEERING. Test Case Specification Template: (IEEE 829-1998) [online]. 2001 [cit. 2013-06-10]. Version 7.0. Dostupné z: http://www.ufjf.br/eduardo_barrere/files/2011/06/SQETestCaseSpecificationTemplate.pdf

[SQE, 2001b] SOFTWARE QUALITY ENGINEERING. Test Procedure Specification Template: (IEEE 829-1998) [online]. 2001 [cit. 2013-06-10]. Version 7.0. Dostupné z: http://www.ufjf.br/eduardo_barrere/files/2011/06/SQETestProcedureSpecificationTemplate.pdf

[SQE, 2001c] SOFTWARE QUALITY ENGINEERING. Test Log Template: (IEEE 829-1998) [online]. 2001 [cit. 2013-06-10]. Version 7.0. Dostupné z: http://www.ufjf.br/eduardo_barrere/files/2011/06/SQETestLogTemplate.pdf

[SQE, 2001d] SOFTWARE QUALITY ENGINEERING. Test Summary Report Template: (IEEE 829-1998) [online]. 2001 [cit. 2013-06-10]. Version 7.0. Dostupné z: http://www.ufjf.br/eduardo_barrere/files/2011/06/SQETestSummaryReportTemplate.pdf

[Systeme evolutif limited] SYSTEME EVOLUTIF LIMITED. Test plan outline: (IEEE 829 format) [online]. [cit. 2013-06-10]. Dostupné z: http://www.computing.dcu.ie/~davids/courses/CA267/ieee829mtp.pdf

[Šplíchalová, 2008] ŠPLÍCHALOVÁ, Marcela. Metodika testování webových aplikací. Praha, 2008. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

[Testování softwaru] Testovací tým. Testování softwaru [online]. [cit. 2013-06-10]. Dostupné z: http://testovanisoftwaru.cz/lang/en/manualni-testovani/testovaci-tym

[The Testing Standards Working Party, 2005] THE TESTING STANDARDS WORKING PARTY. The Testing Standards Working Party [online]. 2005 [cit. 2013-06-10]. Dostupné z: http://www.testingstandards.co.uk/

[TMMi Foundation, 2009] TMMi Foundation: TMMi Assessment Method Application Requirements (TAMAR) [online]. Ver. 2.0, 2009 [cit. 2013-05-13]. Dostupné z: http://www.tmmi.org/pdf/TMMi.TAMAR.pdf

[TMMi Foundation, 2012a] TMMi Foundation: Test Maturity Model Integration (TMMi) [online].Release. 1.0., 2012 [cit. 2013-05-13]. Dostupné z: http://www.tmmi.org/pdf/TMMi.Framework.pdf

[TMMi Foundation, 2012b] Purpose of the Foundation. TMMI FOUNDATION. TMMi Foundation [online]. 2012 [cit. 2013-05-13]. Dostupné z: http://www.tmmi.org/?q=about-us

[Veenendaal, 2013] VEENENDAAL, Erik van a Jan Jaap CANNEGIETER. Results of the first TMMi benchmark – where are we today?. Test Maturity Model integration (TMMi) [online]. 2013 [cit. 2013-05-13]. Dostupné z: http://www.tmmi.org/?q=papers

[Vomáčko, 2012] VOMÁČKO, Vít. Návrh metodiky zátěžového testování s nástrojem IBM Rational Performance tester. Praha, 2012. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.

# Seznam obrázků

# Seznam tabulek

# Příloha A: Hodnocení metodik OpenUP a MMSP podle stanovených kritérií

## A.1 Hodnocení metodiky OpenUP podle stanovených kritérií

Hodnocení bylo provedeno na základě zdrojů [OpenUP, 2012] a [Chodura, 2010].

### A.1.1 Kritérium Zaměření testování

Metodika OpenUP se zabývá závislým i nezávislým testováním. Závislé testování provádějí vývojáři. Za nezávislé testování jsou zodpovědní testeři. V předávací fázi jsou potom do testování zapojeni i budoucí uživatelé, kteří provádějí beta testy za účelem kontroly, zda vyvinutý software splňuje jejich očekávání.

### A.1.2 Kritérium Proces testování

Testování, stejně jako v metodice RUP, probíhá ve všech fázích životního cyklu projektu. Podstata testování je vystižena třemi praktikami, týkajícími se testování, tj. souběžné testování, vývoj řízený případy užití a vývoj řízený testy (test driven development). Testování probíhá během iterací souběžně s vývojem.

V zahajovací fázi je cílem testerů vytvořit testovací případy na základě případů užití.

První části kódu se začínají programovat v přípravné fázi. Vývojáři ještě před vlastním kódování implementují jednotkové testy, které následně po implementaci kódu spouští. Testeři vytváří nové testovací případy a souběžně s vývojáři implementují testovací skripty a následně je spouští. Tento postup je opakován v několika iteracích. Zároveň mohou nastat změny v požadavcích zákazníka, na které je třeba reagovat. Pro vývojáře i testery to z hlediska testování znamená úpravu naimplementovaných testů dle nových požadavků a následné ověření, zda jsou požadavky splněny.

V konstrukční fázi je systém neustále implementován, testován a integrován a z pohledu testování tato fáze vypadá obdobně jako fáze přípravná. S každou iterací implementují testeři i vývojáři nové testy, které následně spouští. Zároveň se očekává více změn požadavků ze strany zákazníka a s tím související potřebu úpravy testů a jejich opětovné spouštění. Na konci fáze konstrukce je vytvořena beta verze produktu, která je následně předána zákazníkovi.

V předávací fázi je cílem reagovat na zpětnou vazbu zákazníka, který prověřuje vydanou beta verzi. Nevytváří se již nové testovací případy, ale stejně jako v ostatních fázích je třeba reagovat na požadavky zákazníka a upravovat testy podle posledních změn a znovu je

spouštět. Nakonec proběhne regresní testování, kdy je produkt otestován všemi skripty a následně je předán zákazníkovi.

### A.1.3 Kritérium Role

Na testování se v metodice OpenUP podílejí role vývojář a tester. Vývojář zodpovídá z hlediska testování za implementaci jednotkových testů a jejich následné spouštění. Tester zodpovídá za identifikaci testů spolu s identifikací vhodného přístupu k testování, implementaci testů, provádění testů spolu s logováním výsledků a ověřováním, že testy byly provedeny a komunikaci výsledků testů zbytku týmu. V OpenUP je definována pouze role tester, která není dále rozdělena na specializované role, jako v metodice RUP. Testeři i vývojáři mají společného nadřízeného, kterým je projektový manažer. Zákazník je sice do testování zapojen v předávací fází, ale nenese za testování zodpovědnost. Má ale možnost ověřit, jak beta verze produktu naplňuje jeho očekávání a vznést případné připomínky.

### A.1.4 Kritérium Úrovně testování

Metodika rozlišuje testy vytvořené podle případů užití, za které je zodpovědný tester a unit testy, které jsou v odpovědnosti vývojáře.

Unit testy jsou zde chápány jako testy založené na očekávaném chování jednotek kódu. Může se jednat o testy operací, polí přidaných do uživatelského rozhraní, tříd při objektovém programování, apod. Jejich účelem je ověřit nové přírůstky v kódu, takže je třeba je vnímat jednak jako testy samotné jednotky kódu a jednak i jako testy interakcí jednotky s jejím okolím. Podle úrovní testování plní unit testy v metodice OpenUP roli jednotkových i integračních testů.

Testy vytvořené podle případů užití jsou zaměřeny na testování funkcionalit a slouží k ověření splnění požadavků uživatelů. Na rozdíl od vývojových testů nejsou zaměřeny na testování kódu, ale na ověřování funkcionalit skrze zadávání vstupů a vyhodnocování výstupů. Z hlediska úrovní testování plní testy dle případů užití roli systémových testů.

Spouštění testů by mělo probíhat co nejčastěji. Ideálně by všechny testy měly být spuštěny proti každému sestavení (buildu), nasazeném na testovací prostředí. Pokud to ovšem není možné, je vhodné se zaměřit především na novou funkcionalitu a následně regresními testy ověřit dosavadní funkcionalitu.

Během předávací fáze přichází s vytvořeným softwarem do styku zákazník a provádí beta testování, což koresponduje s úrovní akceptačních testů. Nicméně role zákazníka je v metodice zahrnuta pod obecnou rolí zainteresovaná strana *(stakeholder)*, která má podpůrný charakter a není vyloženě zodpovědná za žádnou z úloh. Spouštění testů je zde striktně v zodpovědnosti testera, resp. vývojáře u vývojových testů. U této metodiky tedy nelze hovořit o pokrytí úrovně akceptačních testů.

### A.1.5 Kritérium Trasovatelnost

V metodice OpenUP jsou testovací případy odvozeny od požadavků zachycených v podobě případů užití nebo charakteristik výkonu a spolehlivosti. Vazby mezi požadavky a testy vznikají v průběhu vytváření testovacích případů, ale metodika nevyžaduje jejich dokumentaci. Trasovatelnost tedy je zajištěna, ale není dokumentována.

### A.1.6 Kritérium Životní cyklus vady

Životní cyklus vady metodika konkrétně nepopisuje. Nalezené vady jsou ale přidávány na seznam pracovních položek spolu s ostatními úkoly, které mají být v rámci projektu řešeny (případy užití, systémové požadavky, požadavky na změnu nebo vylepšení, úlohy pro vývoj). V rámci tohoto dokumentu mohou být vady prioritizovány, může být odhadována jejich pracnost a může zde být zaznamenáván stav jejich řešení, což může fungovat obdobně jak životní cyklus vady.

Vzhledem k tomu, že testování probíhá neustále a vady jsou objeveny prakticky ihned po nasazení nového sestavení (buildu), spousta vad je řešena rovnou s vývojáři a nemusí být zaznamenávány do seznamu pracovních položek. Lokalizace a odstranění zjištěné vady při soustavném testování je snazší, jelikož se zpravidla jedná o vadu v posledním přírůstku.

### A.1.7 Kritérium Typy testů a testovací techniky

V metodice je zahrnut koncept testování požadavků na kvalitu, který rozlišuje typy testů podle metody FURPS[44].

Testy funkcionalit rozlišuje na testy funkcí *(function tests)*, bezpečnostní testy *(security tests)* a objemové testy *(volume tests)*. Testy použitelnosti rozlišuje na testy použitelnosti *(usability tests)*, testy integrit *(integrity tests)* a testy struktury *(structure tests)*. Testy spolehlivosti dělí na stress testy *(stress tests)*, benchmarkové testy *(benchmark tests)* a testy konfliktů *(contention tests)*. Do výkonnostních testů zařazuje zátěžové testy *(load tests)*, testy výkonnostního profilu *(perfomance profile)* a konfigurační testy *(configuration tests)*. Podtypem testů podpory jsou zde testy instalací *(installation tests)*.

Metodika nespecifikuje smoke testy, beta testy, průzkumné testování, ani jiné techniky. Jsou zde pouze návody na programování a údržbu automatizovaných testů.

### A.1.8 Kritérium Artefakty testování

Metodika OpenUP definuje artefakty Testovací případ *(Test Case)*, Test log *(Test Log)* a Testovací skript *(Test Script)*. Test Case odpovídá dokumentu Level Test Case dle IEEE 829, Test log je ekvivalentní s dokumentem Level Test Log a Testovací skript je obdoba do-

[44] Metoda **FURPS** je vysvětlena v kapitole 6. 2. 1. 3.

kumentu Level Test Procedure podle IEEE 829. Pro hlášení nalezených vad se používá Seznam pracovních položek, který slouží obecně pro záznam prací, které musí být provedeny a nikoliv výhradně pro řešení vad. Tento artefakt tedy neodpovídá dokumentu Anomaly Report dle IEEE 829. V disciplíně Vývoj je potom specifikován artefakt Vývojový test *(Developer Test)*, který není součástí standardu IEEE 829. Pro souhrn výsledků provádění testů metodika nepředepisuje žádný specifický artefakt, ale jedním z kroků provádění testů je poskytnutí zpětné vazby týmu, zda dané sestavení uspokojuje požadavky plánované pro danou iteraci. Forma, v jaké bude zpětná vazba předána, záleží na členech týmu.

### A.1.9 Kritérium Užití nástrojů a automatizace

Jednou z nejlepších praktik metodiky OpenUP je testování souběžné s vývojem během jednotlivých iterací. Neustálý vývoj nových přírůstků vyžaduje neustálé regresní testování a regresní testování vyžaduje vysoký stupeň automatizace testů. Metodika OpenUP proto doporučuje automatizovat testy na všech úrovních. Metodika zahrnuje návod pro údržbu automatizovaných testovacích sad, ve kterém jsou uvedeny tipy pro usnadnění údržby testovacích sad. Kromě tohoto návodu je v metodice obsažen také návod pro programování automatizovaných testů, ve které jsou diskutovány způsoby strukturování automatizovaných testů, nahrávání testů, zadávání testovacích dat, spouštění testů a vypořádání se s nalezenými vadami.

Metodika není integrována s žádným nástrojem pro automatizaci testů. Pro podporu automatizace testování je možné použít jak open-source nástroje, tak i komerční nástroje. Jediným nástrojem svázaným s metodikou je nástroj pro vlastní správu metodiky Eclipse Process Framework Composer. V metodice je uveden stručný popis tohoto nástroje a návod pro úpravu metodiky v nástroji.

## A.2 Hodnocení metodiky MMSP podle stanovených kritérií

Hodnocení bylo provedeno na základě zdrojů [MMSP, 2011] a [Rejnková, 2011].

### A.2.1 Kritérium Zaměření testování

Metodika MMSP se stejně jako OpenUP zabývá závislým i nezávislým testováním. Závislé testování provádějí vývojáři a nezávislé testování mají na starost testeři. V předávací fázi je budoucím uživatelům k dispozici beta verze produktu. Ti nejsou přímo zodpovědní za testování, ale mají možnost beta verzi prověřit, zda splňuje jejich očekávání, a případně nahlásit vývojovému týmu nalezené vady.

### A.2.2 Kritérium Proces testování

Testování je zastoupeno ve všech fázích životního cyklu projektu.

V zahajovací fázi, jejímž cílem je úspěšně projekt odstartovat, proběhne pouze plánování testů. Dílčími kroky plánování testů je definování strategie testování a plánování provedení testů. Metodika uvádí, že plánování testů by mělo proběhnout vždy před začátkem nové iterace vývoje.

Ve fázi rozpracování je cílem porozumět požadavkům na software, navrhnout architekturu řešení, přijmout opatření pro zmírnění dopadů a prevenci rizik a upřesnit plán projektu. Již jsou k dispozici první přírůstky softwaru, které je nutné otestovat. V jednotlivých iteracích fáze rozpracování jsou jednak vývojáři vytvářeny a prováděny unit testy a jednak už probíhá příprava a provádění testů testery. Na rozdíl od OpenUP metodika nezmiňuje vývoj řízený testy, takže není vyžadována implementace unit testů před implementací kódu. Nicméně si myslím, že implementovat unit testy před vlastním kódem by mělo být zvyklostí vývojářů, i když to metodika nevyžaduje. Dílčími kroky tvorby unit testů je výběr testované části implementace a příprava unit testů. Dílčími kroky provedení unit testů je spuštění unit testů a zhodnocení jejich výsledků. Příprava testů, která je v zodpovědnosti testerů, se skládá z kroků mapování testovacích nápadů, tvorby testovacích případů, tvorby testovacích sad a definice testovacích dat. Při vlastním provedení testů proběhne výběr testovacích sad, realizace testů a zhodnocení výsledků testů.

V konstrukční fázi je cílem vyvinout funkční verzi systému, která je připravena na akceptaci. Z pohledu testování tato fáze vypadá obdobně jako fáze přípravná.

Ve fázi zavedení jsou opraveny všechny vady nalezené v beta verzi a vyvíjený výsledný produkt je uveden do provozu. Úlohy testování jsou stejné jako v předchozí fázi. Příprava testování probíhá, jen pokud je nutné připravit nové testy. V průběhu celého projektu musí proces testování, stejně jako v metodice OpenUP, pružně reagovat na změny požadavků.

### A.2.3 Kritérium Role

Role podílející se na testování jsou vývojář a tester. Popis kritéria Role je u metodiky MMSP shodný jako u metodiky OpenUP.

### A.2.4 Kritérium Úrovně testování

V rámci plánování testů metodika uvádí, že je vhodné definovat jednotlivé úrovně testů. Unit testy jsou prováděny vývojáři a jsou zde specifikovány jako ověření funkčnosti malých, soběstačných částí zdrojového kódu implementace jako jsou testy jednotlivých tříd a jejich metod. V Plánu testů metodika doporučuje specifikovat také testy integrační, které ověřují spolupráci částí programového kódu pomocí rozhraní. Zodpovědnost za tuto úroveň testů ale přímo neuvádí. Vzhledem k tomu, že se jedná o testy programového kódu, měli by za ně být nejspíš zodpovědní také vývojáři v rámci úloh tvorba a provedení unit testů. Úroveň systémových testů v metodice odpovídá úlohám příprava a provedení testů rolí tester. Akceptační tes-

ty pak probíhají v rámci fáze zavádění ve spolupráci se zákazníky stejně jako u metodiky OpenUP.

### A.2.5 Kritérium Trasovatelnost

Při přípravě testů nejprve probíhá mapování testovacích nápadů, které vycházejí z nashromážděných požadavků na systém a případů užití. Tvorba testovacích případů navazuje na testovací nápady i specifikaci požadavků, tudíž trasovatelnost je v metodice zajištěna. Zachycení vazeb mezi požadavky a testy v dokumentaci zmiňováno není a ani v navržených šablonách pracovních produktů nejsou definovány položky, do kterých by se vazby zapisovaly.

### A.2.6 Kritérium Životní cyklus vady

Životní cyklus vady metodika nepopisuje. Z metodiky není příliš jasné, jak se při nalezení vady postupuje.

Na jednu stranu uvádí, že v okamžiku, kdy je při provádění testů vada nalezena, *„je nutné ji reportovat, resp. zaznamenat výsledky testů do Testovacích sad a Reportu chyb (součást pracovního produktu Záznam výsledků testů)“* [Rejnková, 2011, s. 67].

