Certifikovaný tester

Učební osnovy pro základní stupeň

Verze 2011 CZ

Historie změn

Verze	Datum	Poznámky
ISTQB 2011 CZ 1.0.0	15.1.2017	Certifikovaný tester Základní stupeň – český překlad Oficiální vydání 1.0.0
ISTQB 2011 CZ RC1	31.12.2016	Certifikovaný tester Základní stupeň – český překlad Aktualizované vydání. Release candidate 1
ISTQB 2011 CZ Beta	15.6.2013	Certifikovaný tester Základní stupeň – český překlad Aktualizované vydání. Beta verze.
ISTQB 2007 CZ Beta 1	14. 8. 2008	Certifikovaný tester Základní stupeň – český překlad Verze Beta 1 – nepublikovaná verze.

Obsah

Poděkování

Pracovní skupina “Základní stupeň” Mezinárodního výboru pro kvalifikaci testování software (International Software Testing Qualifications Board Working Group Foundation Level), vydání 2011: Thomas Müller (předseda), Debra Friedenberg. Klíčový tým děkuje týmu revidujících (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquer, Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal) a všem lokálním výborům za návrhy k současným učebním osnovám.

Pracovní skupina “Základní stupeň” Mezinárodního výboru pro kvalifikaci testování softwaru (International Software Testing Qualifications Board Working Group Foundation Level), vydání 2010: Thomas Müller (předseda), Rahul Verma, Martin Klonk a Armin Beer. Klíčový tým děkuje týmu revidujících (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Tuula Pääkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veendendaal) a všem lokálním výborům za jejich připomínky.

Pracovní skupina “Základní stupeň” Mezinárodního výboru pro kvalifikaci testování softwaru (International Software Testing Qualifications Board Working Group Foundation Level), vydání 2007: Thomas Müller (předseda), Dorothy Graham, Debra Friedenberg a Erik van Veendendaal. Klíčový tým děkuje týmu revidujících (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson a Wonil Kwon) a všem lokálním výborům za návrhy.

Pracovní skupina “Základní stupeň” Mezinárodního výboru pro kvalifikaci testování software (International Software Testing Qualifications Board Working Group Foundation Level), vydání 2005: Thomas Müller (předseda), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson a Erik van Veendendal. Klíčový tým děkuje týmu revidujících a všem lokálním výborům za návrhy.

Překlad do českého jazyka, verze z roku 2007, Czech and Slovak Testing Board (CaSTB): Alexandra Alvarová, Róbert Dankanin, Petr Neugebauer, Jana Podpěrová, Jana Zientková.

Překlad do českého jazyka, vydání 2011, Czech and Slovak Testing Board (CaSTB): David Janota, Martin Malaník, Petr Neugebauer, Lukáš Piška, Jana Podpěrová, Gabor Puhalla, Miroslav Renda a Jana Zientková. Překlad vznikal na dobrovolnické bázi. Byť bylo snahou překladatelů docílit co nejvěrnějšího překladu původního anglického vydání, prostor pro zdokonalování v tak komplexním textu jistě je a stále bude. Postřehy a podněty čtenářů proto rádi uvítáme e-mailem na adrese translation@castb.org.

Úvod k učebním osnovám

Účel dokumentu

Tyto učební osnovy tvoří základ pro mezinárodní kvalifikaci testování softwaru pro základní stupeň. Mezinárodní výbor pro kvalifikace testování softwaru (International Software Testing Qualifications Board, dále již jen ISTQB®) poskytuje lokálním výborům, které akreditují poskytovatele školení a odvozují zkušební otázky v jejich mateřském jazyce. Poskytovatelé školení určují vhodné výukové metody pro akreditaci a vytvářejí učební materiály a pomůcky. Tyto učební osnovy pomáhají kandidátům v přípravě na zkoušku.

Informace o historii a pozadí učebních osnov se nacházejí v Příloze A.

Základní stupeň Certifikovaný tester v testování softwaru

Kvalifikace "Základní stupeň" je určená pro kohokoliv, kdo je určitým způsobem zapojený do testování softwaru. To znamená osoby v rolích jako tester, test analytik, test inženýr, test konzultant, manažer testování, akceptační tester a vývojář softwaru. Kvalifikace základního stupně je též vhodná pro každého, kdo chce porozumět základům testování softwaru, např. pro projektové manažery, manažery kvality, manažery vývoje softwaru, business analytiky, ředitele IT a konzultanty pro management. Držitelé certifikátu základního stupně budou moci pokračovat ve vyšších úrovních kvalifikace.

Studijní cíle / kognitivní úroveň znalosti

Studijní cíle jsou označeny pro každou část těchto učebních osnov a jsou klasifikovány následovně:

- Z1: zapamatovat si,
- Z2: pochopit,
- Z3: použít,
- Z4: analyzovat.

Bližší detaily a příklady studijních cílů jsou uvedené v Příloze B.

Je potřebné zapamatovat si (Z1) také všechny termíny uvedené v části "Základní výrazy" přímo pod názvem kapitoly, ačkoliv nejsou explicitně uvedeny v definovaných cílech.

Zkouška

Certifikační zkouška pro základní stupeň bude založena na těchto učebních osnovách. Odpovědi na zkušební otázky mohou vyžadovat použití více než jedné části osnov. Zkoušet je možné všechny části učebních osnov.

Forma zkoušení je jedna volbaz více možností (single-choice).

Zkoušky mohou být vykonány jako součást akreditovaného školení nebo mohou být vykonány nezávisle (např. ve zkušebním centru nebo na veřejné zkoušce). Absolvování akreditovaného školení není nutným předpokladem pro zkoušku.

Akreditace

Lokální výbor ISTQB může akreditovat poskytovatele školení, jejichž školicí materiály vyhovují těmto učebním osnovám. Lokální výbor nebo osoba vykonávající akreditaci by měla pro poskytovatele školení poskytnout akreditační směrnice.
Akreditované školení je uznáno jako odpovídající těmto osnovám a může obsahovat ISTQB zkoušku jako část školení.

Další rady pro poskytovatele školení se nacházejí v Příloze D.

Úroveň detailu

Úroveň detailu v těchto učebních osnovách umožňuje mezinárodně konzistentní vyučování a zkoušení. Aby bylo možné dosáhnout tohoto cíle, učební osnovy obsahují:

- Všeobecné instruktážní cíle popisující záměry základního stupně.
- Seznam informací pro výuku, včetně popisu a odkazů na další zdroje, jsou-li vyžadovány.
- Studijní cíle pro každou oblast znalostí; ty popisují požadovaný výsledek studia a způsob uvažování, kterých je potřebné dosáhnout.
- Seznam výrazů, které si student musí zapamatovat a kterým musí rozumět.
- Popis klíčových konceptů k výuce, včetně zdrojů jako schválená literatura nebo standardy.

Obsahem těchto učebních osnov není popis celé znalostní oblasti testování softwaru. Odráží úroveň detailu, který má být pokryt ve školeních pro základní stupeň.

Jak jsou tyto učební osnovy uspořádány

Osnovy mají šest hlavních kapitol. Nejvyšší úroveň nadpisu pro každou kapitolu ukazuje nejvyšší úroveň studijních cílů, které jsou v rámci kapitoly pokryty, a specifikuje čas potřebný k výkladu kapitoly. Například:

2. Testování v životním cyklu softwaru (Z2)	115 minut

Tento nadpis říká, že kapitola 2 má studijní cíl Z1 (jehož splnění je předpokládáno, neboť je zobrazen vyšší cíl Z2) a Z2 (ale ne Z3) a délka výuky je určena na 115 minut na vysvětlení kapitoly. Uvnitř každé kapitoly jsou různé počty sekcí. Každá sekce má též studijní cíl a požadovaný čas. Podsekce, které nemají určený čas, jsou zahrnuty v rámci času pro sekci.

1. Základy testování (Z2) — 155 minut

Studijní cíle pro základy testování

Tyto cíle určují, co budete umět po dokončení každého modulu.

1.1 Proč je potřebné testování? (Z2)

SC-1.1.1 Popsat způsoby, a to včetně příkladů, jakými může defekt v softwaru způsobit škodu osobě, prostředí nebo firmě. (Z2)
SC-1.1.2 Rozlišovat mezi prvotní příčinou defektu a jeho následky. (Z2)
SC-1.1.3 Pomocí příkladů zdůvodnit, proč je testování potřebné. (Z2)
SC-1.1.4 Popsat, proč je testování součástí zabezpečení kvality, a uvést příklady, jak testování přispívá k vyšší kvalitě. (Z2)
SC-1.1.5 Pomocí příkladů vysvětlit a porovnat pojmy omyl, defekt, vada , selhání a související pojem bug. (Z2)

1.2 Co je testování? (Z2)

SC-1.2.1 Zapamatovat si všeobecné cíle testování. (Z1)
SC-1.2.2 Uvést příklady cílů testování v různých fázích životního cyklu softwaru. (Z2)
SC-1.2.3 Odlišit testování a ladění. (Z2)

1.3 Sedm principů testování (Z2)

SC-1.3.1 Vysvětlit sedm principů testování. (Z2)

1.4 Základní testovací proces (Z1)

SC-1.4.1 Zapamatovat si pět základních testovacích aktivit a příslušných úkolů - plánováním počínaje, uzavřením konče. (Z1)

1.5 Psychologie testování (Z2)

SC-1.5.1 Zapamatovat si psychologické faktory, které ovlivňují úspěch testování. (Z1)
SC-1.5.2 Posoudit rozdílné postoje testera a vývojáře. (Z2)

1.1 *Proč je potřebné testování (Z2)*	*20 minut*

Základní výrazy
Bug, defekt, omyl, selhání, vada, chyba, kvalita, riziko.

1.1.1 Souvislosti v softwarových systémech (Z1)

Softwarové systémy jsou neoddělitelnou součástí života, od obchodních aplikací (např. bankovnictví) až po spotřebitelské produkty (např. automobily). Většina lidí má již zkušenosti se softwarem, který nepracoval podle očekávání. Software, který nepracuje správně, může způsobit mnoho problémů včetně ztráty peněz, času nebo obchodní reputace. Může dokonce způsobit zranění nebo smrt.

1.1.2 Příčiny softwarových defektů (Z2)

Člověk se může dopustit omylu (chyby), který způsobí defekt (vadu, bug) v programovém kódu nebo v dokumentaci. Když se vykoná kód obsahující defekt, systém nemusí udělat, co by udělat měl (resp. udělá něco, co by neměl) a tím způsobí selhání. Defekty v softwaru, systémech nebo v dokumentaci mohou způsobit selhání, ale u všech defektů tomu tak být nemusí.

K defektům dochází z toho důvodu, že lidé jsou omylní a protože pracují pod časovým tlakem, se složitým kódem, v komplikované infrastruktuře, s měnícími se technologiemi, a/nebo kvůli vzájemnému působení systémů.

Selhání mohou být též zapříčiněna přírodními podmínkami prostředí. Například radiace, magnetismus, elektrické pole a znečištění mohou způsobit chyby ve firmwaru nebo ovlivnit vykonávání softwaru změnou hardwarových podmínek.

1.1.3 Úloha testování ve vývoji softwaru, údržbě a provozu (Z2)

Důsledné testování softwaru a dokumentace mohou pomoci snížit riziko problémů, které nastanou v průběhu provozu. Tím přispívá ke kvalitě softwarového systému, zejména jestliže jsou zjištěné defekty opraveny dřív, než je systém uvolněn do provozu.

Testování softwaru může být vyžadováno též z důvodu splnění smluvních nebo právních požadavků, nebo specifických průmyslových standardů.

1.1.4 Testování a kvalita (Z2)

Pomocí testování je možné měřit kvalitu softwaru ve smyslu zjištěných defektů, a to pro funkcionální a také pro nefunkcionální softwarové požadavky a charakteristiky (např. spolehlivost, použitelnost, účinnost, udržovatelnost a přenositelnost). Pro další informace o nefunkcionálním testování viz kapitola 2; pro další informace o charakteristikách softwaru – viz “Softwarové inženýrství – Kvalita softwarového produktu” (ISO 9126).

Testování může zvýšit důvěru v kvalitu softwaru, pokud najde malé množství defektů nebo žádné defekty. Vhodně navržený test, který úspěšně projde, snižuje celkovou úroveň rizika v systému. Pokud testování prokáže defekty, kvalita softwarového systému se zvýší, jsou-li tyto defekty opraveny.

Z předešlých projektů by mělo dojít k ponaučení. Při pochopení prvotních příčin defektů zjištěných v jiných projektech mohou být zlepšeny procesy, které by následně měly zabránit zopakování těchto defektů, následkem čehož by se měla zlepšit kvalita budoucích systémů. Toto je charakteristika oblasti nazývané zajištění kvality.

Testování by mělo být integrováno jako jedna z aktivit zabezpečení kvality (tj. spolu s vývojářskými standardy, školeními a analýzami defektů).

1.1.5 Kdy je testování dostatečné? (Z2)

Rozhodnutí, kdy je testování dostatečné, by mělo vzít v úvahu míru rizika, včetně technických, bezpečnostních a obchodních rizik, a také projektových omezení, jako je čas a rozpočet. (Riziko je diskutováno podrobněji v kapitole 5.)

Testování by mělo poskytnout dostatek informací zainteresovaným osobám pro kvalifikované rozhodnutí o uvolnění testovaného softwaru nebo systému do následující fáze vývoje nebo o předání zákazníkům.

1.2 *Co je testování? (Z2)*	*30 minut*

Základní výrazy
Ladění, požadavek, revize, testovací případ, testování, cíl testování.

Pozadí
Obecně je testování vnímáno tak, že se jedná pouze o běh testů, tj. spouštění softwaru. To je jedna ze součástí testování, nikoliv však jediná.

Testovací aktivity existují také před a po prováděním (spouštěním) testů. Tyto aktivity zahrnují plánování a řízení, výběr testovacích podmínek, navržení testovacích případů a kontrolu výsledků, vyhodnocování výstupních kritérií, reportování testovacího procesu a testovaného systému a dokončení finalizačních aktivit (aktivit po ukončení fáze testování). Testování též zahrnuje revidování dokumentů (včetně zdrojového kódu) a statickou analýzu.

Statické i dynamické testování může být použito k dosažení podobných cílů. Obě metody poskytují informace, které mohou být použity ke zlepšení testovaného systému a procesů vývoje a testování.

Testování může mít následující cíle:
- nalezení defektů;
- získání důvěry s ohledem na úroveň kvality;
- poskytnutí informací pro rozhodování;
- předcházení defektům.

Myšlenkové procesy a činnosti spojené s návrhem testů zavedené dostatečně včas v životním cyklu (ověření testovací báze prostřednictvím návrhu testů) mohou pomoci zabránit zanesení defektů do kódu. Revidování dokumentů (např. požadavků), identifikace a řešení problémů taktéž pomáhá prevenci defektů, které by se objevily v kódu.

Různé pohledy v testování zohledňují různé cíle. Například ve vývojářském testování (např. testování komponent, integračním a systémovém testování), může být hlavním cílem vyvolání tolika selhání, jak je jen možné s cílem identifikace a opravy defektů. Při akceptačním testování může být hlavním cílem potvrzení, že systém pracuje podle očekávání, abychom tak získali důvěru, že je v souladu s požadavky. V některých případech může být hlavním cílem testování ohodnocení kvality softwaru (bez snahy opravovat defekty), poskytnutí informace zainteresovaným osobám o rizicích uvolnění systému v daném čase. Testování údržby často zahrnuje testování s cílem ověřit, zda nebyly v průběhu vývoje změn zaneseny další defekty. Během provozního testování může být hlavním cílem zhodnocení charakteristik systému, jako je spolehlivost nebo dostupnost.

Ladění a testování jsou navzájem odlišné činnosti. Dynamické testování může ukázat selhání, která jsou způsobena defekty. Ladění je vývojová aktivita, která nachází, analyzuje a odstraňuje příčinu selhání. Následné přetestování testerem slouží k ujištění se, že oprava skutečně řeší selhání. Zodpovědnost za tyto činnosti je obvykle taková, že testeři testují a vývojáři ladí.

Proces testování a jeho aktivit je popsán v části 1.4.

1.3 *Sedm principů testování (Z2)*	*35 minut*

Základní výrazy
Úplné testování.

Principy
Za posledních 40 let bylo formulováno několik principů testování, které poskytují všeobecné pokyny společné pro všechna testování.

Princip 1 – Testování ukazuje přítomnost defektů
Testování může ukázat, že jsou defekty přítomny, ale nemůže dokázat, že v softwaru žádné defekty nejsou. Testování snižuje pravděpodobnost, že v softwaru zůstanou neobjevené defekty, avšak nenalezení žádného defektu stále není důkaz bezchybnosti.

Princip 2 – Úplné testování je nemožné
Testování všeho (všech kombinací vstupů a vstupních podmínek) není realizovatelné s výjimkou triviálních případů. Namísto úplného testování by měly být k určení hlavního předmětu testovacího úsilí použity analýza rizik a stanovení priorit.

Princip 3 – Včasné testování
Pro včasné nacházení defektů musí testovací aktivity začít v rámci životního cyklu vývoje softwaru nebo systému co nejdříve, kdy je to možné, a musí být zaměřeny na definované cíle.