Na druhé straně uvádí, že Záznam výsledků testů je vhodné vypracovat až *„po dokončení veškerých testů v dané iteraci, popřípadě pouze po významnějších iteracích“*. V rámci úlohy zhodnocení výsledků testů jsou potom veškeré vytvořené záznamy z průběhu testování zpřístupněny všem členům týmu a souhrnné výsledky testování by měly být konzultovány na pravidelných setkáních týmu. Zároveň je v úloze zhodnocení výsledků testů uvedeno, že každá vada *„by měla být komunikována s příslušným vývojářem a požadavky na její nápravu by měly být zaznamenány v Seznamu pracovních položek nebo Seznamu požadavků na změnu.“* [Rejnková, 2011, s. 68]

Dle mého názoru by v praxi vypořádání s vadami podle této metodiky mohlo probíhat následovně: Záznam výsledků testů, ve kterém jsou zanořeny i reporty vad, by vývojářům byl přístupný po celou dobu provádění testů (nikoli až po provedení testů), přičemž průběžně by nálezy vad byly vývojářům oznamovány a požadavky na opravu vady zadávány do Seznamu pracovní položek. Vývojáři by vady průběžně opravovali a na konci iterace by testeři do Záznamu výsledků testů doplnili souhrnné informace. Životní cyklus vady by tak byl mapován v Seznamu pracovních položek, obdobně jako v metodice OpenUP.

### A.2.7 Kritérium typy testů a testovací techniky

Metodika MMSP uvádí, že v rámci plánování testů by měly být určeny testovací techniky a typy testů, včetně určení zodpovědností. Typy testů, stejně jako metodika OpenUP,

rozlišuje podle metody FURPS na testy funkcionality, použitelnosti, spolehlivosti, výkonnosti a podpory. Konkrétní techniky testování specifikovány nejsou.

### A.2.8 Kritérium Artefakty testování

MMSP zavádí artefakty Plán testů, Seznam testovacích nápadů, Testovací data, Testovací případ, Testovací sada a Záznam výsledků testů. Plán testů by měl zahrnovat veškeré informace o testování jak napříč úrovněmi testování, tak pro jednotlivé úrovně. Svou povahou tedy odpovídá dokumentům Master Test Plan a Level Test Plan dle IEEE 829.

Seznam testovacích nápadů není vyžadován standardem IEEE 829. V MMSP se jedná o artefakt vytvářený rámci přípravy testů, který zachycuje oblasti testování.

Testovací data také nejsou specifikována standardem IEEE 829. V metodice se jedná o množinu dat, která se využívána při testování.

Testovací případ v MMSP přibližně odpovídá dokumentu sloučení dokumentů Level Test Case, Level Level Test Procedure i Level Test Log standardu IEEE 829. Jedná se o specifikace jednotlivých kroků testera a reakcí systému, přičemž jsou zároveň specifikovány vstupní podmínky. Při provádění testů jsou potom k jednotlivým krokům testovacích případů zaznamenávány výsledky.

Testovací sada není specifikována standardem IEEE 829. V MMSP jde o uspořádání jednotlivých testovacích případů do logické posloupnosti.

Záznam výsledků testů je sjednocením informací o provedených testech a jejich výsledcích, včetně podrobných Reportů vad. Ve standardu IEEE 829 se jedná o 3 separátní dokumenty, které má význam oddělovat tj. Anomaly Report, Level Interim Test Status Report, Level Test Report (příp. Master Test Report).

V disciplíně Vývoj je potom specifikován artefakt Unit test, který není součástí standardu IEEE 829. Pro zadávání požadavků na opravy vad se v MMSP, stejně jako v OpenUP, používá Seznam pracovních položek.

### A.2.9 Kritérium Užití nástrojů a automatizace

MMSP uvádí, že testování může být manuální i automatizované, shrnuje hlavní důvody pro rozhodování o automatizaci testování, ale hlavní pozornost je věnována manuálnímu testování.

Automatizovány jsou zpravidla vývojové testy, při jejichž přípravě a spouštění jsou využívány specializované frameworky, označované jako xUnit. Konkrétně jsou uvedeny např. frameworky JUnit pro Javu a PHPUnit pro PHP, ale volba daného frameworku závisí na použitém programovacím jazyku. Při přípravě testovacích případů v úloze příprava testů na úrov-

ni systémových testů jsou doporučeny automatizované jako je např. IBM Rational Functional Tester, HP QuickTest Professional, nebo Borland SilkTest.

S konkrétním nástrojem pro automatizaci testů metodika provázána není. Spravovat metodiku je možné v nástroji EPF Composer. Zdrojové soubory metodiky pro zájemce o úpravu nebo rozšíření metodiky uveřejněny nejsou, ale jsou k dispozici na vyžádání na Katedře informačních technologií na VŠE v Praze.

# Příloha B: IBM Rational Quality Manager[45]

Nástroj IBM Rational Quality Manager je komerčním nástrojem od společnosti IBM. Je jednou z aplikací sady produktů IBM CLM (Collaborative Lifecyle Management), které jsou dodávány společně se serverem Jazz Team Server. Ideou Jazz Team Serveru[46] je integrace informací a úkolů napříč jednotlivými fázemi životního cyklu za účelem zvýšení spolupráce, produktivity a transparence při vývoji softwaru. Součástí Jazz Team serveru je platforma, produkty a komunita. [IBM, 2013] Na platformě Jazz jsou vzájemně integrovány produkty

- IBM Rational Requirements Composer,
- IBM Rational Team Concert,
- IBM Rational Quality Manager
- IBM Design Management.

IBM Rational Quality Manager má charakter webové aplikace a nabízí rozsáhlé funkce pro plánování a konstruování testů a správu pracovních produktů testování v rámci celého životního cyklu vývoje softwaru. Kompletně nahrazuje dřívější produkty IBM Rational Manual Tester, IBM Rational ClearQuest TestManager a IBM Rational TestManager. Je určen pro testovací týmy všech velikostí a podporuje různé role z oblasti zajišťování kvality softwaru, jako jsou test manažeři, správci testů a testovacího pracoviště a členové týmu, co navrhují a provádějí testy.

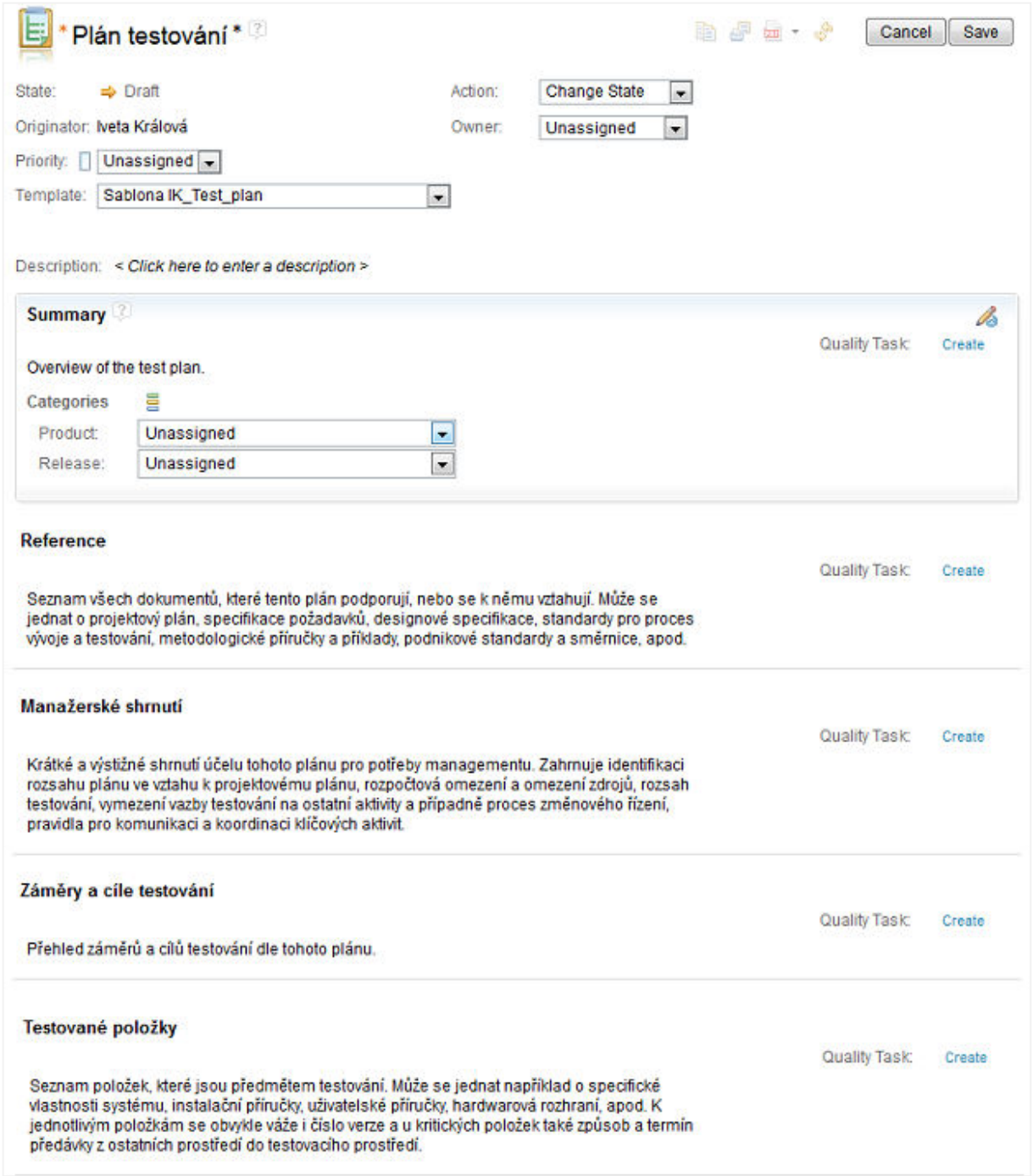
Mezi hlavní funkce nástroje se řadí dynamické plány testování, řízení pracovních úkolů, správa testovacího pracoviště, návrh testovacích případů, možnost vytváření manuálních i automatizovaných testovacích skriptů (a to včetně propojení na nástroje pro automatizaci testů IBM Rational Functional Tester, IBM Rational Performance Tester, Rational Robot, Rational Service Tester for SOA Quality a IBM Rational AppScan Tester Edition), analýza pokrytí testů, sledování metrik pro testování a řada dalších funkcí. Tyto funkce je možné integrovat s jinými pracovními produkty životního cyklu, jako jsou například pracovní položky a požadavky, a s tvorbou sestav a panelů dashboard, které poskytují informace o stavu a průběhu projektu.

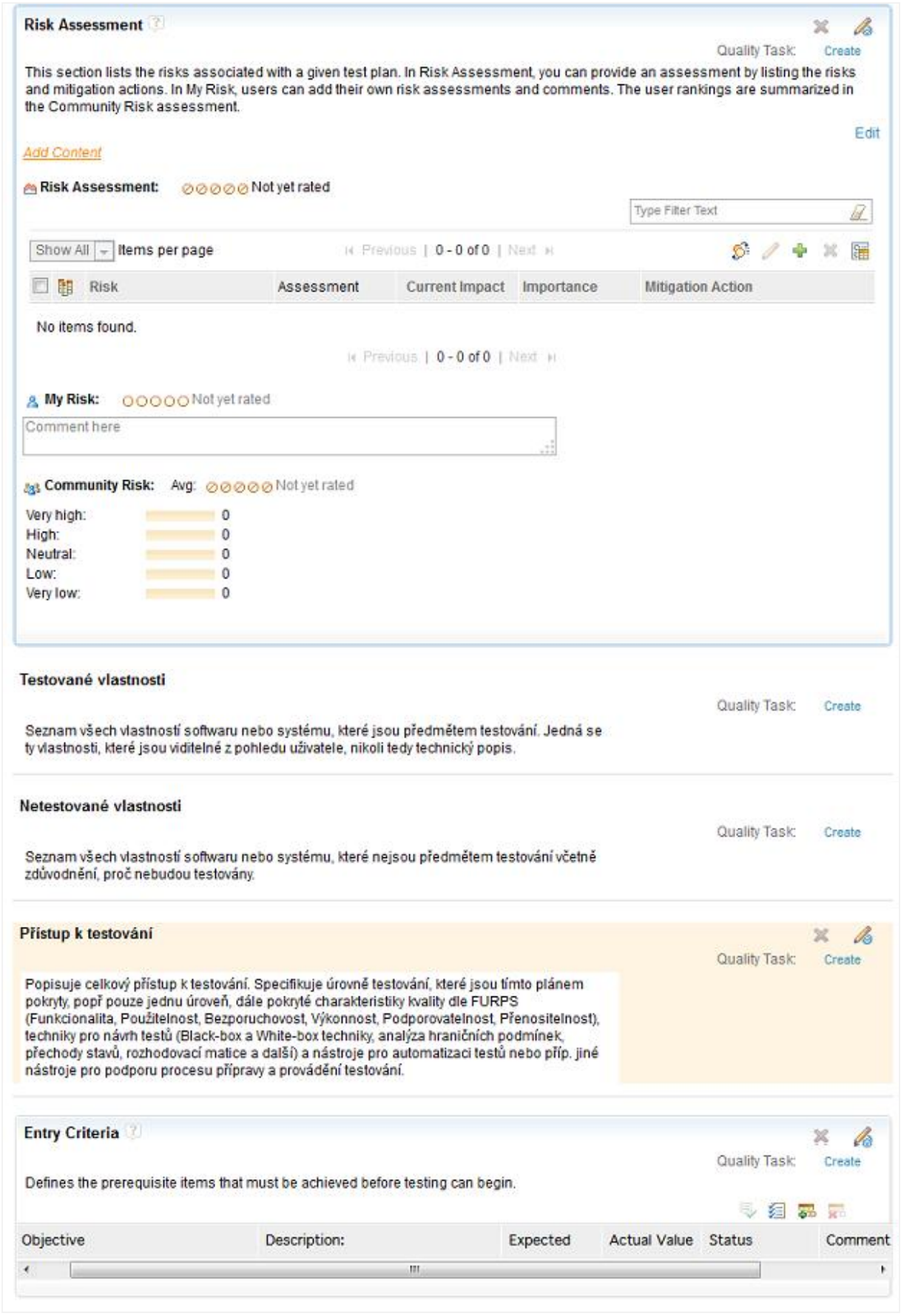
[45] Text této přílohy vychází ze zdroje [IBM, 2012]

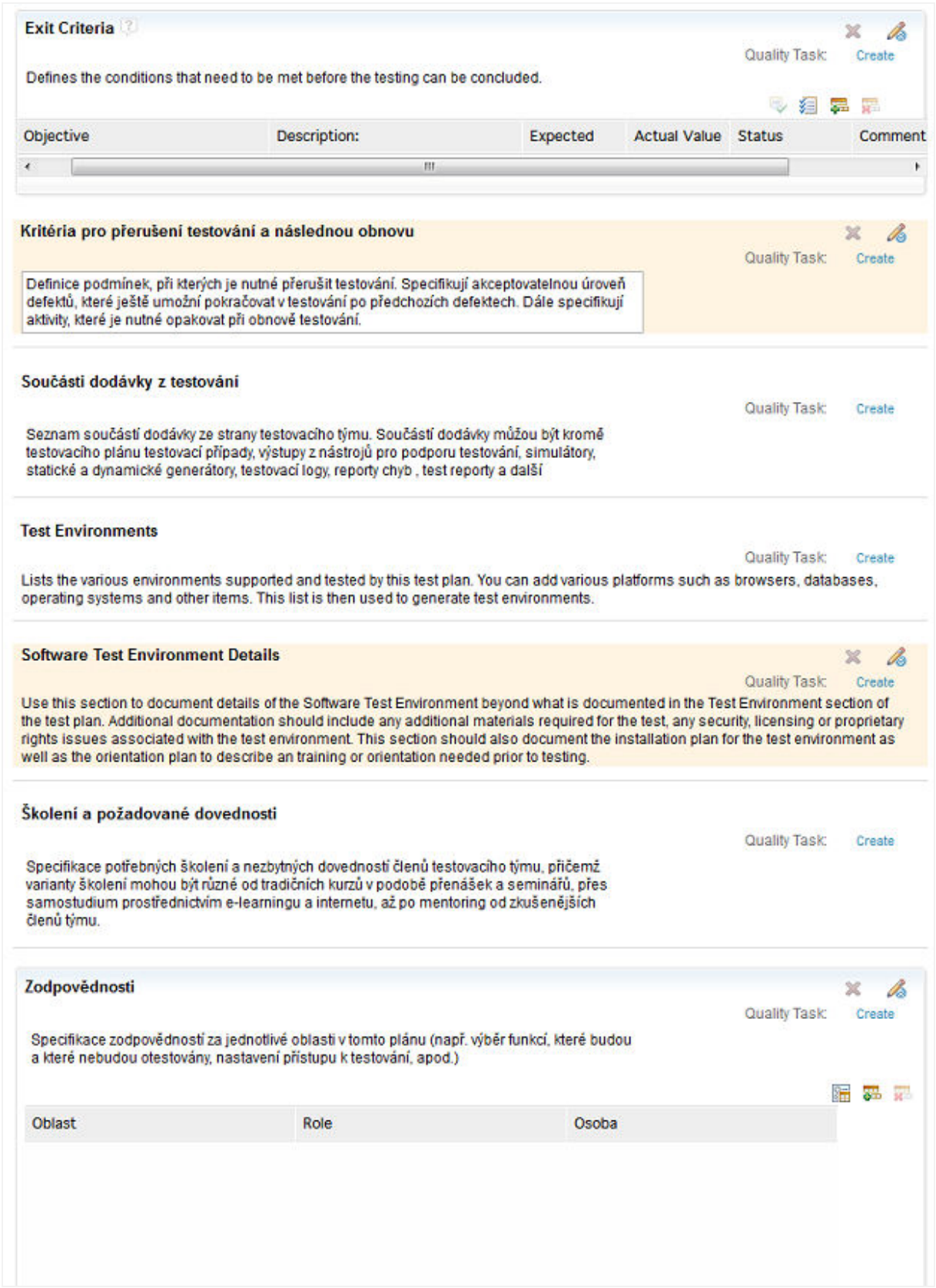
[46] Více o Jazz Team Serveru a platformě Jazz je možné najít v [IBM, 2013].

# Příloha C: Šablony pracovních produktů

Ukázkovou šablonu pracovního produktu **plán testování** v nástroji IBM Rational Quality Manager zachycuje Obrázek č. 18.




Risk Assessment
Quality Task: Create
This section lists the risks associated with a given test plan. In Risk Assessment, you can provide an assessment by listing the risks and mitigation actions. In My Risk, users can add their own risk assessments and comments. The user rankings are summarized in the Community Risk assessment.
Edit
Add Content
Risk Assessment: Not yet rated
Type Filter Text
Show All Items per page
Previous | 0 - 0 of 0 | Next
Risk
Assessment
Current Impact
Importance
Mitigation Action
No items found.
Previous | 0 - 0 of 0 | Next
My Risk: Not yet rated
Comment here
Community Risk: Avg: Not yet rated
Very high: 0
High: 0
Neutral: 0
Low: 0
Very low: 0
Testované vlastnosti
Quality Task: Create
Seznam všech vlastností softwaru nebo systému, které jsou předmětem testování. Jedná se ty vlastnosti, které jsou viditelné z pohledu uživatele, nikoli tedy technický popis.
Netestované vlastnosti
Quality Task: Create
Seznam všech vlastností softwaru nebo systému, které nejsou předmětem testování včetně zdůvodnění, proč nebudou testovány.
Přístup k testování
Quality Task: Create
Popisuje celkový přístup k testování. Specifikuje úrovně testování, které jsou tímto plánem pokryty, popř pouze jednu úroveň, dále pokryté charakteristiky kvality dle FURPS (Funkcionalita, Použitelnost, Bezporuchovost, Výkonnost, Podporovatelnost, Přenositelnost), techniky pro návrh testů (Black-box a White-box techniky, analýza hraničních podmínek, přechody stavů, rozhodovací matice a další) a nástroje pro automatizaci testů nebo příp. jiné nástroje pro podporu procesu přípravy a provádění testování.
Entry Criteria
Quality Task: Create
Defines the prerequisite items that must be achieved before testing can begin.
Objective
Description:
Expected
Actual Value
Status
Comment



**Exit Criteria**

Quality Task: Create

Defines the conditions that need to be met before the testing can be concluded.

| Objective | Description: | Expected | Actual Value | Status | Comment |
|---|---|---|---|---|---|

**Kritéria pro přerušení testování a následnou obnovu**

Quality Task: Create

Definice podmínek, při kterých je nutné přerušit testování. Specifikují akceptovatelnou úroveň defektů, které ještě umožní pokračovat v testování po předchozích defektech. Dále specifikují aktivity, které je nutné opakovat při obnově testování.

**Součásti dodávky z testování**

Quality Task: Create

Seznam součástí dodávky ze strany testovacího týmu. Součástí dodávky můžou být kromě testovacího plánu testovací případy, výstupy z nástrojů pro podporu testování, simulátory, statické a dynamické generátory, testovací logy, reporty chyb , test reporty a další

**Test Environments**

Quality Task: Create

Lists the various environments supported and tested by this test plan. You can add various platforms such as browsers, databases, operating systems and other items. This list is then used to generate test environments.

**Software Test Environment Details**

Quality Task: Create

Use this section to document details of the Software Test Environment beyond what is documented in the Test Environment section of the test plan. Additional documentation should include any additional materials required for the test, any security, licensing or proprietary rights issues associated with the test environment. This section should also document the installation plan for the test environment as well as the orientation plan to describe an training or orientation needed prior to testing.

**Školení a požadované dovednosti**

Quality Task: Create

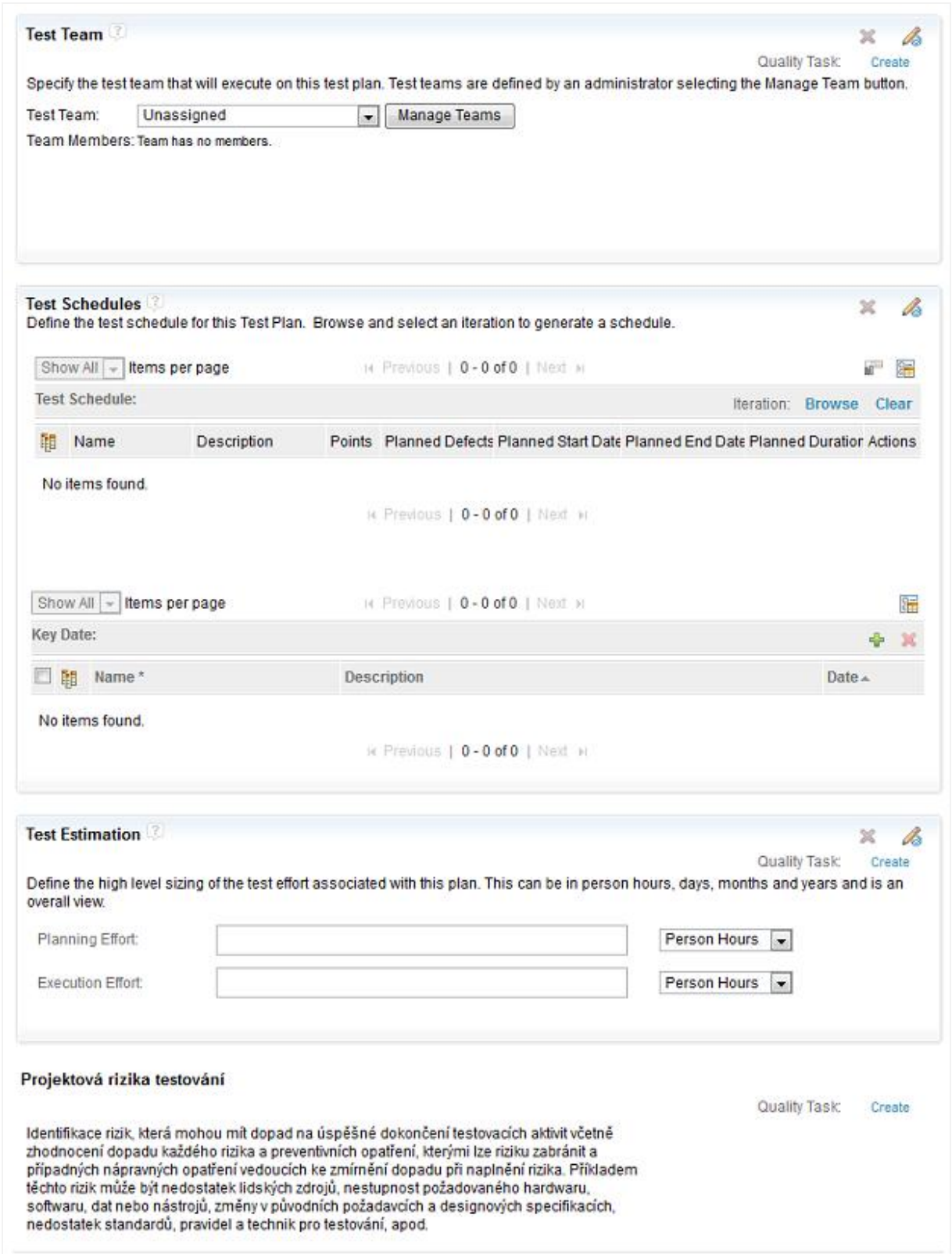
Specifikace potřebných školení a nezbytných dovedností členů testovacího týmu, přičemž varianty školení mohou být různé od tradičních kurzů v podobě přenášek a seminářů, přes samostudium prostřednictvím e-learningu a internetu, až po mentoring od zkušenějších členů týmu.

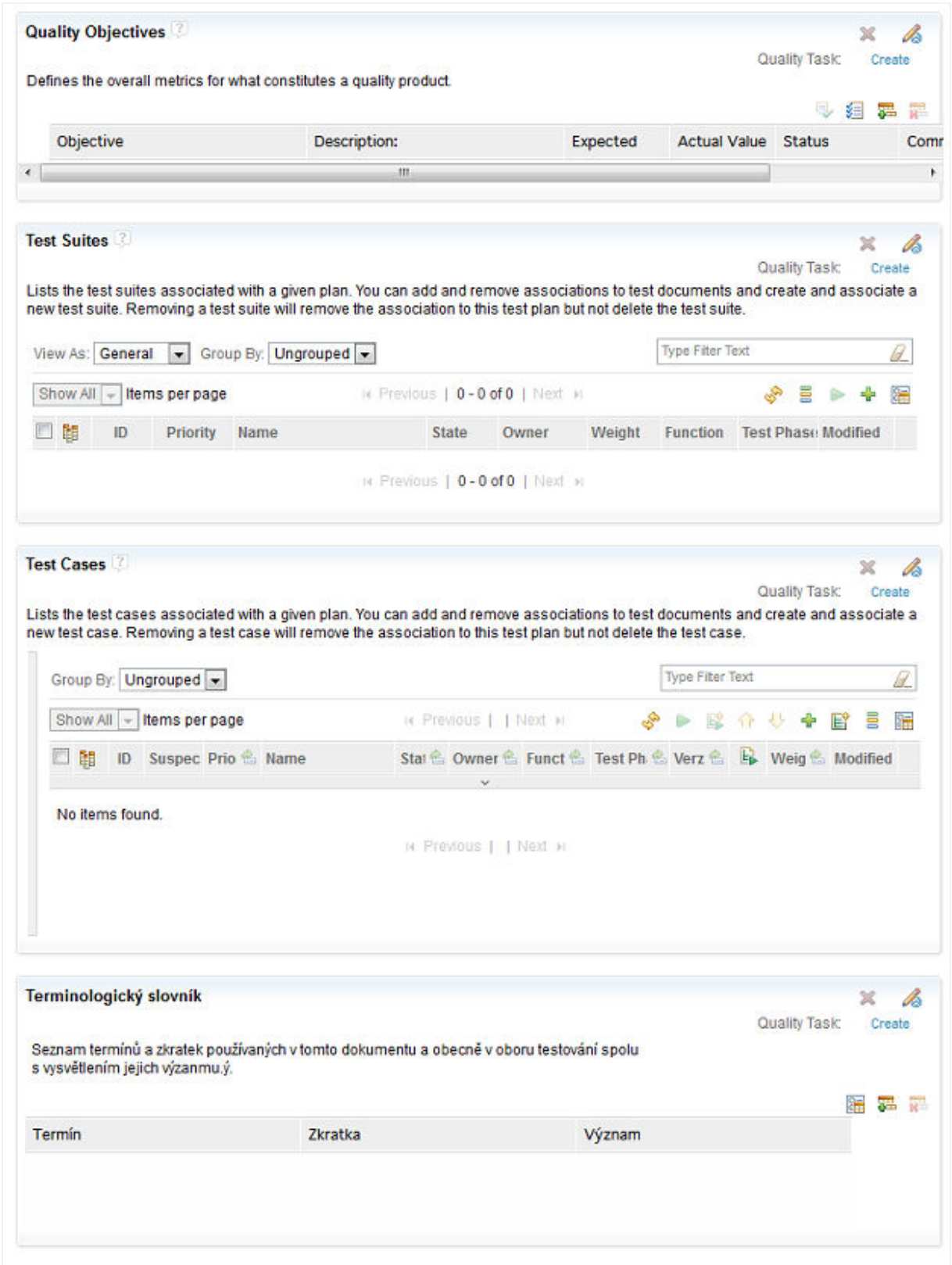
**Zodpovědnosti**

Quality Task: Create

Specifikace zodpovědností za jednotlivé oblasti v tomto plánu (např. výběr funkcí, které budou a které nebudou otestovány, nastavení přístupu k testování, apod.)

| Oblast | Role | Osoba |
|---|---|---|


Test Team
Quality Task: Create
Specify the test team that will execute on this test plan. Test teams are defined by an administrator selecting the Manage Team button.
Test Team: Unassigned Manage Teams
Team Members: Team has no members.
Test Schedules
Define the test schedule for this Test Plan. Browse and select an iteration to generate a schedule.
Show All Items per page Previous | 0 - 0 of 0 | Next
Test Schedule: Iteration: Browse Clear
Name Description Points Planned Defects Planned Start Date Planned End Date Planned Duration Actions
No items found.
Previous | 0 - 0 of 0 | Next
Show All Items per page Previous | 0 - 0 of 0 | Next
Key Date:
Name * Description Date
No items found.
Previous | 0 - 0 of 0 | Next
Test Estimation
Quality Task: Create
Define the high level sizing of the test effort associated with this plan. This can be in person hours, days, months and years and is an overall view.
Planning Effort: Person Hours
Execution Effort: Person Hours
Projektová rizika testování
Quality Task: Create
Identifikace rizik, která mohou mít dopad na úspěšné dokončení testovacích aktivit včetně zhodnocení dopadu každého rizika a preventivních opatření, kterými lze riziku zabránit a případných nápravných opatření vedoucích ke zmírnění dopadu při naplnění rizika. Příkladem těchto rizik může být nedostatek lidských zdrojů, nestupnost požadovaného hardwaru, softwaru, dat nebo nástrojů, změny v původních požadavcích a designových specifikacích, nedostatek standardů, pravidel a technik pro testování, apod.


Quality Objectives
Quality Task:
Create
Defines the overall metrics for what constitutes a quality product.
Objective
Description:
Expected
Actual Value
Status
Comr
Test Suites
Quality Task:
Create
Lists the test suites associated with a given plan. You can add and remove associations to test documents and create and associate a new test suite. Removing a test suite will remove the association to this test plan but not delete the test suite.
View As: General
Group By: Ungrouped
Type Filter Text
Show All
Items per page
Previous | 0 - 0 of 0 | Next
ID
Priority
Name
State
Owner
Weight
Function
Test Phase
Modified
Previous | 0 - 0 of 0 | Next
Test Cases
Quality Task:
Create
Lists the test cases associated with a given plan. You can add and remove associations to test documents and create and associate a new test case. Removing a test case will remove the association to this test plan but not delete the test case.
Group By: Ungrouped
Type Filter Text
Show All
Items per page
Previous | | Next
ID
Suspec
Prio
Name
Sta
Owner
Funct
Test Ph
Verz
Weig
Modified
No items found.
Previous | | Next
Terminologický slovník
Quality Task:
Create
Seznam termínů a zkratek používaných v tomto dokumentu a obecně v oboru testování spolu s vysvětlením jejich význanmu.ý.
Termín
Zkratka
Význam


**Obrázek č. 18 Šablona pro plán testování vytvořená v nástroji IBM Rational Quality Manager, zdroj: autor podle [IEEE, 2008, s. 42-50; Systeme Evolutif Limited]**

Šablona pro **seznam pracovních podmínek** vytvořená v MS Word a připravená k použití je umístěna na následujících dvou stránkách. Jedná se o doplnění šablony, kterou definuje [MMSP, 2011].

# Seznam testovacích podmínek

# 1. Úvod

## 1. 1. Účel

[Jaký je účel tohoto dokumentu?]

## 1. 2. Rozsah

[Jaký je rozsah tohoto dokumentu? Je možné z něj vycházet pouze v rámci tohoto projektu?]

## 1. 3. Definice a zkratky

[Existují nějaké pojmy a zkratky, které by mohly být pro srozumitelnost tohoto dokumentu vysvětleny?]

## 1. 4. Odkazy

[Vychází tento dokument z nějakých jiných výstupů, popř. odkazuje na nějaké další dokumenty?]

## 1. 5. Historie verzí

[Kdo, kdy a jak tento dokument upravoval?]

| Datum | Verze | Popis | Autor |
|---|---|---|---|
| | | | |
| | | | |

<Název projektu>

Seznam testovacích podmínek Datum: <dd/mm/rrrr>

# 2. Testovaná položka

[Specifikace položky (vč. verze), která je předmětem testování ve vazbě na plán testování (specifické vlastnosti systému, instalační příručky, uživatelské příručky, hardwarová rozhraní, apod.)?]

## 2. 1. Testovací podmínky

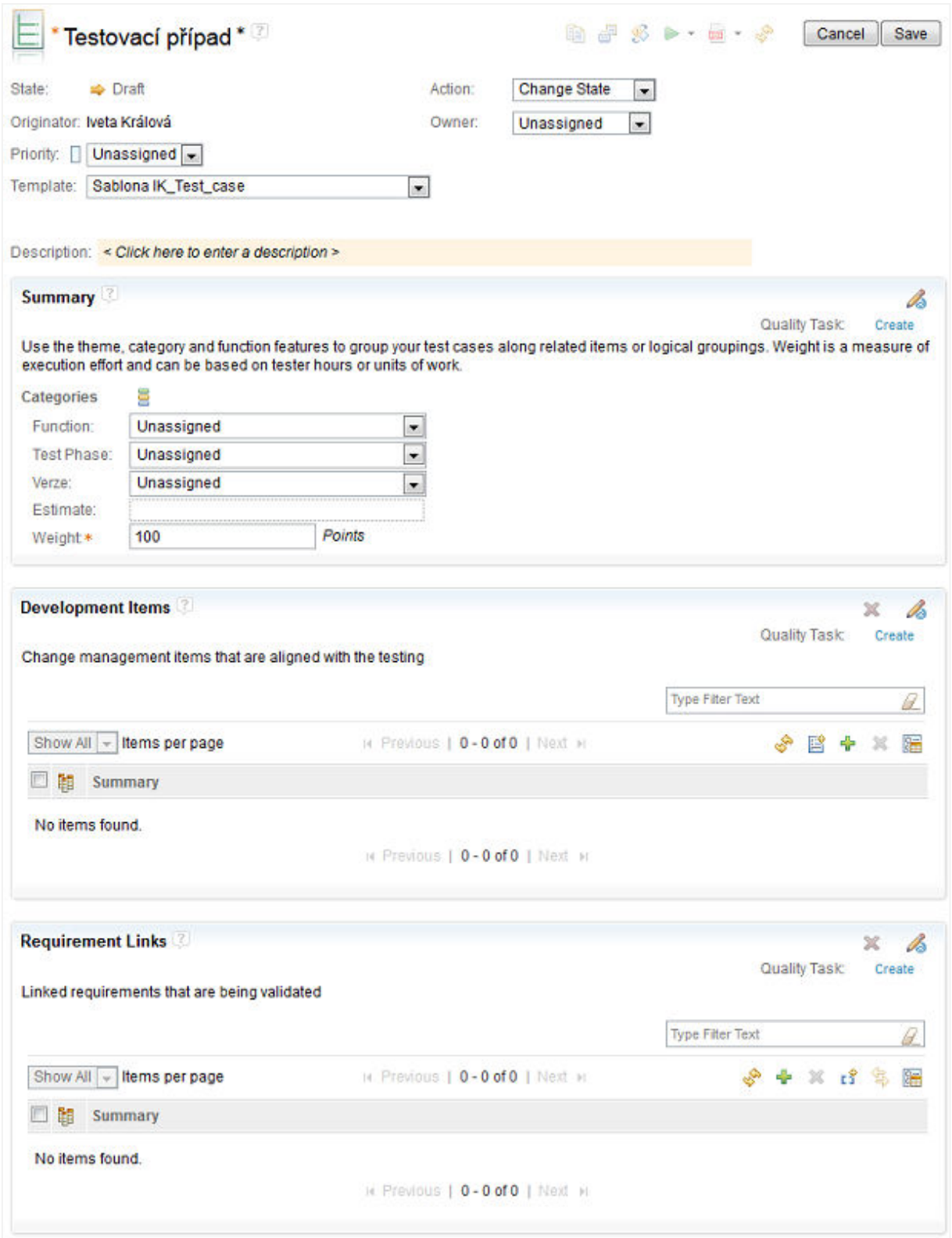
[Tabulka zpětného propojení jednotlivých testovacích podmínek s podklady pro testování, ze kterých daná podmínka vychází, s cíly testování a strategickými cíly, které pokrývá a případně i s produktovými riziky, na jejichž pokrytí se zaměřuje. Následně při přípravě testovacích případů, které z testovacích podmínek vychází, je zde možné zapsat i propojení na jednotlivé testovací případy, které pokrývají jednotlivé testovací podmínky.]