Princip 4 – Shlukování defektů
Testování musí být zaměřené proporčně na očekávanou a později zjištěnou hustotu defektů v modulech. Velmi malé množství modulů obvykle obsahuje většinu defektů zjištěných v průběhu testování, před uvolněním, nebo je zodpovědné za nejvíce provozních selhání.

Princip 5 – Pesticidový paradox
Jsou-li stále opakovány tytéž testy, časem stejný soubor testovacích případů nenalezne žádné nové defekty. K překonání tohoto „pesticidového paradoxu“ je potřeba existující testovací případy pravidelně revidovat a upravovat. Zároveň je potřeba napsat nové, odlišné testy k prověření jiných částí softwaru nebo systému pro případné odhalení dalších defektů.

Princip 6 – Testování je závislé na kontextu
Testování je vykonáváno odlišně v různých kontextech. Například software kritický z pohledu bezpečnosti se testuje jiným způsobem než webové stránky elektronického obchodu.

Princip 7 – Falešná představa o neexistenci omylů
Nalezení a opravení defektů nepomůže, pokud je vytvořený systém nepoužitelný a nesplňuje potřeby a očekávaní uživatelů.

1.4 *Základní testovací proces (Z1)*	*35 minut*

Základní výrazy

Konfirmační testování, přetestování, výstupní kritéria, incident, regresní testování, testovací báze, testovací podmínka, pokrytí testování, testovací data, provedení testů, záznam testování, plán testování, testovací procedura, pravidla testování, testovací sada, souhrnná zpráva z testování , testware.

Pozadí

Nejvíce viditelnou částí testování je provedení testů. Aby bylo efektivní a účinné, měly by plány testování také obsahovat čas na plánování testů, návrh testovacích případů, přípravu pro provedení testů a vyhodnocování výsledků.

Základní testovací proces sestává z následujících hlavních aktivit:

- Plánování a řízení testování;
- Analýza a návrh testování;
- Implementace a vykonávání testování;
- Vyhodnocení výstupních kritérií a reportování;
- Činnosti související s ukončením testování.

Ačkoliv jsou aktivity řazeny logicky za sebou, mohou se v procesu překrývat nebo mohou být vykonávány souběžně. Obvykle se vyžaduje, aby byly tyto hlavní aktivity přizpůsobovány kontextu a systému.

1.4.1 Plánování a řízení testování (Z1)

Plánování testování je aktivita definování cílů testování a specifikace testovacích činností s cílem dosáhnout cíle a poslání.

Řízení testování je průběžná aktivita porovnávající aktuální vývoj oproti plánu a reportování stavu, včetně odchylek od plánu. Zahrnuje realizaci činností nezbytných k dosažení poslání a cílů projektu. Aby mohlo být testování řízené, činnosti testování by měly být v průběhu projektu sledovány. Plánování testování bere v úvahu také zpětnou vazbu z monitorovacích a řídících aktivit.

Úkoly plánování a řízení testování jsou definovány v kapitole 5 této učební osnovy.

1.4.2 Analýza a návrh testování (Z1)

Analýza a návrh testování jsou činnosti, během kterých jsou všeobecné cíle testování transformovány do konkrétních testovacích podmínek a testovacích případů.

Analýza a návrh testování zahrnují následující hlavní činnosti:

- Revidování základu testování (jako jsou požadavky, úroveň integrity softwaru[1] (úroveň rizika), reporty analýzy rizik, architektura, návrh, specifikace rozhraní).
- Vyhodnocení testovatelnosti základu testování a objektů testování.
- Identifikování a stanovení priorit testovacích podmínek, které je založeno na analýze testovacích položek, specifikace, chování a struktuře.
- Navržení a stanovení priorit vysokoúrovňových (high-level) testovacích případů.

[1] Stupeň, který software splňuje, nebo musí splňovat s ohledem na sadu softwaru zvolenou zainteresovanými osobami, a/nebo stupeň systémových charakteristik softwaru (např. složitost softwaru, vyhodnocení rizik, úroveň bezpečnosti, úroveň zabezpečení, požadovaná výkonnost, spolehlivost nebo cena), které jsou definovány, aby odrážely důležitost softwaru pro jeho zainteresované osoby.

- Identifikace potřebných testovacích dat podporujících testovací podmínky a testovací případy.
- Návrh nastavení testovacího prostředí a identifikování požadované infrastruktury a nástrojů.
- Vytvoření obousměrné trasovatelnosti mezi testovací bází a testovacími případy.

1.4.3 Implementace a provedení testů (Z1)

Implementace a provedení testů je činnost, během které jsou specifikovány testovací procedury nebo skripty, a to vhodnou kombinací testovacích případů v určitém pořadí, a zahrnutím všech dalších informací potřebných pro provedení testů. Poté dochází k nastavení testovacího prostředí a testy jsou spuštěny.

Implementace a provedení testů zahrnuje následující hlavní činnosti:

- Finalizaci, implementaci a stanovení priorit testovacích případů (včetně určení testovacích dat).
- Vytvoření a stanovení priorit testovacích procedur, vytvoření testovacích dat a volitelně příprava testovacích frameworků a psaní automatizovaných testovacích skriptů.
- Vytvoření sestav testů z testovacích procedur pro účinné vykonávání testů.
- Ověření, zda je testovací prostředí nastaveno korektně.
- Ověření a aktualizaci obousměrné trasovatelnosti mezi testovací bází a testovacími případy.
- Vykonání testovacích procedur, ať již manuálně nebo s použitím nástrojů pro provedení testů, dle plánovaného pořadí.
- Zaznamenávání výsledků provedených testů a zaznamenávání identifikátorů a verzí testovaného softwaru, testovacích nástrojů a testwaru.
- Porovnávání skutečných výsledků s očekávanými výsledky.
- Reportování neshod jako incidentů a jejich analýza za účelem stanovení jejich příčiny (např. defekt v kódu, v daných testovacích datech, v testovacím dokumentu nebo chyba ve způsobu, jakým byl test proveden).
- Opakování testovacích aktivit jako výsledek přijetí opatření pro každou nesrovnalost. Například opětovné provedení testu, který předtím skončil s chybou, s cílem potvrzení opravy (konfirmační testování), provedení opraveného testu a/nebo provedení testů s cílem ujistit se, že do nezměněných oblastí softwaru nebyly zaneseny defekty nebo že oprava defektu neodhalila další defekty (regresní testování).

1.4.4 Vyhodnocení výstupních kritérií a reportování (Z1)

Vyhodnocení výstupních kritérií je aktivita, ve které je provedení testů hodnoceno vůči definovaným cílům. Mělo by být prováděno pro každou úroveň testování (viz kapitola 2.2).

Vyhodnocení výstupních kritérií má následující hlavní činnosti:

- Kontrolu záznamů testování vůči výstupním kritériím specifikovaným během plánování testování.
- Zhodnocení, zda jsou potřebné další testy, nebo zda by měla být specifikovaná výstupní kritéria změněna.
- Sepsání souhrnné zprávy z testování pro zainteresované osoby.

1.4.5 Aktivity uzavření testu (Z1)

Aktivity uzavření testu shromažďují data z ukončených testovacích aktivit za účelem konsolidace zkušeností, testwaru, faktů a údajů. K aktivitám uzavření testu dochází v rámci projektových milníků, například po uvolnění softwarového systému, po ukončení testovacího projektu (nebo jeho zrušení), po dosažení milníku nebo když byl ukončen opravný release.

Aktivity uzavření testu zahrnují následující hlavní činnosti:

- Kontrolu toho, které plánované dodávky byly doručeny.
- Uzavření záznamů o incidentech nebo nahlášení záznamů o změně pro ty, které zůstaly otevřené.

- Dokumentaci akceptace systému.
- Finalizaci a archivování testwaru, testovacího prostředí a testovací infrastruktury pro pozdější opětovné použití.
- Odevzdání testwaru organizaci zajišťující provoz.
- Analýzu získaných ponaučení na určení změn potřebných pro budoucí releasy a projekty.

1.5 *Psychologie testování (Z2)*	*25 minut*

Základní výrazy
Odhadování omylů, nezávislost.

Pozadí
Přístup, který má být použit v průběhu testování a revidování, se liší od přístupu, který se používá u vývoje softwaru. Při správném přístupu jsou vývojáři schopní testovat svůj vlastní kód, ale přenesení této zodpovědnosti na testera se většinou děje s cílem soustředit úsilí a získat další benefity, jako je nezávislý pohled školených a profesionálních zdrojů z oblasti testování. Nezávislé testování může být realizováno na kterékoliv úrovni testování.

Určitý stupeň nezávislosti (čímž se vyhneme zaujatosti autora) často umožní testerovi být více efektivní při hledání defektů a selhání. Avšak nezávislost není náhradou za znalost (systému) - vývojáři mohou účinně najít mnoho defektů ve svém vlastním kódu. Může být definováno několik stupňů nezávislosti, jak je uvedené níže, od nejnižšího stupně po nejvyšší:

- Testy navržené osobou (osobami), která(é) vytváří testovaný software (nízký stupeň nezávislosti).
- Testy navržené jinou osobou (osobami, např. z vývojářského týmu).
- Testy navržené osobou (osobami) z jiné organizační skupiny (např. nezávislý testovací tým) nebo specialisty na testování (např. specialisty na testování použitelnosti nebo výkonnosti).
- Testy navržené osobou (osobami) z jiné organizace nebo společnosti (tj. outsourcing nebo certifikace externí institucí).

Lidé a projekty se řídí svými cíli. Lidé mají tendenci srovnávat své plány s cíli stanovenými managementem nebo jinými zainteresovanými osobami, například s cílem najít defekty nebo potvrdit, že software funguje. Proto je důležité jasné stanovení cílů testování.

Objevování selhání v průběhu testování může být vnímáno jako kritika vůči produktu a vůči autorovi. Testování je z tohoto důvodu často vnímáno jako destruktivní aktivita, ačkoliv je velice konstruktivní z pohledu řízení produktových rizik. Hledání selhání v aplikaci vyžaduje zvídavost, profesionální pesimismus, kritický pohled, smysl pro detail, dobrou komunikaci s partnery vývojáři a zkušenost, na základě které je možno odhadovat omyly.

Když jsou omyly, defekty nebo selhání sdělovány konstruktivním způsobem, je možné vyhnout se špatným vztahům mezi testery a analytiky, návrháři a vývojáři. Toto platí pro defekty nalezené v průběhu revize a rovněž pro testování.

Tester a vedoucí testování potřebují dobré mezilidské dovednosti, aby byli schopni komunikovat konstruktivním způsobem faktické informace o defektech, pokroku a rizicích. Autorovi softwaru nebo dokumentu může informace o defektu pomoci zlepšit jeho dovednosti. Defekty zjištěné a opravené v průběhu testování ušetří později čas a peníze, a snižují pravděpodobnost rizika.

Komunikační problémy mohou nastat hlavně tehdy, když jsou testeři vnímáni jen jako hlasatelé nechtěných zpráv o defektech. Existuje však několik cest, jak zlepšit komunikaci a vztahy mezi testery a ostatními členy:

- Preferovat spolupráci před soubojem – připomenout každému společný cíl lepší kvality systémů.
- Komunikovat zjištění o produktu nestranným, věcně orientovaným způsobem, bez kritiky zodpovědné osoby, například psát objektivní a faktické zprávy o incidentech a zjištěních z revizí.
- Pokusit se porozumět tomu, jak se cítí jiné osoby, a proč je jejich reakce taková, jaká je.
- Ujistit se, že druhá osoba porozuměla tomu, co jste řekli, a naopak.

1.6 *Etický kodex*	*10 minut*

Zapojení do testování softwaru umožnuje jednotlivcům, aby se dozvídali o důvěrných a chráněných informacích. Etický kodex je potřebný, mimo jiné na zajištění toho, aby tyto informace nebyly používané nevhodně. Uznávajíc etické kodexy ACM a IEEE pro inženýry, uvádí ISTQB® následující etický kodex:

VEŘEJNOST – Certifikovaní testeři softwaru musí konat v souladu s veřejným zájmem.

KLIENT A ZAMĚSTNAVATEL – Certifikovaní testeři softwaru musí konat způsobem, který je v nejlepším zájmu jejich klientů a zaměstnavatele, v souladu s veřejným zájmem.

PRODUKT – Certifikovaní testeři softwaru musí zabezpečit to, že výsledky, které poskytují (pro produkty a systémy, které testují) splňují nejvyšší možné profesionální standardy.

SOUDNOST - Certifikovaní testeři softwaru si musí zachovávat bezúhonnost a nezávislost ve svém profesionálním rozhodování.

MANAGEMENT – Certifikovaní manažeři testování softwaru a vedoucí se musí hlásit k etickému přístupu k managementu testování softwaru a propagovat ho.

PROFESE – Certifikovaní testeři softwaru musí rozšiřovat důvěryhodnost a dobrou pověst své profese v souladu s veřejným zájmem.

KOLEGOVÉ – Certifikovaní testeři softwaru musí být ke svým kolegům spravedliví, podporovat je a napomáhat ve spolupráci s vývojáři softwaru.

MOTIVACE– Certifikovaní testeři softwaru se musí podílet na celoživotním vzdělávání v rámci praxe v jejich oblasti a musí prosazovat etický přístup k výkonu povolání.

Reference

1.1.5 Black, 2001, Kaner, 2002
1.2 Beizer, 1990, Black, 2001, Myers, 1979
1.3 Beizer, 1990, Hetzel, 1988, Myers, 1979
1.4 Hetzel, 1988
1.4.5 Black, 2001, Craig, 2002
1.5 Black, 2001, Hetzel, 1988

2. Testování v životním cyklu softwaru (Z2) — 115 minut

Studijní cíle pro testování v jednotlivých fázích životního cyklu softwaru

Tyto cíle určují, co budete umět po dokončení každého modulu.

2.1 Modely vývoje softwaru (Z2)

SC-2.1.1 Vysvětlit vztah mezi vývojem, testovacími aktivitami a pracovními produkty v životním cyklu vývoje použitím příkladů projektových a produktových typů. (Z2)
SC-2.1.2 Pochopit fakt, že modely vývoje softwaru musí být přizpůsobeny kontextu projektu a produktovým charakteristikám. (Z1)
SC-2.1.3 Zapamatovat si charakteristiky správného testování, které jsou použitelné v jakémkoliv modelu životního cyklu. (Z1)

2.2 Úrovně testování (Z2)

SC-2.2.1 Porovnat různé úrovně testování: hlavní cíle, typické objekty testování, typické cíle testování (např. funkcionální nebo strukturální) a související pracovní produkty, osoby, které testují, typy defektů a selhání, které mají být identifikovány. (Z2)

2.3 Typy testů (Z2)

SC-2.3.1 Porovnat čtyři typy testování softwaru (funkcionální, nefunkcionální, strukturální, testování změn) na základě příkladu. (Z2)
SC-2.3.2 Pochopit, že strukturální testy se vyskytují na kterékoliv úrovni testování. (Z1)
SC-2.3.3 Identifikovat a popsat nefunkcionální typy testování založené na nefunkcionálních požadavcích. (Z2)
SC-2.3.4 Identifikovat a popsat typy testování založené na analýze struktury nebo architektury softwarového systému. (Z2)
SC-2.3.5 Popsat účel konfirmačního testování a regresního testování. (Z2)

2.4 Testování údržby (Z2)

SC-2.4.1 Porovnat testování údržby (testování existujícího systému) s testováním nové aplikace s ohledem na typy testů, důvody spuštění testu a objem testování. (Z2)
SC-2.4.2 Identifikovat indikátory pro testování údržby (modifikace, migrace a vyřazení). (Z1)
SC-2.4.3 Popsat roli regresního testování a dopadové analýzy v oblasti údržby. (Z2)

2.1 *Modely vývoje softwaru (Z2)*	*20 minut*

Základní výrazy
Krabicový software, iterativně-inkrementální vývojový model, validace, ověření, V-model.

Pozadí
Izolované testování neexistuje; testovací aktivity se vztahují k aktivitám vývoje softwaru. Rozdílné modely životního cyklu vývoje vyžadují rozdílné přístupy k testování.

2.1.1 V-model (sekvenční vývojový model) (Z2)

I když existují různé varianty V-modelu, běžný typ V-modelu používá čtyři úrovně testování korespondující se čtyřmi úrovněmi vývoje. Čtyři úrovně používané v těchto učebních osnovách jsou:

- testování komponent (jednotek);
- integrační testování;
- systémové testování;
- akceptační testování.

Ve skutečnosti může mít V-model více nebo méně úrovní resp. rozdílné úrovně vývoje a testování v závislosti na projektu a softwarovém produktu. Například po testování komponent může následovat integrační testování komponent a po systémovém testování systémové integrační testování.

Softwarové pracovní produkty vytvořené v průběhu vývoje (jako například business scénáře nebo případy použití, specifikace požadavků, dokumenty návrhu a kód) jsou často základem pro testování na jedné nebo více úrovních testování. Reference na všeobecné pracovní produkty zahrnuje Capability Maturity Model Integration (CMMI) nebo 'Procesy životního cyklu vývoje softwaru' (IEEE/IEC 12207). Ověření a validace (a včasný návrh testů) mohou být vykonány v průběhu vývoje softwarových pracovních produktů.