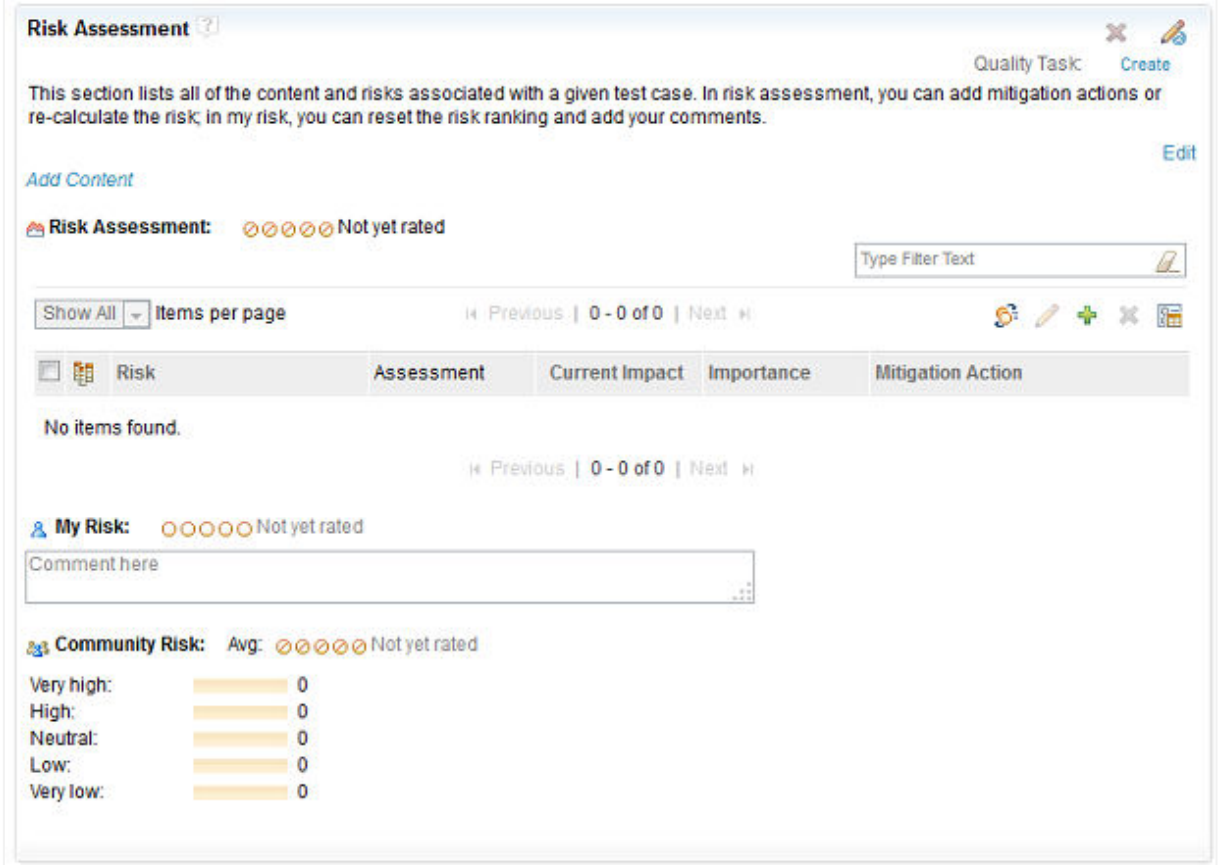
| ID Testovací podmínky | Podklad(-y) pro testování | Cíl(-e) | Produktové riziko | Testovací případy |
|---|---|---|---|---|
| 1. Úroveň hierarchie<br>2. Úroveň hierarchie | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

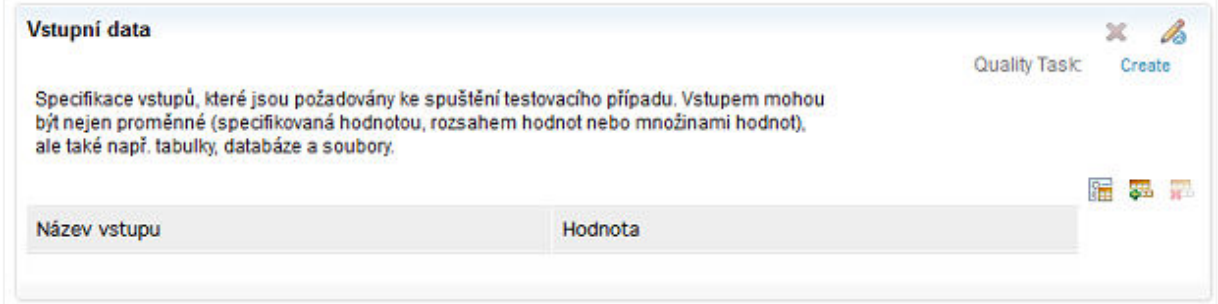
## 2. 2. Popis testovacích podmínek

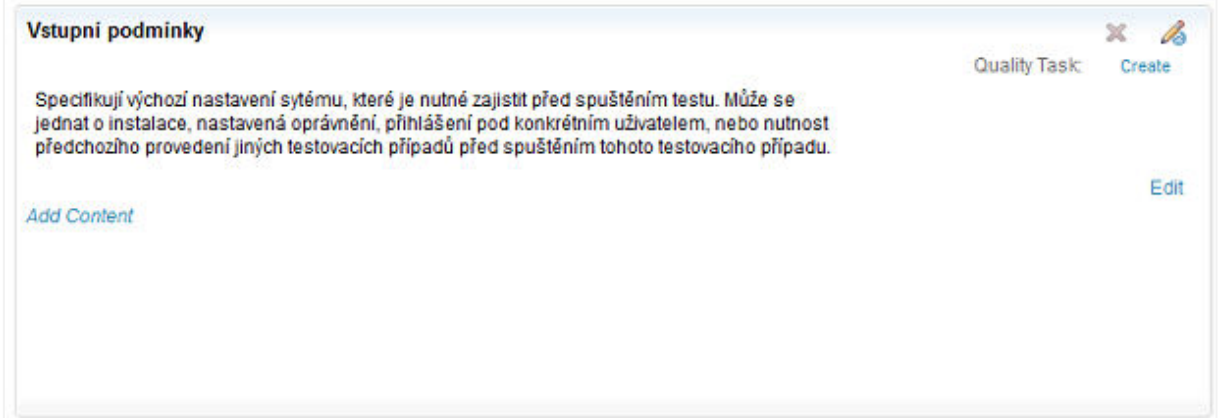
[ID testovací podmínky: Popis testovací podmínky]

Ukázkovou šablonu pracovního produktu **testovací případ** v nástroji IBM Rational Quality Manager zachycuje Obrázek č. 19.




Risk Assessment
Quality Task:
Create
This section lists all of the content and risks associated with a given test case. In risk assessment, you can add mitigation actions or re-calculate the risk; in my risk, you can reset the risk ranking and add your comments.
Edit
Add Content
Risk Assessment:
Not yet rated
Type Filter Text
Show All
Items per page
Previous | 0 - 0 of 0 | Next
Risk
Assessment
Current Impact
Importance
Mitigation Action
No items found.
Previous | 0 - 0 of 0 | Next
My Risk:
Not yet rated
Comment here
Community Risk:
Avg:
Not yet rated
Very high: 0
High: 0
Neutral: 0
Low: 0
Very low: 0


Vstupní data
Quality Task:
Create
Specifikace vstupů, které jsou požadovány ke spuštění testovacího případu. Vstupem mohou být nejen proměnné (specifikovaná hodnotou, rozsahem hodnot nebo množinami hodnot), ale také např. tabulky, databáze a soubory.
Název vstupu
Hodnota


Vstupní podmínky
Quality Task:
Create
Specifikují výchozí nastavení sytému, které je nutné zajistit před spuštěním testu. Může se jednat o instalace, nastavená oprávnění, přihlášení pod konkrétním uživatelem, nebo nutnost předchozího provedení jiných testovacích případů před spuštěním tohoto testovacího případu.
Edit
Add Content

**Očekávané výsledky**

Quality Task: Create

Specifikuje výsledky a očekávané chování testovaných položek. Dokumentuje, jakých výsledků musí být dosaženo, aby provedený testovací případ mohl být vyhodnocen jako úspěšný. Výstupem mohou být obdobně jako u vstupů data (specifikovaná přesnou hodnotou, rozsahem hodnot nebo množinami hodnot), ale také např. tabulky, databáze a soubory. Očekávané výsledky a chování mohou být připojeny v souboru nebo v obrázku, který zachycuje stav uživatelského rozhraní nebo výsledného kódu.

Edit

*Add Content*

**Výstupní podmínky**

Quality Task: Create

Specifikují, co je zapotřebí vykonat před spuštěním testovacího případu, aby byl systém uveden do původního stavu. To může zahrnovat smazání vytvořených souborů, odinstalování některých komponent, nebo nutnost následného provedení jiných testovacích případů.

Edit

*Add Content*

**Potřeby prostředí**

Quality Task: Create

Popis testovacího potředí, které je zapotřebí pro spuštění a provádění testovacího případu a záznam výsledků testovacího případu. Jedná se o specifikaci charakteristik a konfigurací hardwaru konfigurace systému (operační systémy, kompilátory, simulátory a testovací nástroje) nebo interakce s ostatním aplikačním softwarem. Dále zde mohou být další dosud nazahrnuté požadavky jako požadované schopnosti a školení. Tuto sekci má smysl vyplňovat pouze pokud zahrnuje dodatečné informace oproti plánu testování.

Edit

*Add Content*

**Test Scripts**

Quality Task: Create

Automated or manual test scripts that are associated with this test case. Scripts can be reused.

Group By: Ungrouped

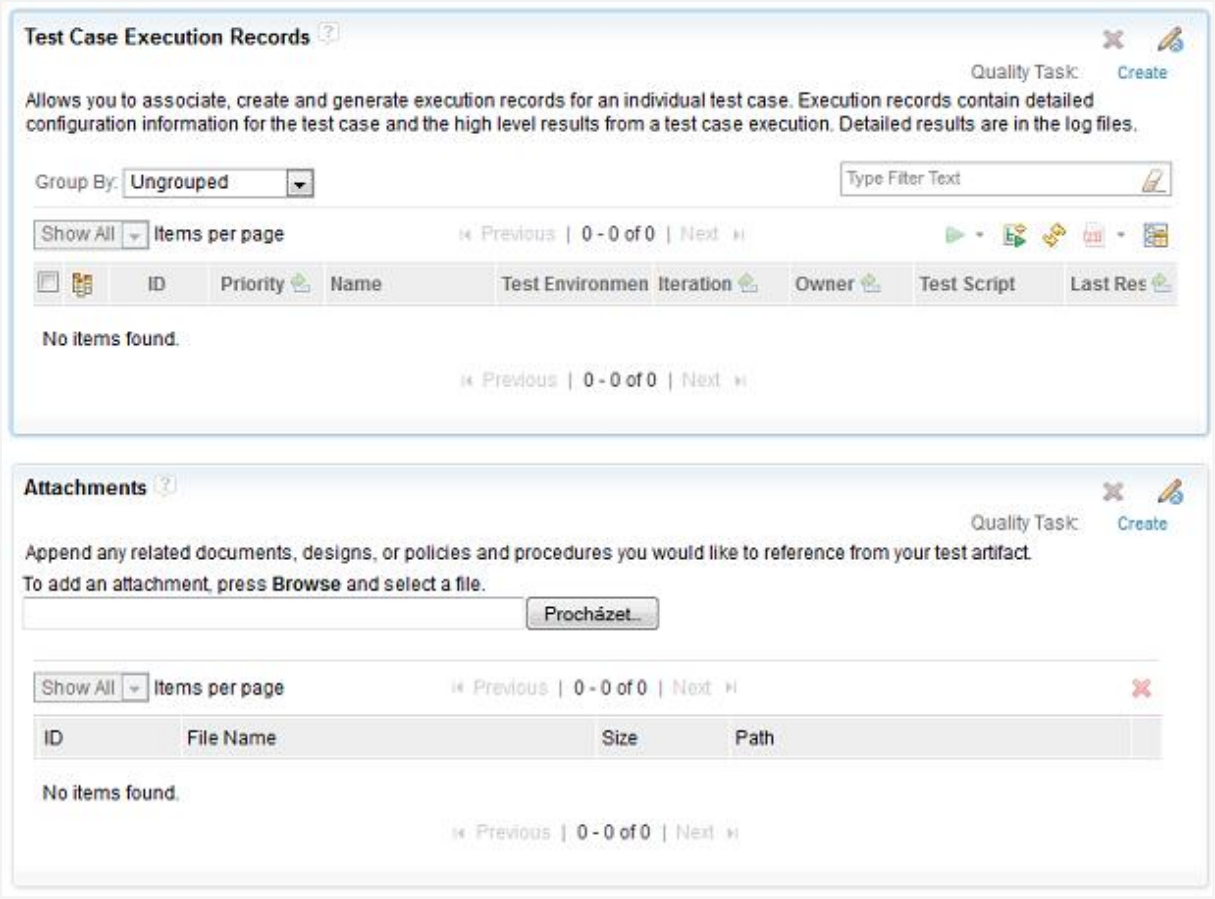
Type Filter Text

Show All Items per page

Previous | 0 - 0 of 0 | Next

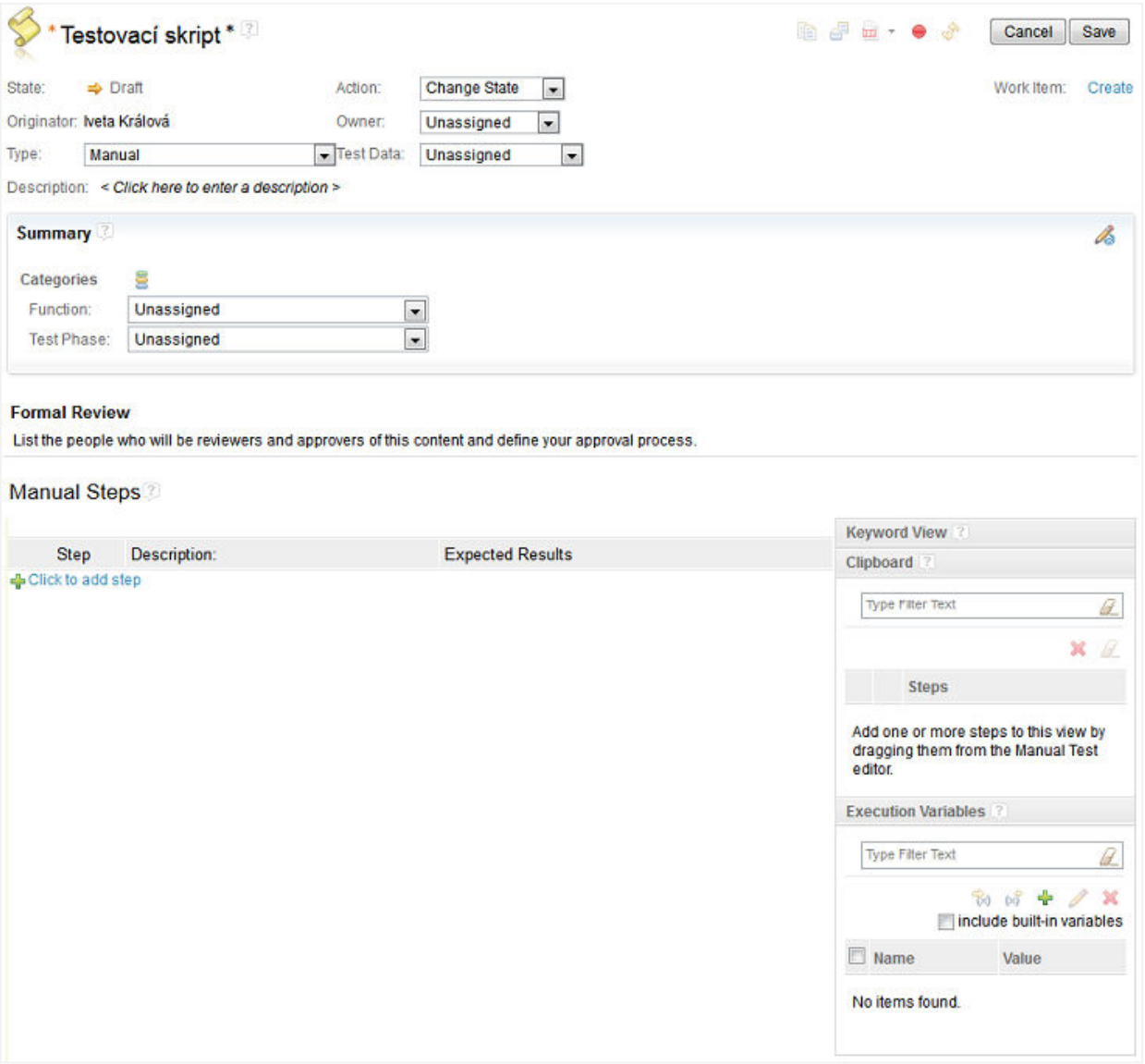
| ID | Name | State | Script Type | Owner | Function | Test Phase | Modified | Validates Requirement |
|---|---|---|---|---|---|---|---|---|