2.1.2 Iterativně-inkrementální vývojové modely (Z2)

Iterativně-inkrementální vývoj je proces zavedení požadavků, návrh, tvorby a testování systému, vykonaný jako série kratších vývojových cyklů. Příklady jsou: prototypování, přístup RAD (Rapid Application Development), RUP (Rational Unified Process) a agilní vývojové modely. Systém, který je produktem těchto modelů, je možno testovat na více úrovních testování v průběhu každé iterace. Inkrement přidaný k jiným, dříve vyvinutým, formuje rostoucí neúplný systém, který by měl být taktéž testován. Regresní testování je stále důležitější ve všech následných iteracích. Ověření a validace mohou být vykonány pro každý inkrement.

2.1.3 Testování v modelu životního cyklu (Z2)

V jakémkoliv modelu životního cyklu je několik charakteristik správného testování:

- Pro každou vývojovou aktivitu existuje odpovídající testovací aktivita.
- Každá úroveň testování má cíle testování specifické pro tuto úroveň.
- Analýza a návrh testů pro danou úroveň testování by měly začít v průběhu odpovídající vývojové aktivity.
- Testeři by měli být zahrnuti do revidování dokumentů hned v okamžiku, kdy jsou k dispozici pracovní verze v životním cyklu vývoje.

Úrovně testování mohou být kombinovány nebo reorganizovány v závislosti na charakteru projektu nebo architektuře systému. Například při integraci krabicového softwaru do systému může kupující vykonat integrační testování na systémové úrovni (např. integrace do infrastruktury a jiných systémů

nebo nasazení systému) a akceptační testování (funkcionální a/nebo nefunkcionální; a uživatelské a/nebo provozní testování).

2.2 *Úrovně testování (Z2)*	*40 minut*

Základní výrazy

Alfa testování, beta testování, testování komponent, ovladač, testování v poli, funkcionální požadavek, integrace, integrační testování, nefunkcionální požadavek, testování robustnosti, stub, systémové testování, testovací prostředí, úroveň testování, vývoj řízený testováním, uživatelské akceptační testování.

Pozadí

Pro každou z úrovní testování může být identifikováno následující: všeobecné cíle, pracovní produkt(y), na který(é) se odkazuje při odvozování testovacích případů (tj. testovací báze), objekt testování (tj. co je testováno), typické defekty a selhání, které mohou být nalezeny, požadavky a nástroj pro testovací prostředky (test harness) a podporu nástrojů, specifické přístupy a zodpovědnosti.

Testování konfiguračních dat systému musí být zváženo v průběhu plánování testů.

2.2.1 Testování komponent (Z2)

Testovací báze:

- Požadavky na komponenty.
- Detailní návrh.
- Kód.

Typické objekty testování:

- Komponenty.
- Programy.
- Konverze dat / migrační programy.
- Databázové moduly.

Testování komponent (též známé jako jednotkové testování, testování modulu anebo testování programu) hledá defekty uvnitř softwarových komponent a verifikuje fungování softwarových komponent, modulů, programů, objektů, tříd, atd., které jsou testovatelné samostatně. Může být vykonáno v izolaci od zbytku systému v závislosti na kontextu životního cyklu vývoje a systému. Při testování komponent mohou být použity stuby, ovladače a simulátory.

Testování komponent může zahrnovat testování funkcionality a specifických charakteristik jako jsou chování zdrojů (např. hledání přetečení paměti) anebo testování robustnosti, jakož i strukturální testování (např. pokrytí rozhodnutí). Testovací případy jsou odvozeny z pracovních produktů, kterými jsou specifikace komponenty, návrh softwaru anebo datový model.

Typicky se testování komponent realizuje s přístupem ke kódu, který je testován, a s podporou vývojového prostředí, například framework jednotkového testování anebo nástroj pro ladění. V praxi je testování komponent obvykle vykonáváno za účasti programátora, který kód napsal. Defekty jsou nejčastěji opravovány hned, jak jsou nalezeny, bez formální správy defektů.

Jedním z přístupů v testování komponent je příprava a automatizace testovacích případů před vlastním kódováním. Nazývá se testování na prvním místě anebo vývoj řízený testováním. Tento přístup je vysoce iterativní a je založený na cyklech, ve kterých se vyvíjejí testovací případy, následně se vytvářejí a integrují malé části kódu a vykonávají se testy komponent a opravují se všechny problémy. Toto se opakuje až do doby, dokud nejsou tyto testy úspěšné.

2.2.2 Integrační testování (Z2)

Testovací báze:

- Návrh softwaru a systému.
- Architektura.
- Pracovní toky (workflows).
- Případy užití.

Typické objekty testování:

- Podsystémy.
- Implementace databází.
- Infrastruktura.
- Rozhraní.
- Konfigurace systému a konfigurační data.

Integrační testování prověřuje rozhraní mezi komponentami, interakce s různými částmi systému jako jsou operační systém, souborový systém, hardware a rozhraní mezi systémy.

Může existovat víc než jedna úroveň integračního testování a integrační testování může být vykonáno na testovaných objektech různé velikosti následovně:

1. Integrační testování komponent prověřuje interakci mezi komponentami softwaru a je vykonáváno po testování komponent.
2. Systémové integrační testování prověřuje interakci mezi různými systémy nebo mezi softwarem a hardwarem a může být vykonáno po systémovém testování. V takovém případě však vyvíjející organizace může kontrolovat pouze jednu stranu rozhraní, takže případné změny by mohly narušit stabilitu systému. To může být považováno za riziko. Business procesy implementované jako workflow mohou zahrnovat série systémů, proto mohou být problémy na úrovni více platforem závažné.

S větším záběrem integrace je stále složitější izolovat defekty na specifickou komponentu nebo systém, což může vést ke zvýšení rizika a dodatečného času pro řešení problémů.

Systematické integrační strategie mohou být založené na architektuře systému (jako shora-dolů, zdola-nahoru), funkcionálních úlohách, sekvencích zpracování transakcí nebo na jiných aspektech systému nebo komponenty. Aby bylo jednodušší izolovat vady a odhalit defekty včas, měla by být integrace realizována jako inkrementální změna namísto “velkého třesku”.

V integračním testování může být zahrnuto testování specifických nefunkcionálních charakteristik (např. výkonnosti), a stejně tak i funkcionální testování.

V každém stádiu integrace se testeři zaměřují výhradně na samotnou integraci. Když například integrují modul A s modulem B, zajímají se o testování komunikace mezi moduly, nikoliv však o funkcionalitu samostatného modulu, jelikož ta byla testována v rámci testování komponent. Mohou být použity oba přístupy: funkcionální i strukturální.

V ideálním případě by testeři měli rozumět architektuře a měli by mít možnost ovlivňovat integrační plánování. Když jsou integrační testy plánovány předtím, než jsou vytvořeny komponenty nebo systémy, mohou být vytvořeny v takovém pořadí, aby bylo testování co nejefektivnější.

2.2.3 Systémové testování (Z2)

Testovací báze:

- Specifikace požadavků systému a softwaru.
- Případy užití.

- Funkcionálníspecifikace.
- Reporty analýzy rizik.

Typické objekty testování:
- Systémové, uživatelské a operační manuály.
- Konfigurace systému a konfigurační data.

Systémové testování se zabývá chováním celého systému/produktu. Rozsah testu musí být jasně vymezen v hlavním a/nebo úrovňovém testovacím plánu pro danou úroveň testování.

Při systémovém testování by mělo testovací prostředí v co největší možné míře korespondovat s finálním cílovým nebo provozním prostředím za účelem minimalizace rizika, že selhání, která jsou specifická pro dané prostředí, nebudou při testování nalezena.

Systémové testování může zahrnovat testy založené na rizicích a/nebo na specifikacích požadavků, business procesech, případech užití nebo jiných vysokoúrovňových popisech anebo modelech chování systému, interakcích s operačním systémem a na systémových zdrojích.

Systémové testování by mělo prozkoumat funkcionální i nefunkcionální požadavky systému a charakteristiky kvality dat. Testeři se taktéž musí vypořádat s nekompletními nebo nedokumentovanými požadavky. Systémové testování funkcionálních požadavků začíná používáním nejvhodnějších technik založených na specifikaci (černá skříňka) zohledňující povahu systému, který bude testován. Například může být vytvořena rozhodovací tabulka pro kombinace následků popsaných v business pravidlech. Techniky založené na struktuře (bílá skříňka) mohou být poté použity pro zhodnocení důkladnosti testování s ohledem na strukturální elementy, jako například struktura menu nebo navigace webové stránky (viz kapitola 4.).

Systémové testování často realizuje nezávislý testovací tým.

2.2.4 Akceptační testování (Z2)

Testovací báze:
- Uživatelské požadavky.
- Systémové požadavky.
- Případy užití.
- Business procesy.
- Reporty analýzy rizik.

Typické objekty testování:
- Business procesy na plně integrovaném systému.
- Provozní a údržbové procesy.
- Uživatelské postupy.
- Formuláře.
- Reporty.
- Konfigurační data.

Akceptační testování je často zodpovědností zákazníků nebo uživatelů systému; rovněž do něj mohou být zapojeny i další zainteresované osoby.

Při akceptačním testování je cílem upevnění důvěry v systém, jeho části nebo specifické nefunkcionální charakteristiky systému. Při akceptačním testování není hlavním účelem nalezení defektů. Akceptační testování může vyhodnotit připravenost systému pro nasazení a používání, i když není nevyhnutelně poslední úrovní testování. Například integrační testování rozsáhlých systémů může být vykonáno po akceptačním testování systému.

Akceptační testování může být přítomno v různých fázích životního cyklu, například:

- Krabicový softwarový produkt může být akceptačně testován, když je instalován nebo integrován.
- Akceptační testování použitelnosti komponenty může být vykonáváno v průběhu testování komponenty.
- Akceptační testování nového funkcionálního rozšíření může být realizováno před systémovým testováním.

Typické formy akceptačního testování jsou následující:

Uživatelské akceptační testování
Obvykle verifikuje připravenost pro použití systému koncovymi uživateli.

Provozní akceptační testování
Akceptace systému systémovými administrátory, včetně:

- testování zálohy/obnovy;
- obnovení po havárii;
- správy uživatelů;
- úkolů údržby;
- načítávání dat a migrační úkoly;
- pravidelné kontroly zranitelností v bezpečnosti.

Smluvní a regulační akceptační testování
Smluvní akceptační testování je vykonáváno vůči smluvním akceptačním kritériím pro provoz softwaru vyvinutého na zakázku. Akceptační kritéria by měla být definována v době, kdy zúčastněné strany odsouhlasí kontrakt.
Regulační testování je vykonáváno vůči jakýmkoliv předpisům, které musí být dodrženy, jako například vládní, právní nebo bezpečnostní předpisy.

Alfa testování a beta testování (nebo testování "v poli")
Vývojáři softwaru na zakázku nebo krabicového softwaru chtějí často dostat zpětnou vazbu od potenciálních nebo existujících zákazníků na jejich trhu předtím, než je software uvolněn do komerčního prodeje. Alfa testování je vykonáváno v prostředí vyvíjející organizace, nikoliv však vývojářským týmem. Beta testování, nebo testování "v poli", je vykonáváno zákazníky nebo potencionálními zákazníky v jejich vlastním prostředí.
Organizace mohou pro systémy testované před tím a po tom, než jsou přeneseny na stranu zákazníka, používat i jiné termíny, jako například „podnikové akceptační testování" a „akceptační testování u zákazníka".

2.3 *Typy testů (Z2)*	*40 minut*

Základní výrazy

Testování černé skříňky, pokrytí kódu, funkcionální testování, testování součinnosti (interoperability), zátěžové testování, testování udržovatelnosti, testování výkonnosti, testování přenositelnosti, testování spolehlivosti, testování bezpečnosti, testy extrémní zátěže, strukturální testování, testování použitelnosti, testování bílé skříňky.

Pozadí

Skupina testovacích aktivit může být zaměřena na ověření softwarového systému (nebo jeho části) na základě specifického důvodu nebo cíle testování.

Každý typ testu je zaměřen na určitý účel testování, kterým může být některý z následujících:

- funkcionalita, jež má být vykonávaná softwarem;
- nefunkcionální kvalitativní charakteristika, jako například spolehlivost či použitelnost;
- struktura nebo architektura softwaru nebo systému;
- účel související se změnou, tj. potvrzení, že defekty byly opraveny (konfirmační testování), a hledání neúmyslných změn (regresní testování).

Model softwaru může být vyvinut a/nebo použit ve strukturálním testování (tj. model řídicích toků nebo model struktury menu), nefunkcionálním testování (tj. zátěžový model, model použitelnosti nebo model bezpečnosti) a funkcionálním testování (tj. model procesních toků, model přechodu stavů nebo jednoduchá slovní specifikace).

2.3.1 Testování funkcionality (funkcionální testování) (Z2)

Funkcionality, které má systém, subsystém nebo komponenta vykonávat, mohou být popsány v pracovních produktech, jako jsou specifikace požadavků, případy užití nebo funkcionální specifikace; nebo mohou být nedokumentované. Funkcionality představují, “co“ systém dělá.

Funkcionální testy jsou založeny na funkcionalitách a vlastnostech (jak jsou popsány v dokumentech nebo chápány testery) a jejich spolupůsobení se specifickými systémy; a mohou být vykonávány na všech úrovních testování (např. testy komponent mohou být založeny na specifikaci komponent).

Techniky založené na specifikaci mohou být použity s cílem odvodit testovací podmínky a testovací případy z funkcionality softwaru nebo systému. (viz kapitola 4.) Funkcionální testování zkoumá externí chování softwaru (testování černé skříňky).

Jeden z typů funkcionálního testování – testování bezpečnosti – zkoumá funkcionalitu (např. firewall) vztahující se k detekci hrozeb, jako například viry zlomyslných outsiderů. Další typ funkcionálního testování – testování součinnosti (interoperability) – hodnotí schopnost interakce softwarového produktu s jednou nebo více určenými komponentami nebo systémy.

2.3.2 Testování nefunkcionálních charakteristik softwaru (nefunkcionální testování) (Z2)

Nefunkcionální testování zahrnuje testování výkonnosti, zátěžové testování (load testing), testy extrémní zátěže (stress testing), testování použitelnosti, testování udržovatelnosti, testování spolehlivosti a testování přenositelnosti, ale neomezuje se pouze na tyto. Je testováním toho, “jak“ systém pracuje.

Nefunkcionální testování může být vykonáváno na všech úrovních testování. Termín nefunkcionální testování popisuje testy vyžadované pro měření charakteristik systému a softwaru, které mohou být kvantifikovány vůči různým stupnicím měření, jako například doby odezvy pro testování výkonnosti. Tyto testy se mohou odvolávat na kvalitativní modely, jako například na model definovaný v standardu 'Softwarové inženýrství – Kvalita softwarového produktu' (ISO 9126).

2.3.3 Testování struktury/architektury softwaru (strukturální testování) (Z2)

Strukturální testování (testování bílé skříňky) může být vykonáváno na všech úrovních testování. Strukturální techniky jsou nejlépe použity po uplatnění technik založených na specifikaci za účelem umožnit měření důkladnosti testování pomocí stanovení pokrytí typu struktury.

Pokrytí je míra, do jaké je struktura prověřena testovací sadou. Vyjádřena je jako procento položek, které daná sada pokrývá. Když není pokrytí 100%, pak mohou být pro položky, které chybějí, navrženy dodatečné testy, které je otestují, a zvýší tak pokrytí. Techniky pokrytí jsou obsahem kapitoly 4.

Na všech úrovních testování, avšak obzvlášť při testování komponent a integračním testování komponent, mohou být použity nástroje na měření pokrytí elementů kódu, jako například pokrytí příkazů nebo rozhodnutí. Strukturální testování může být založeno na architektuře systému, jako například hierarchie volání částí systému.

Přístupy strukturálního testování mohou být též aplikovány na systémové, systémové integrační nebo akceptační úrovni testování (např. na obchodní modely nebo struktury menu).

2.3.4 Testování související se změnami: konfirmační testování (přetestování) a regresní testování (Z2)

Poté, co je defekt nalezen a opraven, měl by být software retestován za účelem potvrzení, že původní defekt byl úspěšně odstraněn. Takovéto testování se nazývá konfirmační testování. Ladění (lokalizace a oprava defektu) je aktivita vývoje, nikoliv aktivita testování.

Regresní testování je opakované testování již testovaného programu po modifikaci s cílem najít všechny defekty, které byly zaneseny nebo objeveny jako následek změny (změn). Tyto defekty se mohou nacházet v testovaném softwaru nebo v jiné související nebo nesouvisející softwarové komponentě. Regresní testování se provádí, když je změněn software nebo jeho prostředí. Rozsah regresního testování je odvozen od rizika nenalezení defektů v softwaru, který předtím fungoval.

Pokud se testy mají používat při konfirmačním testovaní a napomáhat regresnímu testování, měly by být opakovatelné.

Regresní testování může být vykonáváno na všech úrovních testování a využívá se k funkcionálnímu, nefunkcionálnímu a strukturálnímu testování. Sady regresních testů jsou spuštěny mnohokrát a jejich vývoj je obecně pomalý, proto je regresní testování silným kandidátem na automatizaci.