Previous | 0 - 0 of 0 | Next



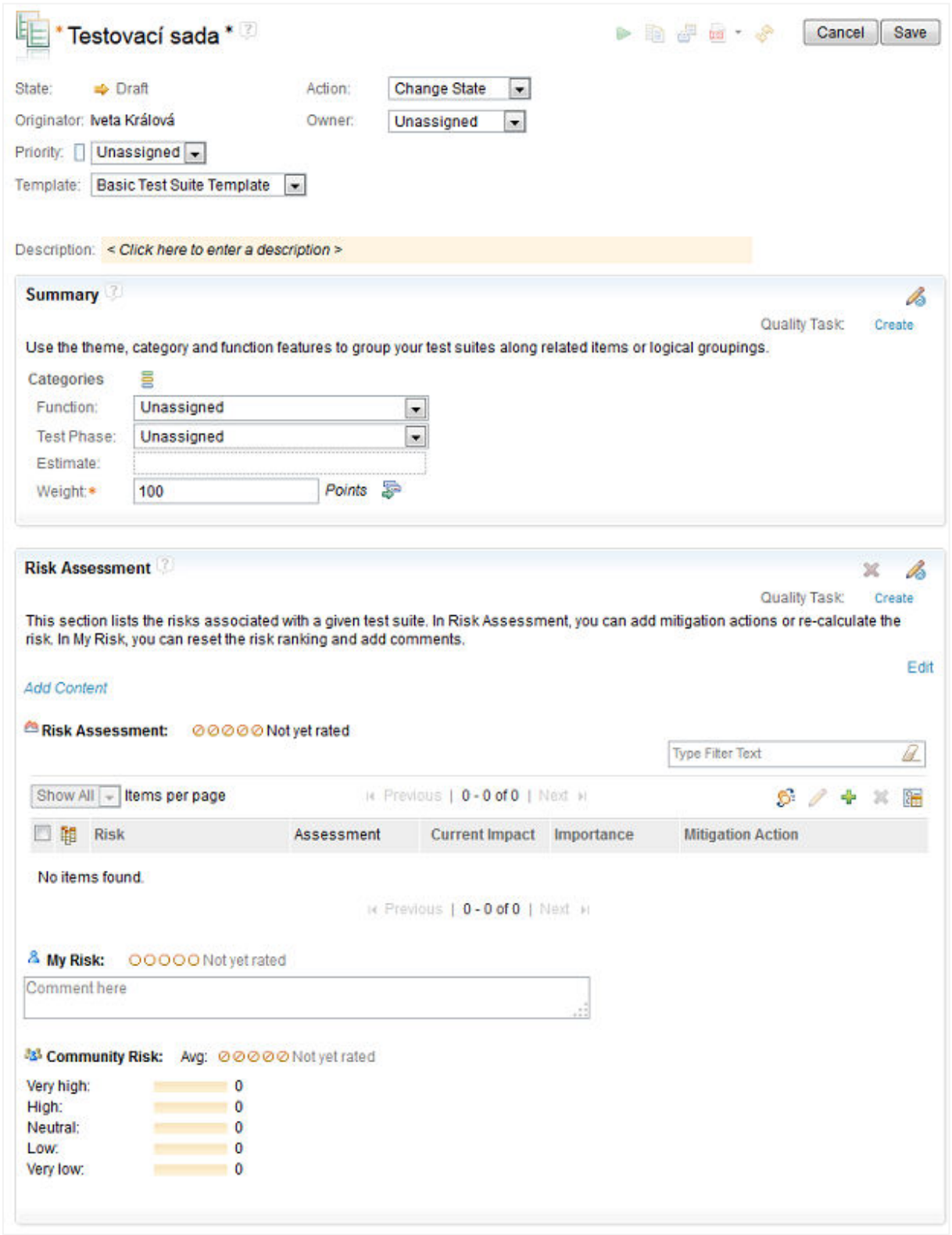

**Obrázek č. 19 Šablona pro testovací případ vytvořená v nástroji IBM Rational Quality Manager, zdroj: autor podle [IEEE, 2008, SQE, 2001a]**

Ukázku výchozí šablony pracovního produktu **testovací skript** v nástroji IBM Rational Quality Manager zachycuje Obrázek č. 20.




**Obrázek č. 20 Výchozí šablona pracovního produktu testovací skript definovaná nástrojem IBM Rational Quality Manager, zdroj: [IBM, 2012]**

Ukázkovou šablonu pracovního produktu **testovací sada** v nástroji IBM Rational Quality Manager zachycuje Obrázek č. 21.



**Vstupní data**

Quality Task: Create

Specifikace vstupů, které jsou požadovány ke spuštění testovací sady. Vstupem mohou být nejen proměnné (specifikovaná hodnotou, rozsahem hodnot nebo množinami hodnot), ale také např. tabulky, databáze a soubory.

**Vstupní podmínky**

Quality Task: Create

Specifikují výchozí nastavení sytému, které je nutné zajistit před spuštěním testovací sady. Může se jednat o instalace, nastavená oprávnění, přihlášení pod konkrétním uživatelem, nebo nutnost předchozího provedení jiných testovacích sad před spuštěním této testovací sady.

**Očekávané výsledky**

Quality Task: Create

Specifikuje výsledky a očekávané chování testovaných položek. Dokumentuje, jakých výsledků musí být dosaženo, aby provedená testovací sada mohla být vyhodnocena jako úspěšná. Výstupem mohou být obdobně jako u vstupů data (specifikovaná přesnou hodnotou, rozsahem hodnot nebo množinami hodnot), ale také např. tabulky, databáze a soubory. Očekávané výsledky a chování mohou být připojeny v souboru nebo v obrázku, který zachycuje stav uživatelského rozhraní nebo výsledného kódu.

**Výstupní podmínky**

Quality Task: Create

Specifikují, co je zapotřebí vykonat před spuštěním testovací sady, aby byl systém uveden do původního stavu. To může zahrnovat smazání vytvořených souborů, odinstalování některých komponent, nebo nutnost následného provedení jiných testovacích sad.

**Potřeby prostředí**

Quality Task: Create

Popis testovacího potředí, které je zapotřebí pro spuštění a provádění testovací sady a záznam výsledků testovací sady. Jedná se o specifikaci charakteristik a konfigurací hardwaru konfigurace systému (operační systémy, kompilátory, simulátory a testovací nástroje) nebo interakce s ostatním aplikačním softwarem. Dále zde mohou být další dosud nazahrnuté požadavky jako požadované schopnosti a školení. Tuto sekci má smysl vyplňovat pouze pokud zahrnuje dodatečné informace oproti plánu testování.

**Test Cases**

Quality Task: Create

List of test cases in the test suite.

Run this suite in a sequence.

Pass execution variable(s) between scripts

Stop suite execution if any test does not pass

Run this suite in parallel.

Test Cell Unassigned

Group By: Ungrouped

Show All Items per page Previous | 0 - 0 of 0 | Next

| Execution Orde | ID | Priority | Test Case | Test Environment | State | Owner | Test Script | Modified |
|---|---|---|---|---|---|---|---|---|

No items found.

Previous | 0 - 0 of 0 | Next

**Vstupní data**

Quality Task: Create

Specifikace vstupů, které jsou požadovány ke spuštění testovací sady. Vstupem mohou být nejen proměnné (specifikovaná hodnotou, rozsahem hodnot nebo množinami hodnot), ale také např. tabulky, databáze a soubory.

**Vstupní podmínky**

Quality Task: Create

Specifikují výchozí nastavení sytému, které je nutné zajistit před spuštěním testovací sady. Může se jednat o instalace, nastavená oprávnění, přihlášení pod konkrétním uživatelem, nebo nutnost předchozího provedení jiných testovacích sad před spuštěním této testovací sady.

**Očekávané výsledky**

Quality Task: Create

Specifikuje výsledky a očekávané chování testovaných položek. Dokumentuje, jakých výsledků musí být dosaženo, aby provedená testovací sada mohla být vyhodnocena jako úspěšná. Výstupem mohou být obdobně jako u vstupů data (specifikovaná přesnou hodnotou, rozsahem hodnot nebo množinami hodnot), ale také např. tabulky, databáze a soubory. Očekávané výsledky a chování mohou být připojeny v souboru nebo v obrázku, který zachycuje stav uživatelského rozhraní nebo výsledného kódu.

**Výstupní podmínky**

Quality Task: Create

Specifikují, co je zapotřebí vykonat před spuštěním testovací sady, aby byl systém uveden do původního stavu. To může zahrnovat smazání vytvořených souborů, odinstalování některých komponent, nebo nutnost následného provedení jiných testovacích sad.

**Potřeby prostředí**

Quality Task: Create

Popis testovacího potředí, které je zapotřebí pro spuštění a provádění testovací sady a záznam výsledků testovací sady. Jedná se o specifikaci charakteristik a konfigurací hardwaru konfigurace systému (operační systémy, kompilátory, simulátory a testovací nástroje) nebo interakce s ostatním aplikačním softwarem. Dále zde mohou být další dosud nazahrnuté požadavky jako požadované schopnosti a školení. Tuto sekci má smysl vyplňovat pouze pokud zahrnuje dodatečné informace oproti plánu testování.

**Test Cases**

Quality Task: Create

List of test cases in the test suite.

Run this suite in a sequence.

Pass execution variable(s) between scripts

Stop suite execution if any test does not pass

Run this suite in parallel.

Test Cell Unassigned

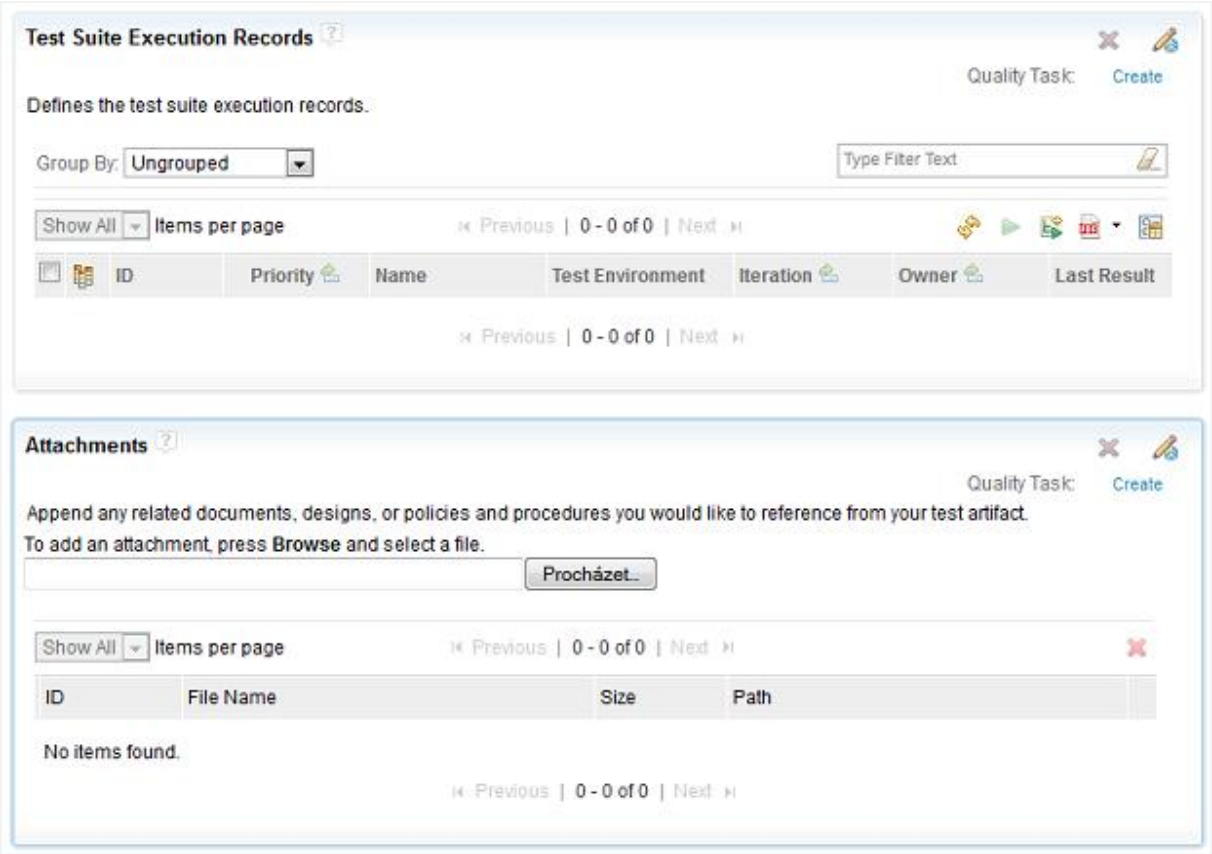
Group By: Ungrouped

Show All Items per page Previous | 0 - 0 of 0 | Next

| Execution Orde | ID | Priority | Test Case | Test Environment | State | Owner | Test Script | Modified |
|---|---|---|---|---|---|---|---|---|

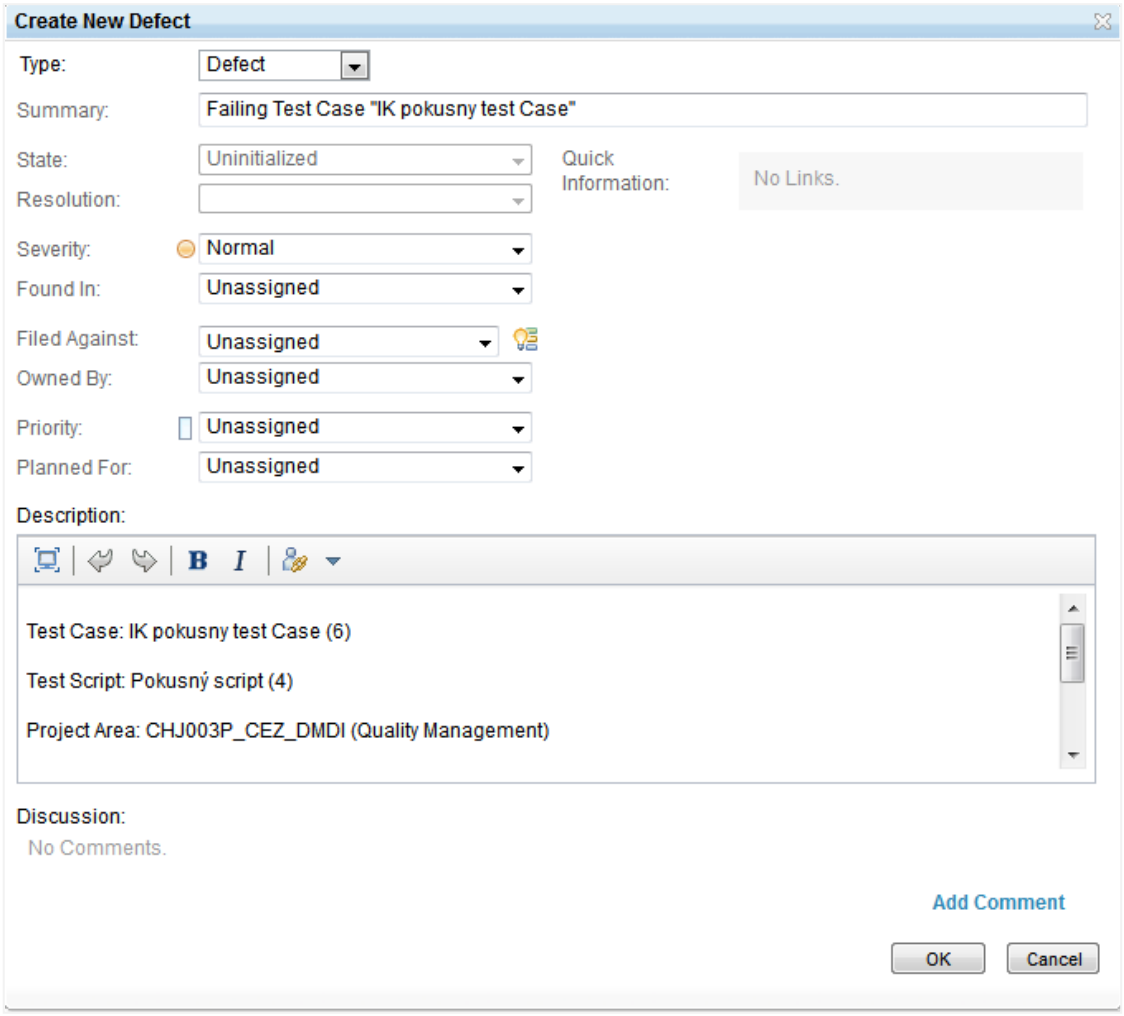
No items found.

Previous | 0 - 0 of 0 | Next



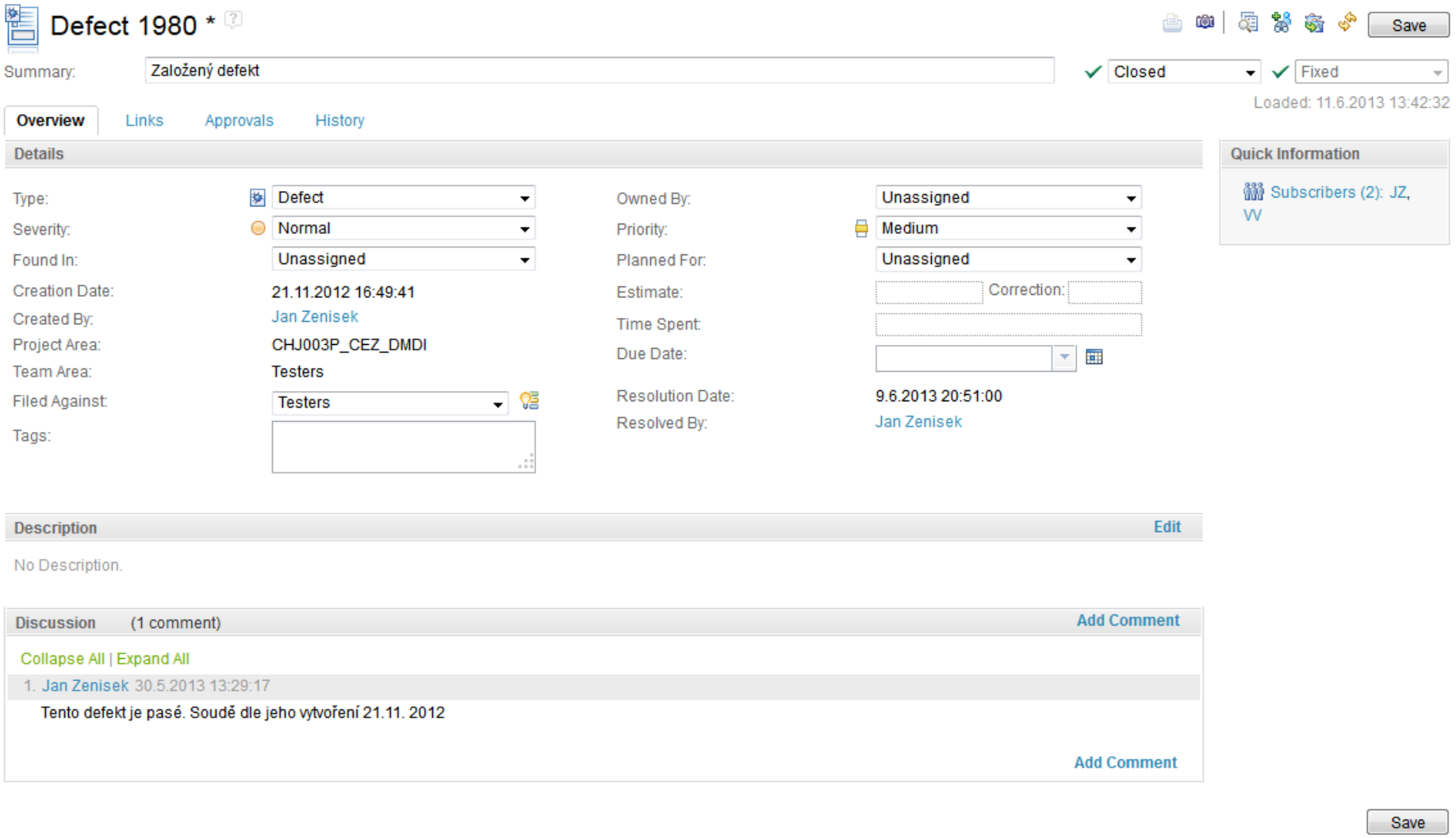

**Obrázek č. 21 Šablona pro testovací sadu vytvořená v nástroji IBM Rational Quality Manager, zdroj: autor podle [IEEE, 2008, SQE, 2001a]**

Formulář pro založení nového **defekt reportu** v nástroji IBM Rational Quality Manager zachycuje Obrázek č. 22.