2.4 *Testování údržby (Z2)*	*15 minut*

Základní výrazy
Analýza dopadu, testování údržby.

Pozadí
Jednou vyvinutý softwarový systém je často v provozu roky až desetiletí. V průběhu tohoto času jsou systém, jeho konfigurační data anebo jeho prostředí často opravovány, měněny nebo rozšiřovány. Plánování testování v předstihu je rozhodující pro úspěšné testování údržby. Testování údržby je vykonáváno na existujícím provozním systému a je vyvoláno modifikacemi, migrací nebo vyřazením (z používání) softwaru nebo systému.

Modifikace zahrnují plánované zlepšující změny (např. založené na vydání softwaru - release), korigující a naléhavé nouzové změny, změny prostředí (jako například plánovaný upgrade operačního systému nebo databáze), plánované aktualizace krabicového softwaru nebo patche opravující nově odkrytá nebo objevená slabá místa operačního systému.

Testování údržby při migraci (např. z jedné platformy na jinou) by mělo zahrnovat provozní testy nového prostředí i změněného softwaru. Testování migrace (testování konverze) je taktéž potřebné v případech, kdy se do udržovaného systému migrují data z jiné aplikace.

V případě, že je při vyřazování systému z používání vyžadována úschova dat na delší časové období, může testování údržby při vyřazení systému z používání zahrnovat testování datové migrace nebo archivování.

Navíc, kromě testování změněných částí softwaru, zahrnuje testování údržby i regresní testování těch částí softwaru, které nebyly změněny. Rozsah testování údržby je závislý na riziku vyplývajícím ze změny, na velikosti existujícího systému a na velikosti změny. V závislosti na změnách může být testování údržby vykonáno na některých nebo na všech úrovních testování pro některé nebo všechny typy testů.

Určení, jak může být existující systém ovlivněn změnami, se nazývá analýza dopadu. Používá se s cílem pomoci rozhodnout se, do jaké míry vykonat regresní testování. Analýza dopadu může být použita k určení regresní testovací sady.

Testování údržby může být náročné v případě neaktuální nebo chybějící specifikace, nebo když testeři se znalostí dané oblasti nejsou k dispozici.

Reference
2.1.3 CMMI, Craig, 2002, Hetzel, 1988, IEEE 12207
2.2 Hetzel, 1988
2.2.4 Copeland, 2004, Myers, 1979
2.3.1 Beizer, 1990, Black, 2001, Copeland, 2004
2.3.2 Black, 2001, ISO 9126
2.3.3 Beizer, 1990, Copeland, 2004, Hetzel, 1988
2.3.4 Hetzel, 1988, IEEE Std 829-1998
2.4 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE Std 829-1998

3. Statické techniky (Z2)	60 minut

Studijní cíle pro statické techniky

Tyto cíle určují, co budete umět po dokončení každého modulu.

3.1 Statické techniky a proces testování (Z2)

SC-3.1.1 Rozpoznat softwarové pracovní produkty, které mohou být prošetřeny různými statickými technikami. (Z1)
SC-3.1.2 Popsat důležitost a význam s ohledem na statické techniky pro hodnocení softwarových pracovních produktů. (Z2)
SC-3.1.3 Vysvětlit rozdíl mezi statickými a dynamickými technikami s ohledem na cíle, typy defektů, které mají být nalezeny, a roli těchto technik v životním cyklu softwaru. (Z2)

3.2 Revizní proces (Z2)

SC-3.2.1 Vybavit si aktivity, role a zodpovědnosti typické formální revize. (Z1)
SC-3.2.2 Vysvětlit rozdíly mezi různými typy revizí: neformální revize, technická revize, předvedení (walkthrough) a inspekce. (Z2)
SC-3.2.3 Vysvětlit faktory úspěšného vykonání revizí. (Z2)

3.3 Statická analýza s použitím nástrojů (Z2)

SC-3.3.1 Vybavit si typické defekty a omyly nalezené statickou analýzou a porovnat je s revizemi a dynamickým testováním. (Z1)
SC-3.3.2 Popsat pomocí příkladů typické přínosy statické analýzy. (Z1)
SC-3.3.3 Vyjmenovat obvyklé defekty v kódu a návrhu, které mohou být odhaleny nástroji pro statickou analýzu. (Z1)

3.1 *Statické techniky a proces testování (Z2)*	*15 minut*

Základní výrazy
Dynamické testování, statické testování.

Pozadí
Na rozdíl od dynamického testování, které vyžaduje spouštění softwaru, statické testovací techniky se spoléhají na manuální prozkoumání (revidování) a automatizovanou analýzu (statickou analýzu) kódu nebo jiné projektové dokumentace bez spuštění kódu.

Revize jsou způsobem testování softwarových pracovních produktů (včetně kódu) a mohou být provedeny mnohem dříve než dynamické testy. Defekty nalezené během revizí v raných fázích životního cyklu (např. defekty v požadavcích) jsou často odstranitelné mnohem levněji než ty, které jsou nalezeny testy běžícími nad vykonaným kódem.

Celá revize může být vykonána jako manuální aktivita, ale existují i podpůrné nástroje. Hlavní manuální aktivitou je prozkoumání pracovního produktu a zaznamenání poznámek o něm. Každý softwarový pracovní produkt může být revidován, včetně specifikací požadavků, specifikací návrhů, kódu, testovacích plánů, testovacích specifikací, testovacích případů, testovacích skriptů, uživatelských příruček nebo webových stránek.

Přínosy revizí zahrnuji včasnou detekci a opravu defektů, vylepšení produktivity vývoje, zkrácený časový interval pro vývoj, snížené náklady a čas na testování, snížené náklady na životní cyklus softwaru, méně defektů a kvalitnější komunikaci. Revize mohou objevit opomenuté věci, například v požadavcích, které by se jen s malou pravděpodobností našly v rámci dynamického testování.

Revize, statická analýza a dynamické testování mají stejný cíl – identifikaci defektů. Vzájemně se doplňují - různé techniky mohou objevit rozdílné typy defektů mnohem efektivněji a účinněji. V porovnání s dynamickým testováním, statické techniky identifikují spíše příčiny selhání (defekty) než samotná selhání.

Typické defekty, které jsou v průběhu revidování snadněji odhalitelné než v dynamickém testování, zahrnují: odchylky vůči standardům, defekty v požadavcích, defekty v návrhu, nedostatečná udržovatelnost a nesprávná specifikace rozhraní.

3.2 *Revizní proces (Z2)*	*25 minut*

Základní výrazy

Vstupní kritéria, formální revize, neformální revize, inspekce, metrika, moderátor, vzájemná revize, revidující, zapisovatel, technická revize, předvedení (walkthrough).

Pozadí

Existují různé typy revizí, od neformálních, které jsou charakteristické tím, že nemají sepsané instrukce pro revidující, až po systematické, které jsou charakteristické týmovou účastí, dokumentovanými výsledky revize a dokumentovanými postupy pro vykonávání revize. Formálnost revizního procesu záleží na různých faktorech, jako jsou například zralost procesu vývoje, právní nebo regulační požadavky nebo potřeby záznamů pro audit.

Způsob, jakým je revize vykonána, závisí na dohodnutých cílech revize (např. najít defekty, získat porozumění, vzdělávat testery a nové členy týmu nebo diskutovat a rozhodovat na základě vzájemné dohody).

3.2.1 Aktivity formální revize (Z1)

Typická formální revize má následující hlavní aktivity:

1. Plánování:
   - definice revizních kriterií,
   - výběr osob,
   - přidělení rolí,
   - definice vstupních a výstupních kritérií při formálnějších typech revizí (např. inspekce),
   - výběr částí dokumentů, na které bude revize zaměřená,
   - prověření vstupních kriterií (u formálnějších typů revizí).
2. Kick-off:
   - distribuce dokumentů,
   - vysvětlení cílů, procesu a dokumentů účastníkům revize.
3. Individuální příprava:
   - příprava na revizním setkání formou revize dokumentu(ů),
   - zaznamenání potenciálních defektů, otázek a připomínek.
4. Prošetření/vyhodnocení/zaznamenání výsledků (revizní setkání):
   - diskuze nebo zaznamenávání, s dokumentovanými výsledky nebo zápisem (při formálnějších typech revizí),
   - zaznamenávání defektů, poskytování doporučení ohledně toho, jak se vypořádat s defekty, rozhodování o defektech.
5. Přepracování:
   - oprava nalezených defektů (obvykle vykonávaná autorem),
   - zaznamenání aktualizovaného stavu defektů (při formálních revizích).
6. Navazující kroky:
   - prověření, že defekty byly řešeny,
   - sběr metrik,
   - kontrola výstupních kriterií (při formálnějších typech revizí).

3.2.2 Role a zodpovědnosti (Z1)

Typická formální revize zahrnuje dále uvedené role:

- Manažer: rozhoduje o vykonávání revizí, alokuje čas v projektových harmonogramech a určuje, zda byly splněny cíle revize.

- Moderátor: osoba, která vede revizi dokumentu nebo sady dokumentů, jenž zahrnuje plánování revize, řízení průběhu setkání a vedení následných kroků po setkání. V případě potřeby může moderátor vykonávat roli zprostředkovatele mezi různými úhly pohledů na věc a je často osobou, na které závisí úspěch revize.
- Autor: autor nebo osoba s hlavní zodpovědností za revidovaný(é) dokument(y).
- Revidující: jednotlivci se specifickou technickou nebo obchodní znalostí (též nazývaní kontroloři nebo inspektoři), kteří, po nutné přípravě, identifikují a popíší nálezy (např. defekty) v revidovaném produktu. Revidující by měli být vybráni tak, aby reprezentovali různé pohledy a role v procesu revize, a měli by se účastnit každého revizního setkání.
- Zapisovatel (nebo zaznamenávající): dokumentuje všechny zásadní otázky, problémy a otevřené body, které byly identifikovány v průběhu setkání.

Prohlížení softwarových produktů anebo souvisejících pracovních produktů z různých perspektiv a používání kontrolních seznamů může zvýšit efektivnost a účinnost revize. Například kontrolní seznam z pohledu uživatele, údržby, testera nebo operátora, nebo kontrolní seznam typických problémů s požadavky může napomoci k odhalení dosud nedetekovaných problémů.

3.2.3 Typy revizí (Z2)

Jeden softwarový produkt anebo související pracovní produkt může být předmětem více než jedné revize. V případě, že je použito více typů revizí, může být jejich pořadí různé. Například neformální revize může být prováděna před technickou revizí, nebo inspekce specifikace požadavků může být prováděna před předvedením (walkthrough) se zákazníky. Hlavní charakteristiky, možnosti a účely běžných typů revizí jsou následující:

Neformální revize

- neexistuje žádný formální proces;
- může mít formu programování v párech nebo formu revize návrhu a kódu technickým vedoucím;
- výsledky mohou být dokumentovány;
- její užitečnost se různí v závislosti na revidujících;
- hlavní účel: levný způsob, jak získat alespoň nějaký přínos.

Předvedení (walkthrough)

- setkání vedené autorem;
- může mít formu scénářů, zkoušek „nanečisto” nebo participace skupiny kolegů;
- sezení s otevřeným koncem;
  - volitelná příprava revidujících před setkáním
  - volitelná příprava reportu revize včetně seznamu zjištění;
- volitelný zapisovatel (který není autorem);
- v praxi se může vyskytovat od relativně neformální až po velice formální formu;
- hlavní účely: učení, získání porozumění, nalezení defektů.

Technická revize

- dokumentovaný, definovaný proces odhalování defektů, který zahrnuje kolegy a technické experty s volitelnou účastí managementu;
- může být provedena jako vzájemná revize bez účasti managementu;
- ideálně vedena zaškoleným moderátorem (ne autorem);
- příprava revidujících před setkáním;
- volitelné použití kontrolních seznamů;
- příprava revizního reportu, který zahrnuje seznam zjištění, rozhodnutí, zda softwarový produkt splňuje požadavky, a tam kde to je vhodné, doporučení vztahující se k těmto zjištěním;
- v praxi se může vyskytovat od relativně neformální až po velice formální formu;

- hlavní účely: diskuze, rozhodování, vyhodnocování alternativ, nacházení defektů, řešení technických problémů a prověřování souladu se specifikacemi, plány, směrnicemi a standardy.

Inspekce

- vedená zaškoleným moderátorem (ne autorem);
- obvykle vykonávaná jako prozkoumání kolegy;
- definované role;
- zahrnuje sběr metrik;
- formální proces založený na pravidlech a kontrolních seznamech;
- specifikovaná vstupní a výstupní kritéria pro akceptaci softwarového produktu;
- příprava před setkáním;
- report inspekce včetně seznamu zjištění;
- formální proces následných kroků (s volitelnými prvky zlepšování procesu);
- volitelný předčítající;
- hlavní účel: najít defekty.

Předvedení (walkthrough), technické revize a inspekce mohou být vykonány v rámci skupiny kolegů – např. kolegy na stejné organizační úrovni. Tento typ revize se jmenuje „vzájemná revize“.

3.2.4 Faktory úspěchu pro revize (Z2)

Faktory úspěchu revize zahrnují:

- Každá revize má jasně předdefinované cíle.
- Revizí se zabývají vhodní lidé s ohledem na její cíle.
- Testeři jsou považováni za hodnotné revidující, kteří přispívají k revizi, a kteří se taktéž seznamují s produktem, což jim umožnuje připravit testy dříve.
- Nalezené defekty jsou vítány a jsou reportovány objektivně.
- Lidské spory a psychologické aspekty jsou zvládnuty (např. se staly pozitivní zkušeností pro autora).
- Revize je prováděna v důvěrné atmosféře, výsledek nebude použit k hodnocení zúčastněných.
- Revizní techniky jsou aplikovány vhodně za účelem dosažení cílů a s ohledem na typ a úroveň softwarových pracovních produktů a revidujících.
- Kontrolní seznamy nebo role jsou použity, pokud je to vhodné za účelem zvýšení efektivnosti identifikace defektů.
- Je zajištěno školení na revizní techniky, obzvlášť při použití formálnějších technik, jako například inspekce.
- Management podporuje náležitý proces revize (např. vyčleněním dostatečného času na revizní aktivity v rámci projektového harmonogramu).
- Klade se důraz na učení a zlepšování procesu.

3.3 *Statická analýza s použitím nástrojů (Z2)*	*20 minut*

Základní výrazy
Kompilátor, složitost, řídící tok, datový tok, statická analýza.

Pozadí
Cílem statické analýzy je najít defekty ve zdrojovém kódu softwaru a v softwarových modelech. Statická analýza je vykonávána bez vlastního spouštění programu zkoumaným nástrojem, zatímco dynamické testování spouští kód programu. Statická analýza dokáže objevit defekty, které jsou špatně odhalitelné v dynamickém testování. Stejně jako u revizí, statická analýza odhalí spíše defekty než selhání. Nástroje statické analýzy analyzují programový kód (např. řídící tok a datový tok), jakož i vygenerované výstupy jako HTML a XML.

Význam statické analýzy je následující:
- Včasná detekce defektů ještě před vykonáním testů.
- Včasné varování o podezřelých aspektech kódu nebo návrhu propočítáním metrik jako je například vysoká míra složitosti.
- Identifikace defektů, které by nebylo snadné odhalit dynamickým testováním.
- Detekce závislostí a nekonzistencí v softwarovém modelu, například propojení.
- Zlepšená udržovatelnost kódu a návrhu.
- Prevence defektů v případě, že došlo při vývoji k ponaučení.

Typické defekty objevené nástroji statické analýzy zahrnují:
- odkaz na proměnnou bez definované hodnoty;
- nekonzistentní rozhraní mezi moduly a komponentami;
- proměnné, které nejsou nikdy použity nebo jsou nesprávně deklarovány;
- nedosažitelný (mrtvý) kód;
- chybějící a chybná logika (potenciálně nekonečné smyčky);
- příliš komplikované konstrukty;
- porušení standardů kódování;
- bezpečnostní nedostatky;
- porušení syntaxe kódu a softwarových modelů.

Nástroje statické analýzy jsou obyčejně používány vývojáři (prověření vůči předdefinovaným pravidlům nebo standardům kódování) před a v průběhu testování komponent a integračního testování, nebo při ukládání kódu pomocí nástroje pro správu konfigurací (verzovací systém), a návrháři v průběhu modelování softwaru. Nástroje statické analýzy mohou produkovat velké množství varovných hlášení, které je nezbytné dobře zpracovat, aby byl nástroj využit co nejefektivněji.

Kompilátory mohou poskytovat částečnou podporu pro statickou analýzu, včetně výpočtu metrik.

Reference
3.2 IEEE 1028
3.2.2 Gilb, 1993, van Veenendaal, 2004
3.2.4 Gilb, 1993, IEEE 1028
3.3 Van Veenendaal, 2004

4. Techniky tvorby testů (Z4) — 285 minut

Studijní cíle pro techniky tvorby testů

Tyto cíle určují, co budete umět po ukončení každého modulu.