**Obrázek č. 22 Formulář pro založení nového defekt reportu v nástroji IBM Rational Quality Manager, zdroj [IBM, 2012]**

Obrázek č. 23 zachycuje strukturu již založeného **defekt reportu** v nástroji IBM Rational Quality Manager.




**Obrázek č. 23 Ukázka struktury založeného defekt reportu v nástroji IBM Rational Quality Manager, zdroj [IBM. 2012]**

# Příloha D: Test Maturity Model integration[47]

**Model Test Maturity Model Integration (TMMi)** je jedním z nejnovějších modelů pro zlepšování procesů testování a snaží se zlepšování procesů testování standardizovat.

Ještě než přistoupím k vlastní charakteristice modelu TMMi, je zapotřebí nejdříve vymezit účel a koncept modelů zlepšování procesů testování.

**Modely zlepšování procesů** testování poskytují nástroje pro systematické zlepšování procesů testování s využitím v praxi ověřených postupů a praktik. Modely zlepšování procesů testování se podrobně zabývá ve své velmi zdařilé diplomové práci Tomáš Došek. Při charakteristice modelů zlepšování procesů testování proto vycházím z jeho myšlenek [Došek, 2012, s. 23-25].

**Účelem modelů** zlepšování procesů je dle [Andersin, 31, s. 3] *„optimalizování kvality, nákladů a doby realizace testování ve vztahu k celkovým informačním službám“*. Modely zlepšování procesů testování jsou nástroji pro systematické zlepšování, které udávají, co a v jakém pořadí je třeba zlepšovat. Transformují komplexní problematiku oboru testování do podoby systematických postupů a praktik, jejichž účinnost a efektivita je ověřena zkušenostmi. Tím poskytují jistotu, že aplikace znalostí v modelech může přinést požadované efekty.

Rozsáhlá škála aktivit testování je v modelech obvykle rozdělena do procesních oblastí, které se následně dělí na menší a menší komponenty, až do úrovně požadavků.

**Ideou modelů** je postupné zlepšování procesů testování. Není reálné, aby podnik jedním krokem dospěl do ideálního stavu. Navíc ideální stav procesů, kdy reálné procesy testování splňují všechny požadavky ideálního procesu modelu, nemusí být v zájmu podniku. Proto modely obvykle definují úrovně kvality procesů označované jako **úrovně zralosti** a procesy testování se po jednotlivých krocích dostávají z nižších úrovní zralosti do vyšších úrovní zralosti. Zavádění ověřených praktik testování potom začíná nejprve aplikací základních prvků plánování a řízení testování, zavádění víceúrovňového testování, zavádění prvků verifikace a řízení rizik, které pomáhají při určení optimálního rozsahu testování, a pokračuje až po dosažení prevenčně orientovaného testování, které přináší maximální redukci nákladů a času.

Většina modelů se nesoustředí pouze na dosažení určité úrovně testování, ale snaží se o **trvalé zlepšování** procesů testování. To je založeno na organizaci a měření procesů testování na celopodnikové úrovni a následném využití zpětné vazby. Výsledkem je neustálé zlepšování testování, úspory z rozsahu a úspory založené na znovupoužitelnosti produktů testování.

---

[47] Text této přílohy jsem převzala z vlastní semestrální práce na předmět Řízení kvality softwaru [Králová, 2013] a mírně jej poupravila.

**Modelů** pro zlepšování procesů testování dosud již bylo **vyvinuto mnoho** a liší se svým zaměřením, rozsahem, flexibilitou, strukturou, obsahem i dostupností. Některé modely mohou být zaměřeny na zlepšování testování na samostatných projektech, jiné jsou určeny pro řízení procesu testování na celopodnikové úrovni. Méně flexibilní modely mají pevně daný postup pro zlepšování, kdežto flexibilní modely umožňují podnikům zvolit prioritní oblasti a přizpůsobit proces a pořadí zlepšování potřebám podniku.

Modely se mimo jiné liší i svým původem. Některé vznikly na akademické půdě, jiné byly vytvořeny komerčními společnostmi nebo nezávislými softwarovými organizacemi. Tvorba nových modelů obvykle staví na již existujících modelech. Častou inspirací pro nové modely jsou modely Test Maturity model (TMM), Capability Maturity model (CMM a jeho novější varianta CMMi) a Test Proces Improvement (TPI). Jelikož dle [Jha, 2012] se modely TPI a TMM v softwarovém odvětví nestaly ve světě tak oblíbenými jako CMM a CMMi, model TMM byl (obdobně jako u CMM a CMMi) předstihnut modelem TMMi, který se v softwarovém odvětví ujal a dále se rozšiřuje.

Tomáš Došek se ve své diplomové práci věnuje srovnání modelů CMMi (Capability Maturity Model Integration, TPI (Test Process Improvement), CTP (Critical Testing Processes) a TMMi (Test Maturity Model integration).

Kromě těchto modelů jsou často dále uplatňovány také modely BTM (Beizer´s Testing Model), TOM (Test Organization Maturity Model), TMAP (Test Management Approach) a TMAP Next (Test Managment Approach for Next generation), TPI Next, TCMM (Technical Capability Maturity Model), TSM (Technical Support Model), TIM (Test Improvement Model) a další [Jha, 2012].

Vzhledem k tomu, že pro objektivní zhodnocení navržené metodiky pro testování jsem se rozhodla využít model TMMI, dále se zaměřuji na charakteristiku tohoto modelu. Jedná se o jeden z nejnovějších modelů, který je perspektivní do budoucna. Byl vytvořen nezávislou softwarovou organizací, je v souladu s **materiály pro ISTQB certifikace** a stává se **uznávaným standardem oboru testování**.

## D.1 Charakteristika modelu TMMi

Model Test Maturity Model integration (TMMi) byl vytvořen mezi lety 2006 a 2008 nezávislou neziskovou organizací TMMi Foundation. Cílem této organizace bylo vyvinout a dále rozvíjet rozsáhlý, robustní a veřejně dostupný model pro zlepšování procesů testování softwaru. [Došek, 2012, s. 41] TMMi je procesním rámcem pro testování v organizaci, který umožňuje organizacím hodnotit a zlepšovat procesy testování. Identifikuje procesní oblasti, cíle a praktiky. Dosavadní praktické zkušenosti s modelem TMMi již ukázaly, že aplikování kritérií zralosti dle TMMi pomáhá zlepšovat procesy testování a má pozitivní dopad na kvalitu produktu, produktivitu testování a dobu trvání testování. [Veenendaal, 2013, s. 1]

Před vznikem modelu TMMi softwarové odvětví při zlepšování kvality vývojového procesu hojně využívalo zralostního modelu CMMi, který je následníkem dříve rozšířeného modelu CMM. Modely CMM a CMMi jsou dosud považovány za standard pro zlepšování procesů vývoje softwaru. Přestože CMMi obsahuje dvě oblasti (verifikace a validace), které se věnují testování, stále postrádá praktické nástroje, jak zlepšit procesy testování krok za krokem. Důraz klade především na organizační, softwarové a systémově inženýrské procesy, ale nezaměřuje se přímo na charakteristiky zralosti procesu testování. Z tohoto důvodu vytvořila organizace TMMi Foundation vlastní zralostní model, speciálně zaměřený právě na zlepšování procesů testování, který model CMMi doplňuje. [Veenendaal, 2013, s. 1]

Struktura TMMi přímo vychází ze struktury CMMi a v některých částech se na model CMMi odkazuje (jako například u procesní oblasti configuration management, kde praktiky pro pracovní produkty dle CMMi jsou aplikovatelné na pracovní produkty testování). Rozsah TMMi pokrývá jak testovací aktivity systémově inženýrského charakteru, které pokrývají celkově vývoj systémů (což může, ale nemusí zahrnovat software), tak testovací aktivity zaměřené na softwarově inženýrské disciplíny, které pokrývají přímo vývoj softwarových systémů. [TMMi Foundation, 2012a, s. 7]

V době vzniku TMMi již existovala řada modelů pro zlepšování procesů testování. Jednotlivé modely ovšem měly různý účel a nepoužívaly jednotnou terminologii a standardy. TMMi navazuje na dřívější modely pro zlepšování procesů testování, ale navíc je v souladu s mezinárodními standardy oboru testování. Terminologie TMMi vychází ze slovníku ISTQB, dále TMMi respektuje IEEE 829 Standard for Software Test Documentation a požadavky na hodnocení zralosti procesu testování jsou v souladu se standardem ISO 15504. [TMMi Foundation, 2012a, s. 7]

TMMi Model může být využíván samostatně, nebo společně s ostatními procesními modely. Kromě volně dostupného referenčního modelu jsou volně k dispozici také požadavky na metody hodnocení, které jsou v souladu se standardem ISO 15504. Konkrétní metodu hodnocení zralosti procesů dle modelu TMMi ovšem zatím organizace TMMi neposkytla. Jedním z přínosů diplomové práce Tomáše Doška je právě navržení konkrétní metody hodnocení dle požadavků TMMi. Úspěšnost navržené metody byla v zápětí potvrzena posudky dvou expertů v oboru testování. Jedním z nich je dokonce samotný místopředseda představenstva TMMi Foundation Eric van Veenendaal.

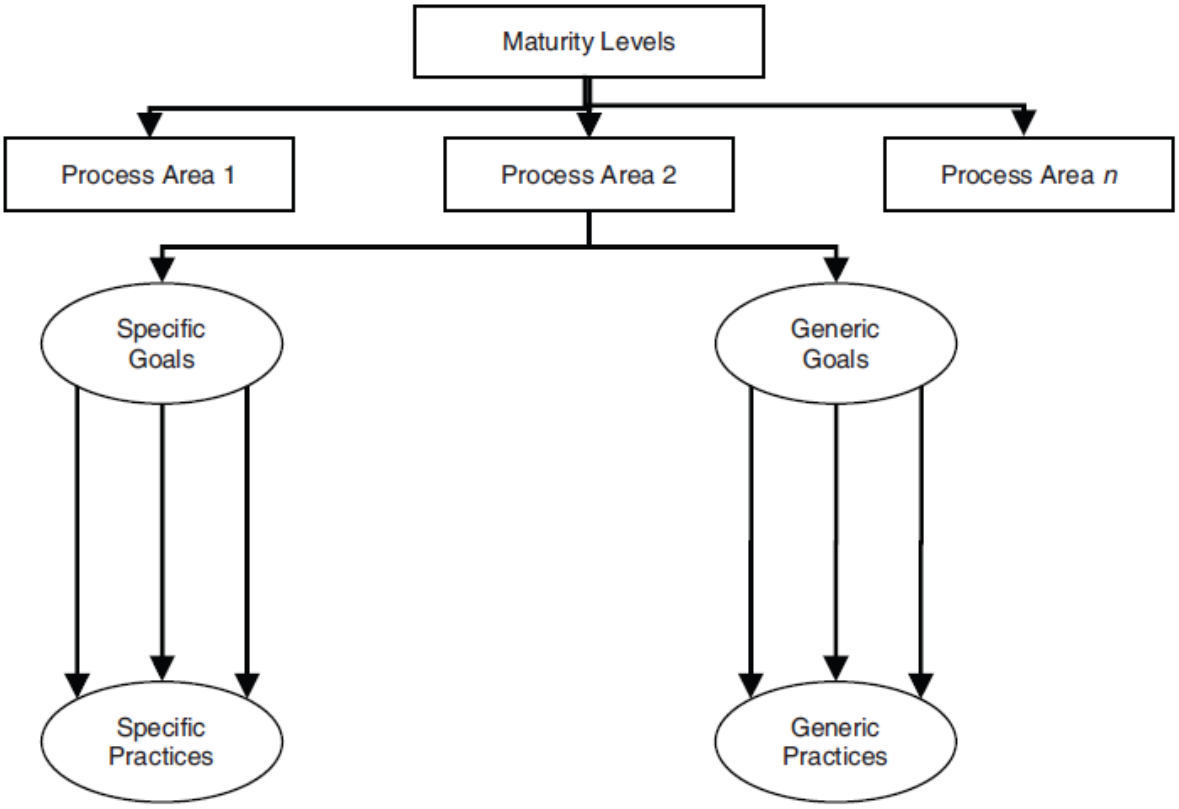
Kromě samotného referenčního modelu organizace TMMi Foundation také poskytuje školení v používání modelu TMMi, akreditace hodnotitelů a akreditace metod hodnocení procesů. [TMMi Foundation, 2012b]

Referenční model je podrobně popsán v dokumentu Test Maturity Model Integration (TMMi). Požadavky na metody hodnocení jsou uvedeny v dokumentu TMMi Assessment Method Application Requirements (TAMAR). Práce Tomáše Doška se vztahuje k verzím v3.1 dokumentu TMMi a verzi TAMAR v2 [Došek, 2012, s. 42]. Ke květnu 2013 zůstává

TAMAR ve verzi v2, ale dokument TMMi byl v průběhu roku 2012 aktualizován na release 1.0. Oproti předchozím verzím přidává detailní popis procesních oblastí páté úrovně zralosti, tzn., doplňuje oblast prevence defektů, řízení kvality a oblast optimalizace procesu testování. [TMMi Foundation, 2012a, s. 4]

### D.1.1 Struktura modelu[48]

Pro využití modelu TMMi ke zlepšování procesů testování a pro hodnocení zralosti procesů testování je třeba se dobře orientovat ve struktuře modelu.




**Obrázek č. 24 Struktura modelu TMMi, zdroj: [TMMi foundation, 2012a, s. 14]**

Struktura modelu TMMi je založena na struktuře modelu CMMi. Stejně jako CMMi rozlišuje mezi praktikami, které jsou povinné, nebo se doporučuje je implementovat.

Komponenty modelu TMMi jsou rozděleny do následujících třech kategorií: Povinné (required), očekávané (expected) a informativní (informative).

**Povinné komponenty** popisují, čeho musí organizace dosáhnout, aby byla daná procesní oblast uspokojena, a bez kterých není dané úrovně zralosti dosaženo. Povinnými komponentami jsou v modelu TMMi specifické a generické cíle, viz dále.

[48] Text této podkapitoly vychází ze zdroje [TMMi Foundation, 2012a, s. 13-15]

**Očekávané komponenty** popisují, co by měla organizace typicky implementovat, aby dosáhla požadovaných komponent. Očekávanými komponentami jsou v modelu TMMi specifické a generické praktiky, viz dále. Organizace se nemusí řídit přesně podle těchto praktik, ale pokud se těmito doporučovanými praktikami neřídí, musí alespoň aplikovat odpovídající alternativu.

**Informativní komponenty** jsou pro organizaci inspirací, jak dosáhnout povinných komponent a jak aplikovat očekávané komponenty. Informativní komponentami jsou v modelu TMMi sub-praktiky, vzorové pracovní produkty, poznámky, příklady a doporučení.

Vztahy mezi komponentami modelu TMMi zachycuje Obrázek č. 24. Následuje stručný popis základních komponent modelu.

Na vrcholu hierarchie stojí **úroveň zralosti *(maturity level)***, která vyjadřuje stupeň kvality procesu testování. Pro dosažení určité úrovně zralosti musí organizace uspokojit všechny specifické a generické cíle všech procesních oblastí, které daná úroveň zralosti obsahuje a taktéž i cíle nižších úrovní zralosti.

**Procesní oblast *(process area)*** identifikuje, kam má organizace zaměřit zlepšování svých procesů, aby dosáhla dané úrovně zralosti. Procesní oblasti jsou základním rámcem sdružujícím druhově podobné aktivity jako je například plánování testů, návrh a provádění testů, nefunkcionální testování, řízení kvality apod.

**Specifické cíle *(specific goals)*** popisují jedinečné charakteristiky, které jsou nutné pro uspokojení dané procesní oblasti. Jedná se o povinné komponenty, které se používají pro vyhodnocování, zda daná procesní oblast splnila požadavky na určitou úroveň zralosti. Specifickým cílem je např. zavést politiku testování, vytvořit plán testování, poskytovat výsledky měření testování, zjistit obvyklé příčiny defektů, apod.

**Generické cíle *(generic goals)*** se objevují skrze více procesních oblastí. Stejně jako u procesních cílů se jedná o povinné komponenty, které se používají pro vyhodnocování, zda daná procesní oblast splnila požadavky na určitou úroveň zralosti. Generickým cílem je např. zavést definovaný proces nebo řídit proces.