4.1 Proces vývoje testů (Z3)

SC-4.1.1 Rozlišovat mezi specifikací návrhu testů, specifikací testovacího případu a specifikací testovací procedury. (Z2)
SC-4.1.2 Porovnat pojmy testovací podmínka, testovací případ a testovací procedura. (Z2)
SC-4.1.3 Vyhodnotit kvalitu testovacích případů s ohledem na přímou trasovatelnost vzhledem k požadavkům a očekávaným výsledkům. (Z2)
SC-4.1.4 Převádět testovací případy do správně strukturované specifikace testovací procedury na úrovni detailu odpovídajícího znalostem testerů. (Z3)

4.2 Kategorie technik návrhu testů (Z2)

SC-4.2.1 Zdůvodnit, v čem jsou užitečné techniky návrhu testů založené na specifikaci (techniky návrhu černé skříňky) a techniky založené na struktuře (techniky návrhu bílé skříňky) a pro každou z nich vyjmenovat běžné techniky. (Z1)
SC-4.2.2 Vysvětlit charakteristiky a rozdíly mezi testováním založeným na specifikaci, struktuře a zkušenosti. (Z2)

4.3 Techniky založené na specifikaci neboli techniky černé skříňky (Z3)

SC-4.3.1 Navrhnout testovací případy z existujících softwarových modelů s použitím rozdělení tříd ekvivalence, analýzy hraničních hodnot, rozhodovacích tabulek a diagramů/tabulek přechodu stavů. (Z3)
SC-4.3.2 Vysvětlit hlavní účel každé ze čtyř testovacích technik, úroveň a typ testů, které mohou dané techniky využívat, a objasnit jakým způsobem může být měřeno pokrytí. (Z2)
SC-4.3.3 Vysvětlit koncept testování případů užití a jeho přínosy. (Z2)

4.4 Techniky založené na struktuře neboli techniky bílé skříňky (Z3)

SC-4.4.1 Popsat koncept a přidanou hodnotu pokrytí kódu. (Z2)
SC-4.4.2 Vysvětlit koncepty pokrytí příkazů a rozhodnutí a uvést důvody, proč se tyto koncepty mohou kromě testování komponent použít i na jiných úrovních testování (např. pro obchodní procedury na úrovni systému). (Z2)
SC-4.4.3 Navrhnout testovací případy pro dané řídící toky s použitím technik návrhu testů příkazů a rozhodnutí. (Z3)
SC-4.4.4 Vyhodnotit pokrytí příkazů a rozhodnutí z hlediska úplnosti, s ohledem na definované výstupní kritéria. (Z3)

4.5 Techniky založené na zkušenostech (Z2)

SC-4.5.1 Uvědomit si důvody psaní testovacích případů založených na intuici, zkušenosti a znalosti typických defektů. (Z1)
SC-4.5.2 Porovnat techniky testování založené na zkušenostech s technikami založenými na specifikaci. (Z2)

4.6 Výběr testovacích technik (Z2)

SC-4.6.1 Klasifikovat techniky návrhu testů podle jejich vhodnosti v daném kontextu, s ohledem na testovací bázi, související modely a charakteristiky softwaru. (Z2)

4.1 *Proces vývoje testů (Z3)*	*15 minut*

Základní výrazy

Specifikace testovacího případu, návrh testů, harmonogram provedení testů, specifikace testovací procedury, testovací skript, trasovatelnost.

Pozadí

Proces vývoje testů popisovaný v této kapitole může být vykonáván různými způsoby, od velmi neformálního se stručnou nebo žádnou dokumentací až po velmi formální (viz níže). Stupeň formálnosti závisí na kontextu testování, včetně úrovně zralosti testování a vývojového procesu, časových omezeních, bezpečnostních a regulatorních požadavcích a zainteresovaných lidech.

V průběhu analýzy testování jsou analyzovány dokumenty vztahující se k základu testování s cílem určit, co se bude testovat, tj. identifikovat testovací podmínky. Testovací podmínka je definována jako položka nebo událost, která může být verifikována jedním nebo více testovacími případy (např. funkcionalita, transakce, kvalitativní charakteristika nebo strukturální element).

Zavedení trasovatelnosti směrem od testovacích podmínek zpět ke specifikacím a požadavkům umožňuje zajistit jak účinnou analýzu dopadu (když se mění požadavky), tak i určení pokrytí požadavků pro danou sadu testů. V průběhu analýzy testování se implementuje detailní přístup k testování za účelem výběru technik návrhu testů, které budou použity, a to (kromě jiných kritérií) na základě identifikovaných rizik (viz kapitola 5, analýza rizik).

V průběhu návrhu testů jsou vytvářeny a specifikovány testovací případy a testovací data. Testovací případ sestává ze sady vstupních hodnot, předpokladů pro provedení testu, očekávaných výsledků a následných podmínek po provedení testu. Je definován tak, aby pokryl určité testovací cíle nebo testovací podmínky. Standard pro dokumentaci testování softwaru (IEEE Std 829-1998) popisuje obsah specifikací návrhu testů (obsahujících testovací podmínky) a specifikací testovacích případů.

Očekávané výsledky by měly být vytvořeny jako část specifikace testovacího případu a měly by zahrnovat výstupy, změny dat a stavů a jakékoliv další důsledky testu. Pokud nebudou očekávané výsledky definovány, může být přijatelný, ale chybný výsledek, interpretován jako správný. Očekávané výsledky by měly být v ideálním případě definovány dříve, než se začnou vykonávat testy.

V průběhu implementace testů jsou vyvíjeny a implementovány testovací případy, je určována jejich priorita a tyto testovací případy jsou uspořádány do specifikace testovacích procedur (IEEE Std 829-1998). Testovací procedura specifikuje pořadí činností pro provedení testu. Pokud jsou testy spuštěny s využitím nástroje pro spouštění testů, je pořadí činností specifikováno v testovacím skriptu (automatizovaná testovací procedura).

Různé testovací procedury a automatizované testovací skripty jsou pak zformovány do harmonogramu provedení testů, který definuje pořadí, v jakém jsou vykonávány různé testovací procedury a případně automatizované testovací skripty. Harmonogram provedení testů bere v úvahu také faktory jako jsou regresní testy, stanovení priorit, technické a logické závislosti.

4.2 *Kategorie technik návrhu testů (Z2)*	*15 minut*

Základní výrazy

Technika návrhu testů černé skříňky, technika návrhu testů založená na zkušenostech, technika návrhu testů, technika návrhu testů bílé skříňky.

Pozadí

Účelem techniky návrhu testů je identifikovat testovací podmínky, testovací případy a testovací data.

Tradičně se testovací techniky rozlišují na techniky černé skříňky nebo techniky bílé skříňky. Techniky návrhu testů černé skříňky (nazývané taktéž jako techniky založené na specifikaci) slouží na odvození a výběr testovacích podmínek, testovacích případů nebo testovacích dat. Tyto techniky vycházejí z analýzy dokumentace základu testování a zahrnují funkcionální a nefunkcionální testování. Techniky testování černé skříňky podle definice nepoužívají žádnou informaci o vnitřní struktuře komponenty nebo systému, který má být testovaný. Techniky návrhu testů bílé skříňky (taktéž nazývány strukturální techniky nebo techniky založené na struktuře) jsou založené na analýze struktury komponenty nebo systému. Testování černé skříňky a testování bílé skříňky může být taktéž kombinováno s technikami založenými na zkušenostech za účelem zužitkovat zkušenosti vývojářů, testerů a uživatelů k určení toho, co má být testováno.

Některé techniky patří výlučně do jedné kategorie; jiné obsahují prvky více než jedné kategorie.
Tato učební osnova označuje techniky návrhu testů založených na specifikaci jako techniky černé skříňky a techniky návrhu testů založených na struktuře jako techniky bílé skříňky. Pokrývá navíc ještě techniky návrhu testů založených na zkušenostech.

Techniky návrhu testů založených na specifikaci mají následující společné charakteristiky:

- Pro specifikování řešeného problému softwaru nebo jeho komponent se používají formální nebo neformální modely.
- Z těchto modelů se mohou systematicky odvozovat testovací případy.

Techniky návrhu testů založených na struktuře mají následující společné charakteristiky:

- Informace o procesu vývoje softwaru, které používá při odvozování testovacích případů (např. kód a detailní informace o návrhu). Při odvozování testovacích případů se používá informace o tom, jak je software vytvořen (např. kód a detailní informace o návrhu).
- Rozsah pokrytí softwaru může být měřen pro existující testovací případy a následné testovací případy mohou být systematicky odvozeny za účelem zvýšení pokrytí.

Techniky návrhu testů založených na zkušenostech mají následující společné charakteristiky:

- Při odvozování testovacích případů se využívají znalosti a zkušenosti lidí.
- Jedním zdrojem informací jsou znalosti testerů, vývojářů, uživatelů a dalších zainteresovaných osob ve vztahu k softwaru, jeho použití a prostředí.
- Dalším zdrojem informací jsou znalosti o pravděpodobných defektech a jejich statistickém rozložení.

4.3 *Techniky založené na specifikaci neboli techniky černé skříňky (Z3)*	*150 minut*

Základní výrazy
Analýza hraničních hodnot, testování rozhodovací tabulky, rozdělení tříd ekvivalence, testování přechodu stavů, testování případů užití.

4.3.1 Rozdělení tříd ekvivalence (Z3)

Při rozdělení tříd ekvivalence jsou vstupy softwaru nebo systému rozděleny do skupin, ve kterých se očekává stejné chování, takže jsou pravděpodobně zpracovány stejným způsobem. Třídy ekvivalence mohou být nalezeny jak pro platná data, tj. takové hodnoty, které by měly být akceptovány, tak i pro neplatná data, tj. takové hodnoty, které by měly být zamítnuty. Třídy mohou být identifikovány též pro výstupy, vnitřní hodnoty, časově související hodnoty (např. před nebo po události) a pro parametry rozhraní (např. integrované komponenty testované v průběhu integračního testování). Testy mohou být navrženy tak, aby pokryly všechny platné a neplatné sekce. Rozdělení tříd ekvivalence je aplikovatelné na všech úrovních testování.

Rozdělení tříd ekvivalence lze použít k dosažení cílů pokrytí vstupu a výstupu. To může být uplatněno při manuálním vstupu, vstupu do systému přes rozhraní nebo parametrech rozhraní při integračním testování.

4.3.2 Analýza hraničních hodnot (Z3)

Chování může být s větší pravděpodobností nesprávné na okraji každého rozdělení tříd ekvivalence než chování uvnitř dané třídy. Existují proto hranice oblastí, kde testování může najít defekty s větší pravděpodobností. Maximální a minimální hodnoty třídy představují její hraniční hodnoty. Hraniční hodnota pro platnou třídu je platná hraniční hodnota; hranice neplatné třídy je neplatná hraniční hodnota. Testy mohou být navrženy tak, aby pokryly obě, platné i neplatné hraniční hodnoty. V průběhu návrhu testovacích případů je vybrán test pro každou hraniční hodnotu.

Analýza hraničních hodnot může být aplikována na všech úrovních testování. Je relativně jednoduše aplikovatelná a její schopnost odhalení defektů je vysoká. Detailní specifikace jsou užitečné při určování zajímavých (např. potenciálně nebezpečných) hranic.

Tato technika je často považována za rozšíření rozdělení tříd ekvivalence nebo jiných technik návrhu testů černé skříňky. Může být použita na třídách ekvivalence pro uživatelský vstup s pomocí obrazovky, pro časové rozsahy (např. čas vypršení, požadavky na rychlost transakcí) nebo rozsahy tabulek (např. velikost tabulky je 256*256).

4.3.3 Testování rozhodovacích tabulek (Z3)

Rozhodovací tabulky jsou užitečným způsobem pro zachycení požadavků obsahujících logické podmínky a pro zdokumentování interního návrhu systému. Mohou být použity pro zaznamenání složitých obchodních pravidel, které má systém implementovat. Při vytváření rozhodovacích tabulek je analyzována specifikace a jsou identifikovány podmínky a činnosti systému. Vstupní podmínky a činnosti jsou nejčastěji určeny užitím hodnot pravda nebo nepravda (logický typ boolean). Rozhodovací tabulka obsahuje spouštěcí podmínky (často kombinace pravda a nepravda) pro všechny vstupní podmínky a výsledné činnosti pro každou kombinaci podmínek. Každý sloupec tabulky odpovídá specifickému pravidlu, které definuje unikátní kombinaci podmínek, jež vedou k vykonání činností souvisejících s tímto pravidlem. Standardně se požaduje, aby existoval nejméně jeden test na každý sloupec v tabulce, což obvykle zahrnuje pokrytí všech kombinací spouštěcích podmínek.

Výhodou testování rozhodovacích tabulek je vytvoření kombinací podmínek, které by jinak nemusely být přezkoušeny v průběhu testování. Technika může být aplikována na všechny situace, kdy je činnost softwaru závislá na několika logických rozhodnutích.

4.3.4 Testování přechodu stavů (Z3)

Systém může obecně poskytovat různou odpověď v závislosti na aktuálních podmínkách nebo předcházející historii (jeho stavu). V tom případě může být tento aspekt chování znázorněn pomocí diagramu přechodu stavů. Dovoluje testerovi chápat software z hlediska jeho stavů, přechodů mezi stavy, vstupů nebo událostí, které spouštějí změny stavů (přechody) a činností, které mohou z těchto přechodů vyplývat. Stavy systému nebo objektu, který je předmětem testování, jsou samostatné, identifikovatelné a je jich konečné množství. Stavová tabulka pak vyjadřuje vztah mezi stavy a vstupy a může zvýraznit přechody, které mohou být neplatné. Testy mohou být navrženy s cílem pokrýt typické sekvence stavů, pokrýt každý stav, vykonat každý přechod, vykonat specifické sekvence přechodů nebo testovat neplatné přechody stavů.

Testování přechodu stavů je velmi používáno v odvětví vestavěných (embedded) softwarových produktů a v technické automatizaci obecně. Tato technika je nicméně taktéž vhodná pro modelování uživatelských objektů, které mají specifické stavy nebo testování toků obrazovek (např. pro internetové aplikace nebo uživatelské scénáře).

4.3.5 Testování případů užití (Z2)

Testy mohou být odvozeny na základě případů užití (use cases). Případ užití popisuje interakce mezi aktéry (uživateli nebo systémy), které produkují výsledek, který má hodnotu pro uživatele systému anebo zákazníka. Případy užití mohou být popsané na abstraktní úrovni (případ užití, bez popisu technologie, na úrovni obchodního procesu) nebo na úrovni systému (systémový případ užití na úrovni funkcionality systému). Každý případ užití má předpoklady, které musí být splněny, aby případ úspěšně fungoval. Každý případ užití je ohraničen výstupními podmínkami, které jsou pozorovatelnými výsledky a finálním stavem systému poté, kdy je případ ukončen. Případ užití má obvykle základní (tj. nejpravděpodobnější) scénář a alternativní scénáře.

Případy užití popisují procesní toky přes systém založené na jeho pravděpodobném reálném použití, proto jsou testovací případy odvozeny z případů užití nejpřínosnější při odhalování defektů v procesních tocích v průběhu skutečného používání systému. Případy užití jsou velmi účelné pro navržení akceptačních testů za účasti zákazníka/uživatele. Taktéž pomohou odhalit integrační defekty způsobené interakcí a vzájemnou interferencí rozličných komponent, které by testování jednotlivých komponent neodhalilo. Navrhování testovacích případů z případů užití může být kombinované s jinými technikami testování založených na specifikaci.

4.4 *Techniky založené na struktuře neboli techniky bílé skříňky (Z4)*	*60 minut*

Základní výrazy
Pokrytí kódu, pokrytí rozhodnutí, pokrytí příkazů, testování založené na struktuře.

Pozadí
Testování založené na struktuře/testování bílé skříňky je založené na identifikované struktuře softwaru nebo systému, jak je zřejmé z následujících příkladů:

- Úroveň komponenty: struktura softwarové komponenty, tj. příkazy, rozhodnutí nebo větve.
- Integrační úroveň: struktura může být strom volání (diagram, ve kterém moduly volají další moduly).
- Systémová úroveň: struktura může být struktura menu, proces nebo struktura webové stránky.

V této části se rozebírají tři (s kódem související) strukturální techniky návrhu testů pro pokrytí kódu, a to založené na příkazech, větvích a rozhodnutích. Pro testování rozhodování může být použit diagram řídících toků na vizualizaci alternativ pro každé rozhodování.

4.4.1 Testování a pokrytí příkazů (Z4)

V testování komponent představuje pokrytí příkazů stanovení procenta vykonatelných příkazů, které byly vykonány sadou testovacích případů. Technika testování příkazů odvozuje testovací případy s cílem vykonat specifické příkazy, zpravidla za účelem zvýšení pokrytí příkazů.

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

4.4.2 Testování a pokrytí rozhodnutí (Z4)

Pokrytí rozhodnutí se vztahuje k testování větvení a je definováno procentem výsledků rozhodování (např. větev pravda a nepravda příkazu IF), které byly vykonány (sadou testovacích případů). Technika testování rozhodování odvozuje testovací případy s cílem vykonat specifické výsledky rozhodování. Větve vytvářejí rozhodovací body v kódu a zobrazují přenos řízení do jiných oblastí v kódu.

Pokrytí rozhodnutí je určené podílem počtu všech výsledků rozhodnutí, které jsou pokryté (navrhnutými nebo vykonanými) testovacími případy, a počtem všech možných výsledků rozhodnutí v testovaném kódu.

Testování rozhodování je forma testování řídicích toků, protože sleduje specifický tok řízení přes rozhodovací body. Pokrytí rozhodnutí garantuje („je silnější“) pokrytí příkazů: 100% pokrytí rozhodnutí garantuje 100% pokrytí příkazů, ale ne opačně.

4.4.3 Další techniky založené na strukturách (Z1)

Kromě pokrytí rozhodnutí existují silnější úrovně pokrytí struktur, například pokrytí podmínek a pokrytí vícenásobných podmínek.

Koncept pokrytí může být aplikován taktéž na další úrovně testů. Například na integrační úrovni může být procento modulů, komponent nebo tříd, které byly vykonány sadou testovacích případů, vyjádřené jako pokrytí modulů, komponent nebo tříd.

Při strukturálním testování kódu se osvědčuje podpora nástrojů.

4.5 *Techniky založené na zkušenosti (Z2)*	*30 minut*

Pojmy
Průzkumné testování, útok (na vady).

Pozadí
Při testování založeném na zkušenostech jsou testy odvozovány ze znalostí a intuice testerů a jejich zkušeností s podobnými aplikacemi a technologiemi. Pokud se použijí na rozšíření systematických technik, mohou být užitečné při identifikaci speciálních testů, u kterých by byla definice pomocí formálních technik náročná. Bohužel tato technika může vykazovat značně proměnlivé stupně efektivity v závislosti na zkušenosti testerů.
Obvykle používanou technikou založenou na zkušenostech je odhadování omylů. Testeři všeobecně předvídají defekty na základě zkušeností. Strukturovaný přístup k technice odhadování omylů znamená vypracování seznamu možných defektů a návrhu testů, které na tyto defekty budou útočit. Tento systematický přístup se nazývá útok na vady. Seznamy defektů a selhání mohou být vytvořeny na základě zkušenosti, dostupných dat o defektech a selháních a na základě všeobecných znalostí o tom, proč software selhává.

Průzkumné testování je souběžný návrh testů, provádění testů, zaznamenávání testů a učení. Je založeno na testovací listině (test charter) obsahující cíle testování a je vykonávané v definovaných časových úsecích. Je to způsob, který je nejužitečnější tam, kde buďto neexistují dostatečné nebo adekvátní specifikace, nebo existuje silný časový tlak. Dále se používá s cílem rozšířit nebo doplnit jiné, více formální testování. Může sloužit jako kontrola procesu testování pomáhající zajistit nalezení nejzávažnějších defektů.

4.6 *Výběr testovacích technik (Z2)*	*15 minut*

Pojmy
Žádné specifické pojmy.

Pozadí
Rozhodnutí, které techniky použít, závisí na mnoha faktorech jako jsou typ systému, regulační standardy, zákaznické nebo smluvní požadavky, úroveň rizika, typ rizika, cíl testování, dostupná dokumentace, znalosti testerů, čas a rozpočet, životní cyklus vývoje, modely případů užití, předcházející zkušenosti se zjištěnými typy defektů.

Některé techniky jsou lépe aplikovatelné pouze na určité situace a úrovně testování, další jsou aplikovatelné na všechny úrovně testování.

Při vytváření testovacích případů testeři obvykle používají kombinace testovacích technik za účelem zajištění dostatečného pokrytí objektu testování, včetně procesních technik a technik založených na pravidlech a datech.

Reference
4.1 Craig, 2002, Hetzel, 1988, IEEE Std 829-1998
4.2 Beizer, 1990, Copeland, 2004
4.3.1 Copeland, 2004, Myers, 1979
4.3.2 Copeland, 2004, Myers, 1979
4.3.3 Beizer, 1990, Copeland, 2004
4.3.4 Beizer, 1990, Copeland, 2004
4.3.5 Copeland, 2004
4.4.3 Beizer, 1990, Copeland, 2004
4.5 Kaner, 2002
4.6 Beizer, 1990, Copeland, 2004

5. Management testování (Z3) — 170 minut

Studijní cíle pro management testování

Tyto cíle určují, co budete umět po ukončení každého modulu.

5.1 Organizace testování (Z2)

SC-5.1.1 Pochopit důležitost nezávislého testování. (Z1)
SC-5.1.2 Vysvětlit výhody a nevýhody nezávislého testování v organizaci. (Z2)
SC-5.1.3 Pochopit různé týmové role potřebné pro sestavení testovacího týmu. (Z1)
SC-5.1.4 Zapamatovat si úlohy typického vedoucího testování a testera. (Z1)

5.2 Plánování a odhadování testování (Z3)

SC-5.2.1 Pochopit různé úrovně a cíle plánování testování. (Z1)
SC-5.2.2 Shrnout účel a obsah testovacího plánu, specifikace návrhu testů a dokumentů testovacích procedur na základě Standardu pro dokumentaci testování softwaru ('Standard for Software Test Documentation' IEEE Std 829-1998). (Z2)
SC-5.2.3 Rozlišovat mezi koncepčně odlišnými přístupy k testování jako jsou analytický, založený na modelech, metodický, zohledňující procesy/standardy, dynamický/heuristický, konzultativní a regresně-averzní. (Z2)
SC-5.2.4 Rozlišovat mezi předmětem plánování testování systému a přípravou harmonogramu provedení testů. (Z2)
SC-5.2.5 Napsat harmonogram provedení testů pro daný soubor testovacích případů s ohledem na stanovení priorit a technické a logické závislosti. (Z3)
SC-5.2.6 Vyjmenovat aktivity přípravy a vykonávání testů, které by měly být zohledněny v průběhu plánování testování. (Z1)
SC-5.2.7 Zapamatovat si typické faktory, které ovlivňují pracnost testování (Z1)
SC-5.2.8 Odlišit dva koncepčně rozdílné přístupy odhadování: přístup založený na metrikách a expertní přístup. (Z2)
SC-5.2.9 Pochopit/zdůvodnit adekvátní vstupní a výstupní kritéria pro specifické úrovně testování a skupiny testovacích případů (např. pro integrační testování, akceptační testování nebo testovací případy pro testování použitelnosti). (Z2)

5.3 Sledování a řízení postupu testování (Z2)

SC-5.3.1 Zapamatovat si běžné metriky používané pro monitorování přípravy a provádění testů. (Z1)
SC-5.3.2 Vysvětlit a porovnat metriky pro reportování a řízení testování (např. nalezené a opravené defekty, úspěšné a neúspěšné testy) podle jejich účelu použití. (Z2)
SC-5.3.3 Shrnout účel a obsah souhrnné zprávy z testování podle Standardu pro dokumentaci testování softwaru ('Standard for Software Test Documentation' IEEE Std 829-1998). (Z2)

5.4 Konfigurační management (Z2)

SC-5.4.1 Shrnout, jak konfigurační management podporuje testování. (Z2)

5.5 Riziko a testování (Z2)

SC-5.5.1 Popsat riziko jako možný problém, který by mohl ohrozit dosažení cílů jedné nebo více zainteresovaných osob na projektu. (Z2)
SC-5.5.2 Zapamatovat si, že úroveň rizika je určena pravděpodobností (výskytu) a dopadem (škody vzniklé, pokud událost nastane). (Z1)

SC-5.5.3 Rozlišit projektová a produktová rizika. (Z2)
SC-5.5.4 Poznat typická produktová a projektová rizika. (Z1)
SC-5.5.5 Popsat příklady, jak může být v plánování testování použita analýza a řízení rizik. (Z2)

5.6 Správa incidentů (Z3)

SC-5.6.1 Pochopit obsah záznamu o incidentu podle Standardu pro dokumentaci testování softwaru ('Standard for Software Test Documentation' IEEE Std 829-1998). (Z1)
SC-5.6.2 Napsat záznam o incidentu, který obsahuje pozorování selhání během testování. (Z3)

5.1 *Organizace testování (Z2)*	*30 minut*

Základní výrazy
Tester, vedoucí testování, manažer testování.

5.1.1 Organizace a nezávislost testování (Z2)

Efektivita nacházení defektů testováním a revidováním může být zvýšena použitím nezávislých testerů. Možnosti nezávislosti jsou odstupňovány takto:

- Žádní nezávislí testeři. Vývojáři testují vlastní kód.
- Nezávislí testeři v rámci vývojových týmů.
- Nezávislý testovací tým nebo skupina v rámci organizace reportující projektovému nebo výkonnému managementu.
- Nezávislí testeři ze stejné obchodní organizace nebo komunity uživatelů.
- Nezávislí specialisté pro specifické typy testů, jako například testeři pro použitelnost, bezpečnost nebo certifikaci (kteří certifikují softwarový produkt vůči standardům a předpisům).
- Nezávislí testeři pronajatí nebo externí vůči organizaci.

Pro velké, složité nebo z hlediska bezpečnosti kritické projekty je obyčejně nejlepší mít víceúrovňové testování, s tím, že některé nebo všechny úrovně jsou realizovány nezávislými testery. Vývoj může na testování participovat, obzvlášť na nižších úrovních, ale jeho nedostatek objektivity často omezuje efektivitu jím provedených testů. Nezávislí testeři mohou mít právo vyžadovat a definovat testovací procesy a pravidla, ale měli by přijímat takové procesně orientované role pouze při existenci jasného mandátu od managementu.

Výhody nezávislosti testování:

- Nezávislí testeři vidí jiné a rozdílné defekty a jsou nezaujatí.
- Nezávislý tester může ověřovat předpoklady, které byly stanoveny během specifikace a implementace systému.

Nevýhody nezávislosti testování:

- Izolace od vývojového týmu (když je tým úplně nezávislý).
- Nezávislí testeři mohou být úzkým hrdlem jako poslední kontrolní bod.
- Vývojáři mohou ztratit smysl pro zodpovědnost za kvalitu.
- Hrozí obviňování nezávislého testovacího týmu za zdržení ve vydání softwaru do produkce (release).

Testovací úlohy mohou být realizovány lidmi ve specifické testovací roli nebo někým v jiné roli, jako například projektovým manažerem, manažerem kvality, vývojářem, expertem v příslušném obchodním oboru, osobou z oddělení infrastruktury nebo IT provozu.

5.1.2 Úlohy vedoucího testování a testera (Z1)

Tyto učební osnovy se zabývají dvěma testovacími rolemi: vedoucí testování a tester. Činnosti a úlohy vykonávané lidmi v těchto dvou rolích jsou závislé na kontextu projektu a produktu, lidech v daných rolích a organizaci.

Někdy je vedoucí testování nazýván manažerem testování nebo koordinátorem testování. Role vedoucího testování může být vykonávána projektovým manažerem, manažerem vývoje, manažerem pro zabezpečení kvality nebo manažerem testovací skupiny. Ve větších projektech mohou existovat dvě pozice: vedoucí testování a manažer testování. Vedoucí testování zpravidla plánuje, monitoruje a řídí testovací aktivity a úlohy, jak bylo definováno v sekci 1.4.

Typické úlohy vedoucího testování mohou zahrnovat:

- Koordinování testovací strategie a plánu testování s projektovými manažery a jinými osobami.
- Vytváření nebo revidování testovací strategie pro projekt a politiky testování pro organizaci.
- Podílení se na projektových činnostech z pohledu testování, jako např. integrační plánování.
- Plánování testů – se zřetelem na kontext a pochopením cíle testování a rizika – včetně výběru přístupů k testování, odhadování času, pracnosti a nákladů na testování, získávání zdrojů, definování úrovní testování, cyklů a plánování správy incidentů.
- Iniciování specifikace, přípravy, implementace a provedení testů, monitorování výsledků testování a kontrola výstupních kriterií.
- Přizpůsobování plánování založené na výsledcích a postupu testování (někdy dokumentovaném ve status reportu) a přijímání opatření potřebných na odstranění problémů.
- Nastavení adekvátní správy konfigurací testwaru za účelem trasovatelnosti.
- Zavedení vhodných metrik pro měření postupu testování a hodnocení kvality testování a produktu.
- Rozhodování o tom, co by mělo být automatizováno, v jaké míře a jakým způsobem.
- Výběr nástrojů na podporu testování a organizování školení na používání nástrojů pro testery.
- Rozhodování o implementaci testovacího prostředí.
- Vytváření souhrnných zpráv z testování založených na informacích získaných během testování.

Typické úkoly testera mohou zahrnovat:

- Revidování a podílení se na tvorbě testovacích plánů.
- Analýzu, revidování a posouzení požadavků uživatelů, specifikací a modelů testovatelnosti.
- Tvorbu specifikací testů.
- Nastavování testovacího prostředí (často koordinováno s administrátory systému a sítí).
- Přípravu a získávání testovacích dat.
- Implementaci testů na všech úrovních testování, provedení a zaznamenávání testů, vyhodnocování výsledků testů a dokumentování odchylek od očekávaných výsledků.
- Používání nástrojů pro administraci a management testování a nástrojů pro monitorování testování podle potřeby.
- Automatizování testů (může být podpořeno vývojářem nebo expertem pro automatizaci testů).
- Měření výkonnosti komponent nebo systémů (když je to vhodné).
- Revidování testů vytvořených jinými testery.

Členové týmu, kteří pracují na analýze testování, návrhu testování, specifických typech testování nebo na automatizaci testování, mohou být specialisty v těchto rolích. V závislosti na úrovni testování a rizicích vztahujících se k produktu a projektu, mohou roli testera převzít různí lidé s cílem udržet určitý stupeň nezávislosti. Obvykle by testeři na úrovni testování komponent a integračních testů mohli být vývojáři, testeři na akceptační úrovni testování obchodní experti a uživatelé, testeři na provozní akceptační úrovni testování mohou být operátoři.

5.2 *Plánování a odhadování testování (Z3)*	*40 minut*

Základní výrazy
Přístup k testování, testovací strategie.

5.2.1 Plánování testování (Z2)

Tato část vysvětluje důvody pro plánování testování v rámci vývojových a implementačních projektů a pro aktivity údržby. Plánování může být dokumentováno v projektovém nebo hlavním testovacím plánu a v oddělených plánech pro různé úrovně testování jako jsou systémové nebo akceptační testy. Vzory dokumentů pro plánování testů pokrývá Standard pro dokumentaci testování softwaru 'Standard for Software Test Documentation' (IEEE Std 829-1998).

Plánování je ovlivněno pravidly testování organizace, rozsahem testování, cíli, riziky, omezeními, kritičností, testovatelností a disponibilitou zdrojů. Při postupu v projektovém plánování a plánování testů se objevuje stále více informací a tak do plánu může být průběžně zahrnováno stále více podrobností. Plánování testování je kontinuální činnost a je vykonáváno ve všech procesech a aktivitách životního cyklu. Zpětná vazba z testovacích činností se používá k rozpoznání měnících se rizik, aby se plánování mohlo přizpůsobovat.

5.2.2 Aktivity plánování testování (Z3)

Aktivity plánování testování pro celý systém nebo části systému mohou zahrnovat:

- Určení rozsahu a rizik a identifikování cílů testování.
- Definování celkového přístupu testování, včetně definice úrovní testování, vstupních a výstupních kriterií.
- Integraci a koordinaci testovacích aktivit do aktivit životního cyklu softwaru: akvizice, nabídka, vývoj, provoz a údržba.
- Vykonávání rozhodnutí o tom, co testovat, které role budou vykonávat testovací aktivity, jak budou testovací aktivity vykonány a jak budou vyhodnocovány výsledky testování.
- Přípravu harmonogramů aktivit analýzy a návrhu testování.
- Přípravu harmonogramů implementace, provedení a vyhodnocení testů.
- Přidělování zdrojů na různé definované úlohy.
- Definování množství, úrovně detailu, struktury a šablon pro testovací dokumentaci.
- Výběr metrik pro monitoring a řízení přípravy a provedení testů, řešení defektů a sledování rizik.
- Nastavení úrovně detailu pro testovací procedury s cílem poskytnout dostatek informací na podporu reprodukovatelné přípravy a provedení testů.

5.2.3 Vstupní kritéria (Z2)

Vstupní kritéria definují, kdy začít testovat, jako například na začátku úrovně testování nebo když je sada testů připravena pro spuštění.

Typická vstupní kritéria mohou být následující:

- Dostupnost a připravenost testovacího prostředí
- Připravenost testovacího nástroje v testovacím prostředí
- Dostupnost testovatelného kódu
- Dostupnost testovacích dat

5.2.4 Výstupní kritéria (Z2)

Výstupní kritéria definují, kdy přestat testovat, jako například na konci jedné úrovně testování nebo když sada testů dosáhne specifický cíl.

Typická výstupní kritéria mohou být:

- Měření důslednosti, jako například pokrytí kódu, funkcionality nebo rizika.
- Odhady hustoty defektů nebo měření spolehlivosti.
- Náklady.
- Reziduální rizika, jako například neopravené defekty nebo nedostatečné pokrytí testy v určitých oblastech.
- Harmonogramy, například založené na době uvedení na trh.

5.2.5 Odhadování testování (Z2)

Tyto učební osnovy pokrývají dva přístupy k odhadování pracnosti testování:

- Přístup založený na metrikách: odhadování pracnosti testování na základě metrik minulých nebo podobných projektů nebo založené na typických hodnotách.
- Přístup založený na expertíze: odhadování úloh jejich vlastníky nebo experty.