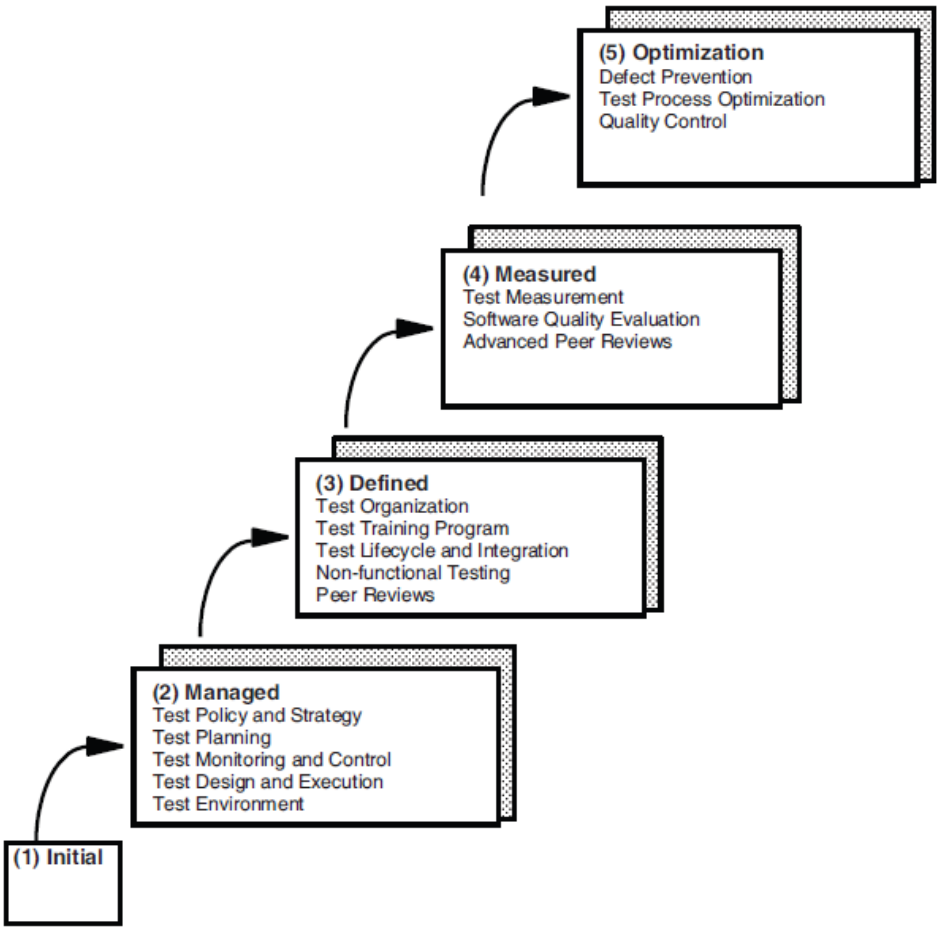
**Specifické praktiky *(specific practices)*** popisují aktivity, kterými lze nejlépe dospět k naplnění specifických cílů. Jedná se o očekávané komponenty. Očekává se, že pokud budou aplikovány, tak bude dosaženo specifických cílů pro danou procesní oblast. Mezi specifické praktiky se řadí aktivity jako např. sestav harmonogram testů, identifikuj projektová rizika testování, zaveď standardní proces testování, definuj nefunkcionální testovací přístup, vyber defekty pro analýzu, apod.

Specifické praktiky mohou být dále obohaceny o **sub-praktiky *(sub-practices)***, což jsou návody pro interpretaci a implementaci specifických praktik, o **vzorové pracovní produkty *(example work products)*** a další **rozšiřující informace *(notes, examples, references)***.

**Generické praktiky** ***(generic practices)*** se objevují napříč několika procesními oblastmi. Popisují aktivity, kterými lze nejlépe dospět k naplnění generických cílů. Jedná se o očekávané komponenty. Mezi generické praktiky se řadí aktivity např. naplánuj proces, přiděl odpovědnosti, zaškol lidi, monitoruj a řiď proces, apod.

### D.1.2 Úrovně zralosti

Model TMMi má stupňovitou reprezentaci. Základem je pět úrovní zralosti (jednotlivých stupňů), kterými organizace při zlepšování procesů testování postupně prochází. Nejnižší úrovní jsou neřízené a ad hoc procesy. Tato úroveň neklade žádné




**Obrázek č. 25 Úrovně zralosti a procesní oblasti v modelu TMMi, zdroj: [TMMi Foundation, 2012a, s. 9]**

požadavky pro splnění a je výchozí úrovní pro všechny organizace, kde dosud nebyla snaha o zlepšování procesů testování, a kde nebyly dosud splněny požadavky vyšších úrovní. Dosažení následujících úrovní, tj. úrovně řízené, definované, měřené a optimalizované, již vyžaduje splnění specifických požadavků, které jsou indikátorem zlepšování procesů testování. Jednotlivé úrovně zralosti a jejich procesní oblasti znázorňuje Obrázek č. 25.

Dosažení každé úrovně zralosti je předpokladem pro dosažení následující úrovně zralosti. Snaha o přeskočení některé úrovně je obvykle kontraproduktivní. Vnitřní struktura TMMi poskytuje množství praktik, jejichž použití pomáhá při systematickém zlepšování procesů testování v postupných krocích. Každá úroveň je zaměřena na určitý počet procesních oblastí, ve kterých má organizace dosáhnout zlepšení, aby dané úrovně zralosti bylo dosaženo. Zkušenosti ukázaly, že v jednom kroku nelze dosáhnout zlepšení ve všech procesních oblastech najednou a zlepšování by se vždy mělo soustředit pouze na omezený počet oblastí. Ve vyšších stupních je vyžadováno celopodnikové řízení a měření procesu testování a následně i prvky trvalého zlepšování a zpětné vazby na celopodnikové úrovni, čehož nelze dosáhnout bez zavedení základních prvků řízení testování. [TMMi Foundation, 2012a, s. 9]

Vzhledem k tomu, že navržená metodika testování je hodnocena oproti úrovni zralosti 2, podrobněji na tomto místě charakterizuji pouze první a druhou úroveň zralosti.

### *C.1.2.1 Úroveň 1 – Počáteční (Initial)*[49]

Na úrovni zralosti 1 je testování chaotický, nedefinovaný proces, který často ani není oddělen od debuggingu. Organizace typicky neposkytuje stabilní prostředí pro podporu procesů testování. Úspěch testování závisí na kompetencích a hrdinství jednotlivců, kteří se obětují ve prospěch organizace a vynaloží potřebné úsilí. Nicméně takto dosažený úspěch nelze přisuzovat prostředkům řízeného a opakovatelného procesu testování. Testy jsou připravovány ad hoc, až po dokončení kódování. Testování je prokládáno debuggingem za účelem odstranění defektů ze systému. Cílem testování na první úrovni je pouze ukázat, že software je provozuschopný bez významnějších selhání. Výsledné produkty jsou nasazovány do produkčního prostředí, aniž by byly dány na vědomí informace o jejich kvalitě a rizicích. Produkty tak často nenaplňují očekávání zákazníků, jsou nestabilní, nebo příliš pomalé. Pro testování není zajištěn dostatek zdrojů, nástrojů, ani kvalifikovaných testerů. Často testují jen vývojáři.

Na úrovni zralosti 1 nejsou definovány žádné procesní oblasti. Pro úroveň 1 je charakteristická tendence k překračování rozpočtu a stanoveného harmonogramu, nenaplnění očekávání zákazníků, opouštění od procesů v dobách krize a neschopnost zopakovat případné úspěchy procesů.

### *C.1.2.2 Úroveň 2 – Řízená (Managed)*[50]

Na druhé úrovni zralosti se testování stává řízeným procesem, který je jasně oddělen od debuggingu. V kontextu zlepšování procesů testování již na úrovni organizace existuje strategie testování nebo alespoň programová strategie testování, která je určena pro určitou

[49] Text této podkapitoly vychází ze zdroje [TMMi Foundation, 2012a, s. 10]

[50] Text této podkapitoly vychází ze zdroje [TMMi Foundation, 2012a, s. 23]

skupinu projektů. Jsou vytvářeny plány testování, ve kterých je definován přístup k testování, založený na výsledcích hodnocení produktových rizik. K identifikaci rizik se využívají techniky pro řízení rizik. Plán testování definuje, co je cílem testování, kdy bude testování probíhat, jak a kdo za něj bude odpovědný. Zároveň jsou vymezeny závazky jednotlivých stran zainteresovaných na testování. Testování je monitorováno a řízeno směrem k naplnění stanoveného plánu testování. Pokud jsou v průběhu testování zjištěny nějaké odchylky, jsou provedena nápravná opatření. Stav pracovních produktů testování a testovacích prací je dáván na vědomí managementu. Při vytváření testovacích případů se využívají techniky návrhu testů. Nicméně testování se do životního cyklu vývoje softwaru zapojuje relativně pozdě, nejčastěji až během fáze návrhu nebo implementace software.

Testování je rozděleno do jednotlivých úrovní (testy komponent, integrační testy, systémové testy a akceptační testy). Pro každou úroveň testů jsou v rámci strategie testování specifikovány určité cíle.

Hlavním cílem testování na úrovni zralosti 2 je ověřit, zda produkt uspokojuje definované požadavky. Jelikož testování se do životního cyklu vývoje zapojuje až v pozdních fázích, vyskytuje se na této úrovni mnoho problémů kvality vyvíjeného softwaru. Defekty, které vznikly při definování požadavků a při specifikaci návrhu se dostávají až do kódu. Nejsou zde žádné formální revize pracovních produktů a kódu, které by se na tento problém soustředily. Testování je stále považováno za fázi, která následuje až po implementaci.

Na druhé úrovni zralosti jsou předmětem zlepšování následující procesní oblasti:

- Politika a strategie testování,
- plánování testů,
- monitorování a řízení testů,
- návrh a provádění testů,
- testovací prostředí.

### D.1.3 Hodnocení zralosti procesů testování

Předpokladem pro zlepšování procesů testování je určení aktuální úrovně zralosti procesů testování. Požadavky na hodnocení úrovně zralosti procesů testování jsou definovány v dokumentu TAMAR (poslední dostupnou verzí je v2). V tomto dokumentu jsou popsány pouze požadavky, které se při hodnocení musí splnit, ale není zde uveden konkrétní hodnotící přístup. Organizace TMMi předpokládá, že organizace si vyvinou vlastní metodu hodnocení, která odpovídá jejich byznysu a pokud tato metoda splňuje všechny požadavky TAMARU, může být oficiálně akreditována. [Veenendaal, 2013, s. 4]

TAMAR rozlišuje dva typy hodnocení: Formální a neformální.

**Formální hodnocení** vede k oficiálnímu výsledku, zda daná organizace dosáhla určité úrovně zralosti nebo ne. Striktně vyžaduje splnění všech specifických a generických cílů relevantních k daným procesním oblastem. Formální hodnocení jde hodně do hloubky a platí pro něj přísná pravidla. [TMMi Foundation, 2009, s. 8]

**Neformální hodnocení** nevede k oficiálnímu výsledku, nevztahují se na něj tak přísná pravidla jako pro formální hodnocení a obvykle se používá pro identifikaci dalších zlepšování procesu testování. [TMMi Foundation, 2009, s. 8-9]

Rozdíly mezi formálním a neformálním hodnocením shrnuje Tabulka č. 28.

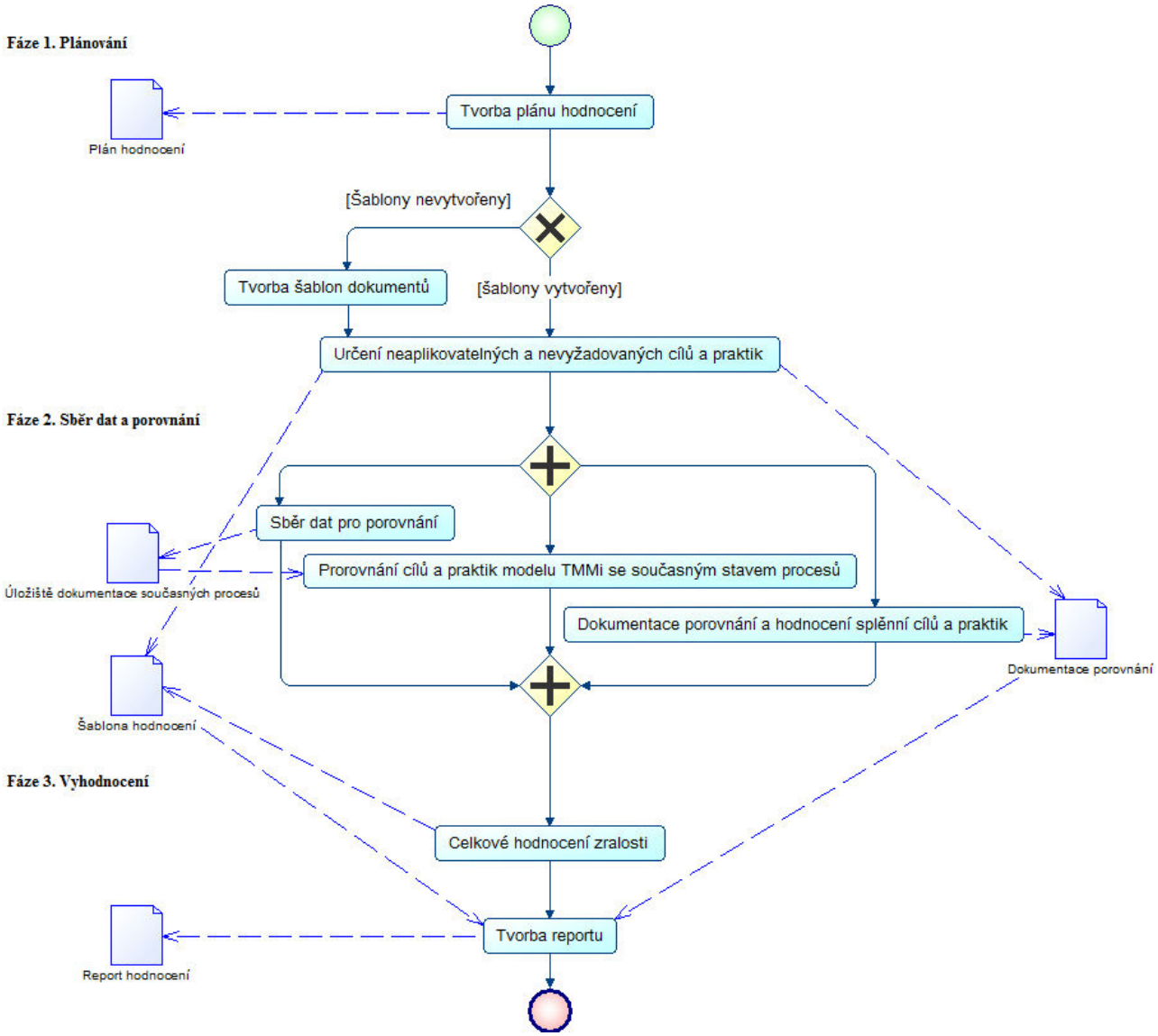
**Tabulka č. 28 Rozdíly mezi formálním a neformálním hodnocení, zdroj: inspirováno [TMMi Foundation, 2009, s. 8-9; Veenendaal, 2013, s. 4-5]**

| Formální hodnocení | Neformální hodnocení |
|---|---|
| Hodnotící tým musí vést hlavní akreditovaný hodnotitel *(lead assessor).*<br><br>(akreditaci uděluje výhradně organizace TMMi Foundation). | Akreditace hodnotitele je doporučována, ale není vyžadována. |
| Hodnotící tým musí obsahovat minimálně dvě osoby. Druhá osoba nemusí mít akreditaci. | Hodnotící tým může tvořit i pouze jedna osoba. |
| Pro potvrzení korektnosti hodnocení je vyžadováno porovnávání informací o procesech testování z mnoha zdrojů. | Pro kompletaci závěrů hodnocení postačuje jeden typ evidence. Není vyžadováno porovnávání různých zdrojů. |
| Vede k oficiálnímu výsledku, na základě kterého je možné získat certifikaci. | Nevede k oficiálnímu výsledku a na základě neformální hodnocení nelze získat certifikaci. |
| Pomalejší, dražší, ale precizní | Rychlejší, levnější, ale méně precizní |

Tomáš Došek ve své diplomové práci navrhnul metodu neformálního hodnocení dle TMMi. V TAMARU je hodnocení rozděleno do následujících třech fází:

- Plánování (Planning),
- sběr dat a porovnání (Data Management),
- vyhodnocení (Process Attribute Rating).

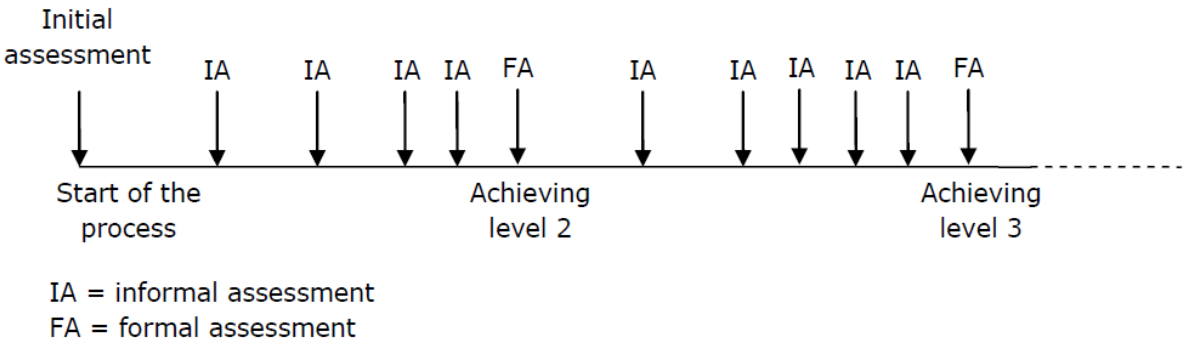
Tyto fáze rozpracoval na úroveň jednotlivých aktivit a jejich výstupů a k výstupním dokumentům vytvořil šablony. Výsledný proces metody hodnocení zachytil v procesním diagramu, viz Obrázek č. 26.




**Obrázek č. 26 Proces metody hodnocení dle TMMi navržený Tomášem Doškem, zdroj: [Došek, 2012, s. 50]**

Navržená metoda hodnocení byla odborně posouzena dvěma hodnotiteli. Místopředseda představenstva TMMi Foundation, Eric van Veenendaal, metodu shledal *„adekvátní pro neformální hodnocení dle TMMi“*. Senior test analytik společnosti Komix, s. r. o., Luboš Uličný, se k navržené metodě vyjádřil, že *„poskytuje konkrétní, praktický a na pochopení i realizaci poměrně jednoduchý přístup k hodnocení podle TMMi“*. Hodnocení metody odborníky bylo pozitivní, a proto autor metody shledává svůj návrh jako úspěšný. [Došek, 2012, s. 73-74]

Rozhodnutí, zda použít formální či neformální hodnocení, závisí na požadavcích a očekáváních organizace. Ukázkový příklad použití těchto dvou typů hodnocení v čase zachycuje Obrázek č. 27.