Poté, co je odhadnuta pracnost testování, mohou být identifikovány zdroje a sestaven harmonogram.

Pracnost testování je závislá na množství faktorů, mezi které patří:

- Charakteristiky produktu: kvalita specifikace a jiných informací používaných v testovacích modelech (tj. testovací báze), velikost produktu, složitost domény problému, požadavky na spolehlivost a bezpečnost a požadavky na dokumentaci.
- Charakteristiky procesu vývoje: stabilita organizace, používané nástroje, proces testování, schopnosti lidí a časový tlak.
- Výsledek testování: počet defektů a množství požadovaného přepracovávání.

5.2.6 Testovací strategie, přístupy k testování (Z2)

Přístup k testování implementuje testovací strategie pro specifický projekt. Přístup k testování je definován a vypracován v testovacích plánech a návrzích testů. Obvykle zahrnuje rozhodnutí učiněné na základě cílů a ohodnocení rizika (testovacího) projektu. Je výchozím bodem pro plánování procesu testování, výběru technik návrhu testů a použitých typů testů a taktéž pro definování vstupních a výstupních kriterií.

Vybraný přístup závisí na kontextu a může posuzovat rizika, nebezpečí a bezpečnost, dostupné zdroje a schopnosti, technologii, povahu systému (např. software vyvinutý na zakázku versus krabicový software).
Typické přístupy zahrnují:

- Analytické přístupy, například testování založené na rizicích, kde je testování směřováno k nejrizikovějším oblastem.
- Přístupy založené na modelech jako je stochastické testování používající statistické informace o mírách selhání (jako modely růstu spolehlivosti) nebo používání (jako provozní profily).
- Metodické přístupy, např. založené na selháních (zahrnující odhadování druhu chyb a útoky na vady), založené na zkušenostech, kontrolních seznamech a kvalitativních vlastnostech.
- Přístupy založené na procesech nebo odpovídající standardům (například standardům specifických pro průmysl) nebo různé agilní metody.
- Dynamické a heuristické přístupy, například průzkumné testování, které spíše reaguje na události namísto toho, aby bylo předem plánované; zde jsou provádění a vyhodnocování (testů) současně bežícími úlohami.
- Konzultativní přístupy, například takové, kde je pokrytí testy primárně řízeno doporučeními a radami expertů z technologických a/nebo obchodních kruhů mimo testovací tým.
- Regresně-averzní přístupy, které zahrnují opětovné použití existujícího testovacího materiálu, rozsáhlou automatizaci funkcionálních regresních testů a standardní testovací sady.

Je možné kombinovat různé přístupy, například dynamický přístup založený na rizicích.

5.3 *Sledování postupu testování a a řízení postupu testování (Z2)*	*20 minut*

Základní výrazy
Hustota defektů, míra selhání, řízení testování, sledování testů, souhrnná zpráva z testování.

5.3.1 Sledování postupu testování (Z1)

Účelem sledování testování je poskytnout zpětnou vazbu a zviditelnit testovací činnosti. Monitorované informace mohou být sbírány manuálně nebo automaticky a mohou být použity na měření výstupních kriterií, jako je například mapa pokrytí. Metriky mohou být také použity na zhodnocení postupu vůči plánovanému harmonogramu a rozpočtu. Běžné testovací metriky zahrnují:

- Procentuální podíl vykonané práce v přípravě testovacích případů (nebo procentuální podíl připravených testovacích případů z plánovaných).
- Procentuální podíl vykonané práce v přípravě testovacího prostředí.
- Spouštění testovacích případů (např. počet vykonaných/nevykonaných testovacích případů nebo počet testovacích případů vykonaných úspěšně/neúspěšně).
- Informace o defektech (např. hustota defektů, nalezené a opravené defekty, míra selhání a výsledky retestů).
- Pokrytí testování požadavků, rizik nebo kódu.
- Subjektivní důvěra testerů v daný produkt.
- Data testovacích milníků.
- Náklady na testování, včetně nákladů v porovnání vůči přínosům nalezení dalšího defektu nebo spuštění dalšího testu.

5.3.2 Reportování z testování (Z2)

Reportování z testování se zabývá sumárními informacemi o testovacích aktivitách. Zahrnuje:

- Co se událo během období testování, například datumy, kdy byla splněna výstupní kritéria.
- Analyzované informace a metriky na podporu doporučení a rozhodnutí o následujících krocích, jako jsou odhad zbývajících defektů, ekonomické přínosy pokračujícího testování, zbývající rizika a úroveň důvěry v testovaný software.

Vzor souhrnné zprávy z testování se nachází ve Standardu pro dokumentaci testování softwaru (IEEE Std 829-1998).

Metriky by měly být sbírány během úrovně testování s cílem zhodnotit na jejím konci:

- Adekvátnost cílů testování pro úroveň testování.
- Adekvátnost zvolených přístupů testování.
- Efektivnost testování s ohledem na jeho cíle.

5.3.3 Řízení testování (Z2)

Řízení testování popisuje veškeré usměrňující nebo korekční činnosti, které byly vykonány na základě získaných a reportovaných informací a metrik. Činnosti mohou pokrývat jakékoliv testovací aktivity a mohou ovlivnit jakékoliv aktivity nebo úkoly životního cyklu softwaru.

Činnosti řízení testování zahrnují například:

- Vykonávání rozhodnutí založených na informacích z monitoringu testů.
- Změny priorit testů v případě výskytu identifikovaného rizika (např. pozdní dodávka softwaru).
- Změny testovacího harmonogramu s ohledem na dostupnost nebo nedostupnost testovacího prostředí.
- Nastavování vstupních kritérií, kdy je vyžadováno přetestování (konfirmační testování) oprav vývojářem předtím, než budou akceptovány do buildu.

5.4 *Konfigurační management (Z2)*	*10 minut*

Základní výrazy
Konfigurační management, řízení verzí.

Pozadí
Účelem konfiguračního managementu je vytvořit a udržovat integritu softwarových produktů (komponenty, dat a dokumentace) nebo systému v průběhu projektového a produktového životního cyklu.

Pro testování může konfigurační management zajistit:
- Aby všechny položky testwaru byly identifikované, verze byly řízené, změny sledované, vzájemná propojení a propojení s vývojovými položkami (objekty testování) existovala tak, aby bylo možné udržet trasovatelnost během celého procesu testování.
- Aby se na všechny identifikované dokumenty a položky softwaru dalo jednoznačně odkázat v testovací dokumentaci.

Testerům konfigurační management pomáhá jednoznačně identifikovat (a reprodukovat) testované položky, testovací dokumenty, testy a testovací prostředky (test harness).

Procedury a infrastruktura (nástroje) pro konfigurační management by měly být zvoleny, dokumentovány a implementovány v průběhu plánování testů.

5.5 *Riziko a testování (Z2)*	*30 minut*

Základní výrazy
Produktové riziko, projektové riziko, riziko, testování založené na rizicích.

Pozadí
Riziko může být definované jako možnost výskytu události, hrozby, nebezpečí nebo situace, která vede k nežádoucím následkům nebo k potenciálním problémům. Úroveň rizika se určuje podle pravděpodobnosti výskytu nepříznivé události a dopadu (škody vyplývající z této události).

5.5.1 Projektová rizika (Z2)
Projektová rizika jsou rizika ohraničující schopnost projektu dosáhnout svých cílů, například:

- Organizační faktory:
  - nedostatek schopností, školení a personálu;
  - personální problémy;
  - sociální („politické“) problémy, jako například:
    - problémy s testery komunikujícími své potřeby a výsledky testů;
    - selhání týmu při využití informací vyplývajících z testování a revizí (např. nezlepšování vývojových a testovacích praktik).
  - nevhodný postoj k testování nebo nevhodné očekávání od testování (např. nedoceněná hodnota nalezení defektů v průběhu testování).
- Technické problémy:
  - problémy v definování správných požadavků;
  - rozsah, ve kterém nemohou být požadavky naplněny při existujících omezeních;
  - nepřipravenost testovacího prostředí;
  - pozdní konverze dat, pozdní plánování migrace a vývoje, pozdní příprava nástrojů na konverzi/migraci testovacích dat;
  - nízká kvalita návrhu, kódu, konfiguračních dat, testovacích dat a testů.
- Dodavatelské problémy:
  - selhání třetí strany;
  - smluvní problémy.

Pokud jsou tato rizika analyzována, řízena a zmírňována, manažer testování postupuje podle dobře zavedených principů projektového řízení. Vzor testovacího plánu ze Standardu pro dokumentaci testování softwaru (IEEE Std 829-1998) požaduje stanovit rizika a záložní plány.

5.5.2 Produktová rizika (Z2)
Oblasti potenciálních selhání (nepříznivé budoucí události nebo nebezpečí) v softwaru nebo systému jsou známé jako produktová rizika. Jsou to rizika ohrožující kvalitu produktu a zahrnují:

- Dodaný software je náchylný k chybám.
- Možnost, že software/hardware poškodí jednotlivce nebo organizaci.
- Nedostatečné vlastnosti softwaru (např. funkcionalita, spolehlivost, použitelnost a výkonnost).
- Slabá integrita a kvalita dat (např. problémy migrace, konverze a přenosu dat, nedodržení datových standardů).
- Software, který nevykonává zamýšlenou funkcionalitu.

Rizika se využívají pro rozhodování, kde začít testovat a kde testovat více; testování se využívá ke snížení rizika výskytu nepříznivé události nebo ke snížení dopadu nepříznivé události.

Produktová rizika jsou speciálním typem rizik ohrožujících úspěch projektu. Testování jako aktivita kontrolující riziko poskytuje zpětnou vazbu o reziduálním riziku měřením efektivnosti odstraňování kritických defektů a záložních plánů.
Přístup k testování založený na rizicích poskytuje příležitosti proaktivně redukovat úroveň produktového rizika, počínaje úvodními stádii projektu. Zahrnuje identifikaci produktových rizik a jejich aplikaci při využití plánování a řízení testování, specifikaci, přípravě a provedení testů. V přístupu založeném na rizicích mohou být identifikovaná rizika použita na:

- Určení testovacích technik, které budou použity.
- Určení rozsahu testování, který má být vykonán.
- Stanovení priorit testování s cílem najít kritické defekty tak brzy, jak je to jen možné.
- Určení, zda mohou být na redukování rizika použity některé aktivity, které nesouvisí přímo s testováním (např. poskytnutí školení pro nezkušené návrháře).

Testování založené na rizicích vychází z kolektivních vědomostí a přehledu zainteresovaných osob na projektu. Jeho cílem je určit rizika a úroveň testů požadovanou na vypořádání se s těmito riziky.

K zajištění toho, aby pravděpodobnost selhání produktu byla minimalizována, poskytují aktivity managementu rizik disciplinovaný přístup k:

- Zhodnocení (a pravidelnému přehodnocení) toho, co se může vyvíjet nepříznivě (rizika).
- Určení, kterými riziky je důležité se zabývat.
- Provedení činností sloužících k vypořádání se s riziky.

Testování může navíc podporovat identifikaci nových rizik, může pomáhat určit rizika, která by měla být redukována, a může snížit nejistotu vzhledem k rizikům.

5.6 *Správa incidentů (Z3)*	*40 minut*

Základní výrazy
Zaznamenávání incidentů, správa incidentů, záznam o incidentu.

Pozadí
Vzhledem k tomu, že jedním z cílů testování je najít defekty, musí být rozdíly mezi skutečnými a očekávanými výsledky zaznamenány jako incidenty. Incident je třeba prozkoumat. Může se ukázat, že se jedná o defekt. Pro řešení incidentů a defektů je třeba definovat adekvátní úkony. Incidenty a defekty je třeba sledovat od svého nalezení a klasifikace, po opravu a potvrzení řešení. Za účelem řídit všechny incidenty až do jejich vyřešení by organizace měla zavést proces správy incidentů a pravidla pro jejich klasifikaci.

Incidenty mohou být objeveny během vývoje, revidování, testování nebo používání softwarového produktu. Mohou vzniknout z důvodu problémů v kódu nebo provozním systému nebo v jakémkoliv typu dokumentace včetně požadavků, vývojové dokumentace, dokumentů testování a uživatelských informací jako nápověda nebo instalační manuál.

Záznamy o incidentu mají následující cíle:
- Poskytnout vývojářům a jiným stranám zpětnou vazbu o problému s cílem umožnit identifikaci, izolaci a opravu podle potřeby.
- Poskytnout vedoucím testování prostředek pro sledování kvality testovaného systému a postupu testování.
- Poskytnout nápady pro zlepšení procesu testování.

Detaily záznamu o incidentu by měly zahrnovat:
- Datum vydání, organizaci a autora záznamu.
- Očekávaný a skutečný výsledek.
- Identifikaci předmětu testování (konfigurační položky) a prostředí.
- Proces životního cyklu softwaru nebo systému, ve kterém byl incident zpozorován.
- Popis incidentu za účelem reprodukce a řešení, může obsahovat logy, dumpy databází nebo screenshoty.
- Rozsah nebo stupeň dopadu na zájmy zainteresovaných osob na projektu.
- Závažnost dopadu na systém.
- Naléhavost/priorita opravy.
- Stav incidentu (např. otevřený, odložený, duplicitní, čekající na opravu, opravený a čekající na potvrzení nebo uzavřený).
- Závěry, doporučení a schválení.
- Globální problémy, jako například jiné oblasti, které mohou být ovlivněny změnou vyvolanou incidentem.
- Historie změn, jako například sled akcí vykonaných členy projektového týmu v souvislosti s incidentem, s cílem incident izolovat, opravit a potvrdit, že byl opraven.
- Odkazy, například na specifikaci testovacího případu, který odhalil problém.

Struktura záznamu o incidentu je taktéž zahrnuta v Standardu pro dokumentaci testování softwaru (IEEE Std 829-1998).

Reference
5.1.1 Black, 2001, Hetzel, 1988
5.1.2 Black, 2001, Hetzel, 1988
5.2.5 Black, 2001, Craig, 2002, IEEE Std 829-1998, Kaner 2002

5.3.3 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE Std 829-1998
5.4 Craig, 2002
5.5.2 Black, 2001, IEEE Std 829-1998
5.6 Black, 2001, IEEE Std 829-1998

6. Podpůrné nástroje pro testování (Z2) — 80 minut

Studijní cíle pro podpůrné nástroje pro testování

Cíle identifikují, co budete umět po dokončení každého modulu.

6.1 Typy testovacích nástrojů (Z2)

SC-6.1.1 Klasifikovat různé typy testovacích nástrojů podle jejich účelu, aktivit základního testovacího procesu a životního cyklu softwaru. (Z2)
SC-6.1.3 Vysvětlit pojem testovací nástroj a účel podpory nástrojů při testování. (Z2)[2]

6.2 Efektivní používání nástrojů: možné výhody a rizika (Z2)

SC-6.2.1 Shrnout možné výhody a rizika automatizace testů a podpůrných nástrojů pro testování. (Z2)
SC-6.2.2 Zapamatovat si specifika nástrojů pro provedení testů, nástrojů pro statickou analýzu a nástrojů pro management testování. (Z1)

6.3 Zavedení nástroje v organizaci (Z1)

SC-6.3.1 Uvést hlavní principy zavedení nástroje v organizaci. (Z1)
SC-6.3.2 Určit cíle zkoušky konceptu (proof of concept) pro vyhodnocení nástroje a pilotní fáze pro implementaci nástroje. (Z1)
SC-6.3.3 Pochopit, že pro zajištění dostatečné podpory nástroje nepostačuje pouze jeho pořízení, ale jsou potřebná i další podpůrná opatření. (Z1)

[2] SC-6.1.2. Záměrně přeskočeno

6.1 *Typy testovacích nástrojů (Z2)*	*45 minut*

Základní výrazy
Nástroj pro konfigurační management, nástroj pro pokrytí, nástroj pro ladění, nástroj pro dynamickou analýzu, nástroj pro správu incidentů, nástroj pro zátěžové testování, modelovací nástroj, monitorovací nástroj, nástroj pro testování výkonnosti, efekt měřícího zařízení, nástroj pro správu požadavků, revizní nástroj, bezpečnostní nástroj, nástroj statické analýzy , nástroj pro testy extrémní zátěže, komparátor testu, nástroj pro přípravu testovacích dat, nástroj pro návrh testů, testovací prostředky (test harness), nástroj pro provedení testů, nástroj pro management testování, testovací framework jednotkového testování.

6.1.1 Podpůrné nástroje pro testování (Z2)

Testovací nástroje mohou být použity pro různé aktivity podporující testování, a neomezují se nutně pouze na jednu. Tyto aktivity zahrnují:
1. Nástroje, které jsou přímo použity v testování, jako například nástroje pro spouštění testů, nástroje pro generování testovacích dat a nástroje pro porovnávání výsledků.
2. Nástroje, které pomáhají při řízení procesu testování, jako například nástroje používané pro management testů, pro organizaci výsledků testů, dat, požadavků, správu incidentů, defektů atd. a pro reportování a monitorování provedení testů.
3. Nástroje, které jsou použity k prozkoumávání (např. nástroje, které monitorují aktivitu souborů pro aplikaci).
4. Jakýkoliv nástroj, který napomáhá v testování (v tomto významu je tabulkový procesor též testovacím nástrojem).