**Obrázek č. 27 Příklad použití formálního a neformálního hodnocení dle TMMi, zdroj: [Veenendaal, 2013, s. 4]**

Neformální hodnocení *(Informal assessment)* je používáno v průběhu zlepšování k identifikaci zlepšení a k orientačnímu určení stupně zralosti. Formální hodnocení *(Formal assessment)* se provádí v okamžiku, kdy si organizace myslí, že již naplnila všechny požadavky pro danou úroveň zralosti dle TMMi. [Veenendaal, 2013, s. 4]

## D.2 Shrnutí textu přílohy

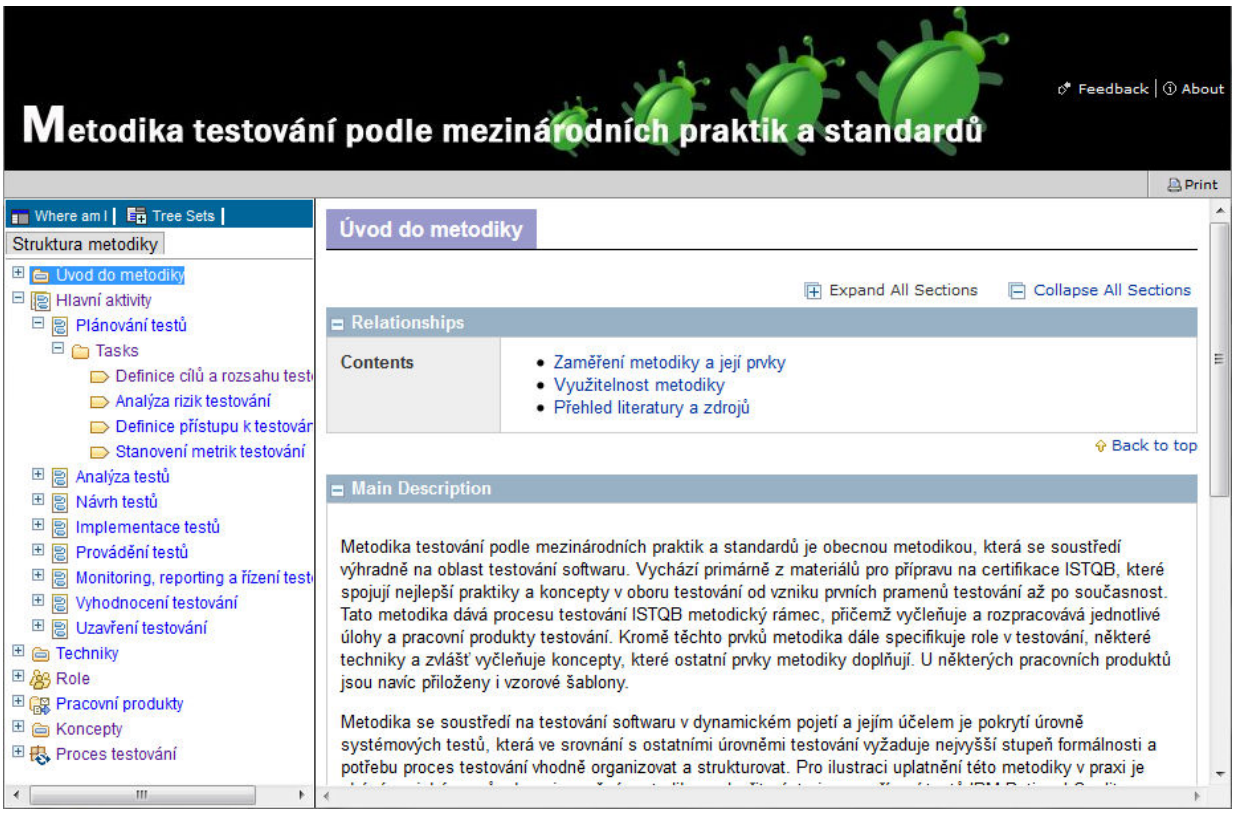
V této příloze byl objasněn princip modelů zlepšování procesů testování. Stěžejní část textu byla zaměřena na model Test Maturity Model integration (TMMi). Jelikož je tento model v souladu s materiály pro ISTQB certifikace, podle kterých byla vytvářena metodika testování v této diplomové práci, byl model TMMi vybrán pro účely objektivního zhodnocení navržené metodiky testování. Jedná se o jeden z nejnovějších modelů, který je perspektivní do budoucna a stává se stejně jako materiály ISTQB standardem oboru testování.

Objektivně mohu zhodnotit, že model TMMi je využitelný zejména pro velké podniky, které vyžadují systematický způsob zlepšování svých procesů. Je zaměřen na celopodnikové řízení procesu testování a strategický rozvoj. Jedná se tedy o dlouhodobé zlepšování a vzhledem k obrovskému rozsahu požadavků není stavěný pro aplikaci na samostatných projektech nebo pro rychlé řešení problémů. V současnosti nabízí pouze stupňovitou reprezentaci, která je omezením, pokud se potřebuje daný podnik zaměřit pouze na zlepšování některých procesních oblastí, nebo potřebuje změnit pořadí aplikovaných praktik. Dosažení úrovně zralosti je podmíněno splněním všech cílů na všechny procesní oblasti na dané úrovni. Řešením problému by bylo zavedení plynulé reprezentace vedle stupňovité, jako je tomu u modelu CMMi. Od použití modelu může také odrazovat náročnost hodnocení a chybějící oficiální konkrétní metoda hodnocení vydaná TMMi Foundation, kterou by podniky samy mohly používat pro neformální hodnocení. Jak již bylo zmíněno, tento nedostatek odhalil Tomáš Došek a vypracoval ve své diplomové práci metodu určenou pro neformální hodnocení. Její použití podnikům může hodnocení úrovně zralosti velmi usnadnit.

Dle mého názoru je důležité vybrat si z modelu TMMi jen oblasti, u kterých má smysl, aby organizace dosáhla zlepšení, a které zefektivní proces testování. Není nutné, aby každá organizace splňovala všechny požadavky na formální certifikaci. Každá organizace má jiné charakteristiky a záleží na konkrétních potřebách organizace, které znalosti z modelu TMMi využije.

# Příloha E: Náhledy z interaktivní verze metodiky vytvořené v nástroji EPF Composer

**Úvodní stránku** z publikované verze metodiky zachycuje Obrázek č. 28.




**Obrázek č. 28 Úvodní stránka metodiky, zdroj: [autor]**

Detail **role tester** zachycuje Obrázek č. 29.

# Role: Tester

Hlavní zodpovědností testera je provádění testů a zaznamenávání jejich výsledků.

Role Sets: Role

Expand All Sections Collapse All Sections

## Relationships


| | |
|---|---|
| **Additionally Performs** | • Archivace pracovních produktů<br>• Monitorování testování<br>• Průběžné reportování o stavu testování<br>• Předání pracovních produktů z procesu testování<br>• Příprava testovacích dat<br>• Retrospektivní mítinky<br>• Vyhodnocení výstupních kritérií<br>• Vytváření testovacích skriptů<br>• Vytvoření souhrnného reportu za testování |
| **Modifies** | • Defekt report<br>• Test log |

Back to top

## Main Description

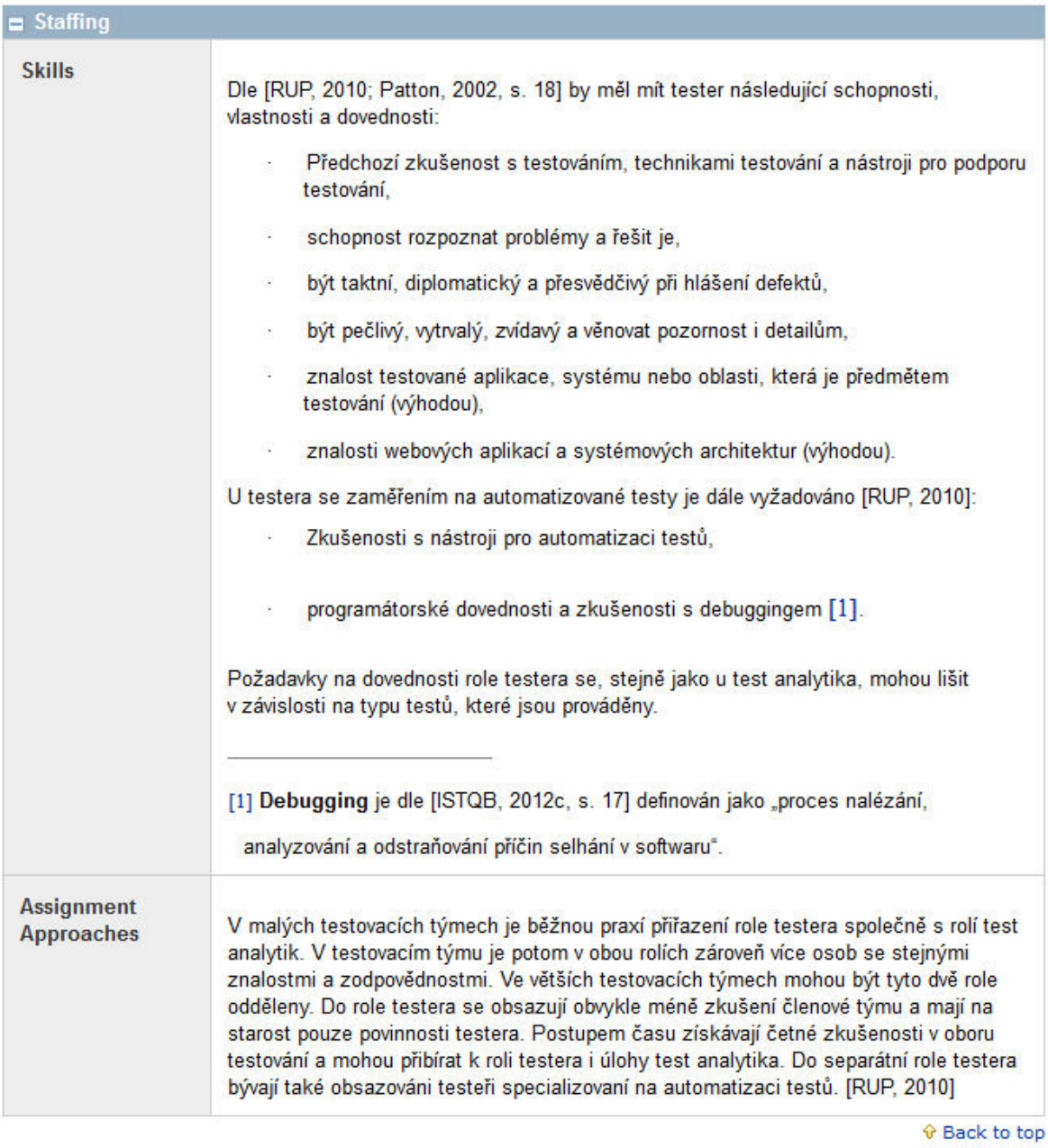
Role testera se do procesu testování zapojuje až ve fázi implementace testů, kde pomáhá připravovat testovací data a ověřuje, zda infrastruktura pro testování je připravena na zahájení provádění testů. Někteří testeři mohou být specializovaní na automatizaci testů. Do jejich zodpovědnosti pak přirozeně spadá nejen provádění testů, ale i vytváření automatizovaných skriptů.

Jakmile jsou splněna vstupní kritéria pro zahájení provádění testů, úlohou testerů je provádět testy podle připravených testovacích případů, sad a skriptů, zaznamenat jejich výsledky do test logu, ohlásit nalezené defekty a následně opakovat testy v souvislosti se změnami v kódu a opravami defektů. Při procházení testů podle připravených testovacích případů by tester měl zapojit kreativitu a nejen splnit povinnost „odklikat" test podle návodu. Dobrý tester rád vyzkouší i věci navíc, nad rámec testovacího případu. Někdy je v rámci provádění testů přímo vyhrazen čas na provádění testů mimo připravené testovací případy, sady a skripty. Uplatňují se reaktivní strategie testování, kde se využívá technik testování založených na zkušenostech a defektech. Účinnost reaktivních strategií přímo závisí na intuici, zkušenostech a znalostech daného testera. Pokud jsou testeři schopni najít defekty v oblastech dosud nepokrytých připravenými testy, je to velkým přínosem pro rozhodování o dalším postupu testování.

V průběhu celého procesu testování tester, stejně jako test analytik, podává test manažerovi průběžná hlášení o stavu testování, což jsou v případě testera hlášení o průběhu a stavu provádění testů. Při zaznamenávání výsledků a reportování defektů musí tester vyhodnocovat, zda testovací případ označit jako „prošlo","neprošlo", nebo „prošlo s výhradou", jaký je stupeň závažnosti a priority nalezeného defektu a další informace. Je důležité, aby testeři zadávání těchto dat nepodceňovali a zadávali tato data přesně a konzistentně, protože na jejich základě jsou potom počítány hodnoty metrik a monitorován a řízen průběh testování.

Po dokončení provádění testů jsou úlohy testera obdobné jako úlohy test analytika. Pomáhá test manažerovi se shromažďováním metrik z nástrojů pro řízení testování a s posouzením celkového pokrytí a průběhu testů, může být pověřen předáním pracovních produktů testování relevantním osobám a týmům, archivací pracovních produktů a může být přizván jako důležitý zdroj informací na retrospektivní mítinky.

Back to top



Staffing

**Skills**

Dle [RUP, 2010; Patton, 2002, s. 18] by měl mít tester následující schopnosti, vlastnosti a dovednosti:

- Předchozí zkušenost s testováním, technikami testování a nástroji pro podporu testování,
- schopnost rozpoznat problémy a řešit je,
- být taktní, diplomatický a přesvědčivý při hlášení defektů,
- být pečlivý, vytrvalý, zvídavý a věnovat pozornost i detailům,
- znalost testované aplikace, systému nebo oblasti, která je předmětem testování (výhodou),
- znalosti webových aplikací a systémových architektur (výhodou).

U testera se zaměřením na automatizované testy je dále vyžadováno [RUP, 2010]:

- Zkušenosti s nástroji pro automatizaci testů,
- programátorské dovednosti a zkušenosti s debuggingem [1].

Požadavky na dovednosti role testera se, stejně jako u test analytika, mohou lišit v závislosti na typu testů, které jsou prováděny.

---

[1] **Debugging** je dle [ISTQB, 2012c, s. 17] definován jako „proces nalézání, analyzování a odstraňování příčin selhání v softwaru".

**Assignment Approaches**

V malých testovacích týmech je běžnou praxí přiřazení role testera společně s rolí test analytik. V testovacím týmu je potom v obou rolích zároveň více osob se stejnými znalostmi a zodpovědnostmi. Ve větších testovacích týmech mohou být tyto dvě role odděleny. Do role testera se obsazují obvykle méně zkušení členové týmu a mají na starost pouze povinnosti testera. Postupem času získávají četné zkušenosti v oboru testování a mohou přibírat k roli testera i úlohy test analytika. Do separátní role testera bývají také obsazováni testeři specializovaní na automatizaci testů. [RUP, 2010]

Back to top

**Obrázek č. 29 Detail role tester, zdroj: [autor]**

Detail **procesu testování** zachycuje Obrázek č. 30.

## Delivery Process: Proces testování

Description | Work Breakdown Structure | Team Allocation | Work Product Usage

### Work Breakdown

| Breakdown Element | Steps | Index | Predecessors | Model Info | Type |
|---|---|---|---|---|---|
| Plánování testů | | 1 | | | Phase |
| Plánování testů | | 2 | | | Activity |
| Definice cílů a rozsahu testování | | 3 | | | Task Descriptor |
| Analýza rizik testování | | 4 | | | Task Descriptor |
| Definice přístupu k testování | | 5 | | | Task Descriptor |
| Stanovení metrik testování | | 6 | | | Task Descriptor |
| Analýza testů | | 7 | | | Phase |
| Analýza testů | | 8 | | | Activity |
| Analýza podkladů pro testování | | 9 | | | Task Descriptor |
| Identifikace testovacích podmínek | | 10 | | | Task Descriptor |
| Monitoring, reporting a řízení testování | | 11 | | | Activity |
| Monitorování testování | | 12 | | | Task Descriptor |
| Průběžné reportování o stavu testování | | 13 | | | Task Descriptor |
| Řízení testování | | 14 | | | Task Descriptor |
| Návrh testů | | 15 | | | Phase |
| Návrh testů | | 16 | | | Activity |
| Návrh testovacích případů | | 17 | | | Task Descriptor |
| Monitoring, reporting a řízení testování | | 18 | | | Activity |
| Monitorování testování | | 19 | | | Task Descriptor |
| Průběžné reportování o stavu testování | | 20 | | | Task Descriptor |
| Řízení testování | | 21 | | | Task Descriptor |
| Implementace testů | | 22 | | | Phase |
| Implementace testů | | 23 | | | Activity |
| Vytváření testovacích skriptů | | 24 | | | Task Descriptor |
| Příprava testovacích dat | | 25 | | | Task Descriptor |
| Organizace a prioritizace testovacích případů | | 26 | | | Task Descriptor |
| Sestavení harmonogramu provádění testů | | 27 | | | Task Descriptor |
| Ověření připravenosti infrastruktury pro testování | | 28 | | | Task Descriptor |
| Monitoring, reporting a řízení testování | | 29 | | | Activity |
| Monitorování testování | | 30 | | | Task Descriptor |
| Průběžné reportování o stavu testování | | 31 | | | Task Descriptor |
| Řízení testování | | 32 | | | Task Descriptor |
| Provádění testů | | 33 | | | Phase |
| Provádění testů | | 34 | | | Activity |
| Provádění testů a logování výsledků | | 35 | | | Task Descriptor |
| Analýza a reportování defektů | | 36 | | | Task Descriptor |
| Retesty a regresní testování | | 37 | | | Task Descriptor |
| Monitoring, reporting a řízení testování | | 38 | | | Activity |
| Monitorování testování | | 39 | | | Task Descriptor |
| Průběžné reportování o stavu testování | | 40 | | | Task Descriptor |
| Řízení testování | | 41 | | | Task Descriptor |

| | | | | | |
|---|---|---|---|---|---|
| Vyhodnocení testování | | 42 | | | Phase |
| Vyhodnocení testování | | 43 | | | Activity |
| Vyhodnocení výstupních kritérií | | 44 | | | Task Descriptor |
| Vytvoření souhrnného reportu za testování | | 45 | | | Task Descriptor |
| Uzavření testování | | 46 | | | Phase |
| Uzavření testování | | 47 | | | Activity |
| Kontrola kompletnosti testování | | 48 | | | Task Descriptor |
| Předání pracovních produktů z procesu testování | | 49 | | | Task Descriptor |
| Retrospektivní mítinky | | 50 | | | Task Descriptor |
| Archivace pracovních produktů | | 51 | | | Task Descriptor |

**Obrázek č. 30 Detail procesu testování, zdroj: [autor]**