Podpůrný nástroj pro testování může mít různé účely v závislosti na kontextu, jako například:
- Zlepšení účinnosti testovacích aktivit automatizací opakovaných úkolů nebo podporou manuálních testovacích aktivit, jako například plánování testování, návrh testů, reportování a monitorování testování.
- Automatizace aktivit, které vyžadují významné množství zdrojů v případě, že jsou prováděny manuálně (např. statické testování).
- Automatizace aktivit, které nemohou být prováděny manuálně (např. testování výkonnosti velkého rozsahu klient-server aplikací).
- Zvýšení spolehlivosti testování (např. automatizací porovnávání velkého množství dat anebo simulací chování).

Pojem „framework testování“ se také často používá v oboru, a to nejméně ve třech významech:
- Znovupoužitelné a rozšiřitelné testovací knihovny, které mohou být použity na vytvoření nástrojů pro testování (taktéž nazývané nástroj pro provedení testů).
- Typ návrhu automatizace testů (např. řízený daty, řízený klíčovými slovy).
- Celkový proces provedení testů.

Pro účely těchto učebních osnov je pojem „framework testování“ použit v jeho prvních dvou významech, tak jak to je popsané v kapitole 6.1.6.

6.1.2 Klasifikace testovacích nástrojů (Z2)

Existuje řada nástrojů, které podporují různé oblasti testování. Nástroje mohou být klasifikovány na základě více kriterií, jakými jsou účel nástroje, jeho dostupnost (komerční, volně dostupný, open-source nebo shareware), použitá technologie a tak dále. Nástroje v těchto osnovách jsou rozděleny podle testovacích aktivit, které podporují.

Některé nástroje jasně podporují jednu aktivitu. Jiné mohou podporovat více než jednu aktivitu. Ty jsou klasifikovány v rámci aktivity, se kterou jsou nejvíce spojené. Nástroje od jednoho poskytovatele, (obzvláště ty, které byly navrhnuty, aby pracovaly spolu) mohou být součástí jednoho balíku.

Některé typy testovacích nástrojů mohou být rušivé, což znamená, že mohou ovlivnit skutečný výsledek testu. Například naměřený čas může být odlišný kvůli instrukcím navíc, které musí nástroj vykonat, nebo je možné získat rozdílnou míru pokrytí kódu. Následek rušivých nástrojů nazýváme efekt měřícího zařízení.

Některé nástroje nabízejí podporu vhodnou spíše pro vývojáře (např. nástroje, které jsou použity v průběhu testování komponent nebo integračního testování komponent). Takovéto nástroje jsou v rozdělení uvedeném níže označeny písmenem “V”.

6.1.3 Podpůrné nástroje pro management testování a testů (Z1)

Nástroje pro management se používají pro všechny testovací aktivity v průběhu celého životního cyklu softwaru.

Nástroje pro management testování
Tyto nástroje poskytují rozhraní pro provedení testů, sledování defektů a správu požadavků spolu s podporou pro kvantitativní analýzu a reportování objektů testování. Rovněž podporují trasování těchto objektů, testování vzhledem ke specifikacím požadavků a mohou mít schopnost nezávislé správy verzí, anebo poskytovat rozhraní k externímu nástroji.

Nástroje pro správu požadavků
Tyto nástroje uchovávají popisy požadavků, uchovávají atributy pro požadavky (včetně priority), poskytují jedinečné identifikátory a podporují trasování těchto požadavků k jednotlivým testům. Tyto nástroje mohou taktéž pomoci identifikovat nekonzistentní nebo chybějící požadavky.

Nástroje pro správu incidentů (Nástroje pro sledování defektů)
Tyto nástroje uchovávají a řídí záznamy o incidentech, tj. defektech, selháních, požadavcích na změnu nebo pozorovaných problémech a anomáliích a pomáhají při řízení životního cyklu incidentů, případně s podporou pro statistickou analýzu.

Nástroje pro konfigurační management
I když tyto nástroje nejsou vysloveně testovacími nástroji, jsou přesto nezbytné pro uchovávání a správu verzí testwaru. Tyto nástroje se uplatňují v případě, kdy se konfigurují prostředí sestávající z více než jedné hardwarové nebo softwarové komponenty ve smyslu verzí operačního systému, překladačů, prohlížečů, atd.

6.1.4 Podpůrné nástroje pro statické testování (Z1)

Nástroje pro statické testování poskytují nákladově efektivní způsob nacházení většího počtu defektů v raném stádiu procesu vývoje.

Revizní nástroje
Tyto nástroje pomáhají s procesy revize, kontrolními seznamy, revizními směrnicemi a jsou používány pro uchovávání a komunikaci připomínek revize, záznamů o defektech a o pracnosti. Mohou být dále nápomocné při „online“ revizích pro velké nebo geograficky rozptýlené týmy.

Nástroje statické analýzy (V)
Tyto nástroje pomáhají vývojářům a testerům při nacházení defektů před dynamickým testováním prostřednictvím prosazování standardů testování (zahrnující bezpečné kódování) a analýzu struktur a závislostí. Mohou též pomoci při plánování anebo analýze rizik poskytováním metrik pro kód (např. složitost).

Modelovací nástroje (V)
Tyto nástroje jsou používány pro validaci modelů softwaru (např. fyzický datový model (PDM) pro relační databázi) vyjmenováváním nekonzistencí a hledáním defektů. Tyto nástroje mohou často pomoci při generování některých testovacích případů založených na modelech.

6.1.5 Podpůrné nástroje pro specifikaci testů (Z1)

Nástroje pro návrh testů
Tyto nástroje se používají pro generování testovacích vstupů nebo spustitelných testů a/nebo testovacích prognóz z požadavků, z uživatelských grafických rozhraní, z návrhových modelů (stavový, datový, objektový) nebo z kódu.

Nástroje pro přípravu testovacích dat
Nástroje pro přípravu testovacích dat manipulují s databázemi, soubory nebo přenosy dat takovým způsobem, aby připravily testovací data k použití během provedení testů za účelem zabezpečení prostřednictvím anonymity dat.

6.1.6 Podpůrné nástroje pro vykonání a zaznamenávání testů (Z1)

Nástroje pro provedení testů
Tyto nástroje umožňují automatické nebo poloautomatické provedení testů, a to použitím uložených vstupů a očekávaných výstupů s využitím skriptovacího jazyka a obvykle poskytují záznam testování pro každý testovací běh. Mohou být též použity na nahrávání testů a obvykle podporují skriptovací jazyky nebo konfiguraci založenou na GUI pro parametrizaci dat a možnost dalších úprav v testech.

Testovací prostředky (test harness) / framework jednotkového testování (V)
Testovací prostředky (test harness) nebo frameworky jednotkového testování usnadňuje testování komponent nebo částí systému simulováním prostředí, ve kterém bude běžet objekt testování, a to prostřednictvím poskytnutí simulovaných objektů jako stubu nebo ovladače.

Komparátory testů
Komparátory testů určují rozdíly mezi soubory, databázemi nebo výsledky testování. Nástroje pro provedení testů obyčejně zahrnují dynamické komparátory, nicméně porovnávání po vykonání může být uskutečněno v nezávislém porovnávacím nástroji. Komparátor testů může využívat testovací prognózu, a to především v případě, kdy je test automatizovaný.

Nástroje na pokrytí (V)
Tyto nástroje měří procento specifických typů struktur kódu (např. příkazů, větví nebo rozhodování, a modulů nebo volání funkcí) prozkoušených sadou testů za pomoci rušivých a nerušivých prostředků.

Bezpečnostní nástroje
Tyto nástroje jsou používány pro vyhodnocování bezpečnostních charakteristik softwaru. To zahrnuje vyhodnocování schopnosti softwaru ochránit důvěrnost dat, integritu, autentifikaci, autorizaci, dostupnost a nepopíratelnost. Bezpečnostní nástroje jsou většinou zaměřeny na příslušnou technologii, platformu nebo účel.

6.1.7 Podpůrné nástroje pro výkonnost a monitorování (Z1)

Nástroje pro dynamickou analýzu (V)
Nástroje pro dynamickou analýzu nacházejí defekty, které jsou zjevné, až když je software používán – například časové závislosti nebo přetečení paměti. Jsou obyčejně používány při testování komponent, integračním testování komponent a při testování střední vrstvy (middleware).

Nástroje pro testování výkonnosti/zátěžové testování/stres testování
Nástroje pro testování výkonnosti sledují a oznamují chování systému v různých simulovaných podmínkách používání ve smyslu počtu souběžně pracujících uživatelů, modelu jejich růstu, frekvence

a relativního procenta transakcí. Simulace zátěže je dosažena způsobem vytváření virtuálních uživatelů, kteří vykonávají vybranou sadu transakcí rozloženou napříč různými testovacími automaty. Tyto automaty se obvykle nazývají generátory zátěže.

Monitorovací nástroje
Monitorovací nástroje nepřetržitě analyzují, verifikují a reportují využívání specifických systémových zdrojů a varují před možnými provozními problémy.

6.1.8 Podpůrné nástroje pro specifické oblasti testování (Z1)

Hodnocení kvality dat
Data jsou klíčovým aspektem některých projektů, jakými jsou projekty datové konverze/migrace a aplikací jako datové sklady. Jejich atributy se mohou lišit ve smyslu závažnosti a objemu. V takovém kontextu je potřebné nasadit nástroje pro hodnocení kvality dat za účelem revize a ověření konverze dat a migračních pravidel tak, aby bylo zajištěno, že zpracovaná data jsou správná, kompletní a že jsou v souladu s předdefinovaným standardem, který je specifický pro daný kontext.

Existují i další nástroje pro podporu testování použitelnosti.

6.2 *Efektivní použití nástrojů: možné výhody a rizika (Z2)*	*20 minut*

Základní výrazy
Testování řízené daty, testování řízené klíčovými slovy, skriptovací jazyk.

6.2.1 Možné výhody a rizika nástroje pro podporu testování (pro všechny nástroje) (Z2)

Prostý nákup nebo pronájem nástroje úspěch s daným nástrojem negarantuje. Každý typ nástroje může vyžadovat dodatečné úsilí pro dosažení skutečných a dlouhotrvajících výhod. S použitím nástroje přicházejí určité potenciální výhody a příležitosti, ale existují také rizika.

Možné výhody používání nástrojů zahrnují:
- Opakované činnosti jsou redukovány (např. běhy regresních testů, opětovné zadávání stejných testovacích dat, kontrola standardů kódování).
- Větší konzistence a opakovatelnost (např. testy vykonávané nástrojem ve stejném pořadí a se stejnou frekvencí a testy odvozené z požadavků).
- Objektivní hodnocení (např. statické měření, pokrytí).
- Jednoduchý přístup k informacím o testech nebo testování (statistiky a grafy postupu testování, poměry incidentů a výkonnost).

Rizika používání nástrojů zahrnují:
- Nerealistické očekávání od nástroje (včetně funkcionality a jednoduchosti používání).
- Podcenění času, nákladů a pracnosti zavádění nástroje (včetně školení a externí podpory).
- Podcenění času a úsilí, které jsou potřebné pro dosažení významných a trvalých výhod nástroje (včetně potřeby změny procesů testování a kontinuálního zlepšování způsobů použití nástroje).
- Podcenění úsilí vyžadovaného na údržbu testovacích produktů generovaných nástrojem.
- Přehnané spoléhání se na nástroj (náhrada návrhu testů nebo automatizace testů v případě, kdy by manuální testování bylo vhodnější).
- Zanedbání kontroly verzí artefaktů testování v daném nástroji.
- Zanedbání vztahů a spolupůsobení problémů mezi kriticky důležitými nástroji, jako jsou nástroje pro správu požadavků, nástroje pro konfigurační management, nástroje pro správu incidentů, nástroje pro sledování defektů a nástroje od různých prodejců.
- Riziko spojené s dodavatelem, který nepokračuje v dané podnikatelské oblasti, zastaralost nástroje nebo prodej nástroje jinému dodavateli.
- Nedostatečná reakce ze strany dodavatele s ohledem na podporu, aktualizace a opravy chyb v nástroji.
- Riziko přerušení projektu vývoje u open-source nebo volně dostupných nástrojů.
- Nepředvídatelná rizika, jako například neschopnost podporovat nové platformy.

6.2.2 Specifické úvahy k některým typům nástrojů (Z1)

Nástroje pro provedení testů
Nástroje pro provedení testů spouští testované objekty s použitím automatizovaných testovacích skriptů. Tento typ nástroje často vyžaduje k dosažení podstatných výhod značné úsilí.

Zaznamenávání testů nahráváním akcí manuálního testera vypadá atraktivně, ale tento přístup přestává být efektivní při velkém množství automatizovaných testovacích skriptů. Zaznamenaný skript je

přímočarou reprezentací se specifickými daty a akcemi, které jsou součástí každého skriptu. Tento typ skriptu může být nestabilní v případě výskytu neočekávaných událostí.

Přístup testování řízeného daty typicky používá testovací vstupy (data) v tabulkovém procesoru a používá obecnější testovací skript, který umí tato vstupní data číst a vykonat tak stejný skript s různými daty. Testeři, kteří nejsou obeznámení se skriptovacím jazykem, mohou potom vytvářet data pro tyto předdefinované skripty.

Existují i další techniky použité v rámci technik řízenými daty, kde namísto kombinace napevno nakódovaných dat umístěných v tabulkovém procesoru, jsou data generována použitím algoritmů založených na konfigurovatelných parametrech v čase běhu a jsou poskytována aplikaci. Například nástroj může používat algoritmus, který generuje náhodné uživatelské ID a kvůli opakovatelnosti ve vzorku se pro kontrolu náhodnosti použije tzv. random seed (hodnota počáteční inicializace generátoru náhodných čísel, vzniklá např. z aktuálního času nebo z pohybu myší).

V případě přístupu testování řízeného klíčovými slovy obsahuje tabulkový procesor klíčová slova popisující akce, které mají být vykonány (taktéž nazývány jako “akční slova”) a testovací data. Testeři (i když nejsou obeznámeni se skriptovacím jazykem) mohou definovat testy použitím klíčových slov, která mohou být přizpůsobena na míru testované aplikaci.

Technická znalost skriptovacích jazyků je potřebná při všech přístupech (buď u testerů nebo u specialistů na automatizaci testování).

Bez ohledu na použitou techniku skriptování musí být kvůli pozdějšímu porovnání uloženy očekávané výsledky pro každý test.

Nástroj statické analýzy
Nástroje statické analýzy nasazené na zdrojový kód mohou vynucovat dodržování standardů kódování, ale v případě aplikace na existující kód mohou generovat značné množství zpráv. Varovné zprávy nezastaví překlad kódu do vykonatelného programu, ale v ideálním případě by neměly zůstat nepovšimnuty, protože to ulehčí údržbu kódu v budoucnosti. Efektivním přístupem implementace nástroje pro analýzu je postupná implementace s úvodní filtrací, která vylučuje některé typy zpráv.

Nástroje pro management testování
Nástroje pro management testování vyžadují rozhraní k ostatním nástrojům nebo tabulkovým procesorům za účelem produkce užitečných informací ve formátu, který je vhodný pro potřeby organizace.

6.3 *Zavedení nástroje v organizaci (Z1)*	*15 minut*

Základní výrazy
Žádné specifické výrazy.

Pozadí
Mezi nejdůležitější úvahy při výběru nástroje pro organizaci patří:
- Posouzení zralosti organizace, silných a slabých stránek spolu s identifikací možností zlepšení testovacího procesu s podporou nástrojů.
- Vyhodnocení vůči jasným požadavkům a objektivním kritériím.
- Zkouška konceptu (proof-of-concept) použitím nástroje pro testování v průběhu vyhodnocovací fáze ke stanovené efektivního fungování s testovaným softwarem v současné infrastruktuře, anebo k identifikaci změny, které tato infrastruktura potřebuje k efektivnímu použití nástroje.
- Vyhodnocení dodavatele (včetně školení, podpory a komerčních stránek) anebo poskytovatelů podpůrných služeb v případě nekomerčních nástrojů.
- Identifikace interních požadavků pro koučování a mentorování při používání nástroje.
- Vyhodnocení potřeb na školení s ohledem na současné zkušenosti testovacího týmu s automatizací testů.
- Vyhodnocení poměru nákladů a výnosů, které je založeno na konkrétním obchodním případu.

Zavedení zvoleného nástroje do organizace začíná pilotním projektem, který má následující cíle:
- Dozvědět se více podrobností o nástroji.
- Zhodnotit, jak bude nástroj zapadat do existujících procesů a praktik, a určit, co bude potřebné změnit.
- Rozhodnout o standardním způsobu používání, správě, zálohování a údržbě nástroje a testovacích produktů (tj. dohoda o jmenné konvenci pro soubory a testy, vytváření knihoven a definice modularity testovacích sad).
- Posouzení, zda budou výhody dosaženy za přiměřené náklady.

Faktory úspěchu pro zavedení nástroje v rámci organizace zahrnují:
- Postupné zavádění nástroje do zbytku organizace.
- Adaptace a zlepšování procesů tak, aby odpovídaly použití nástroje.
- Zabezpečování školení a koučování/mentorování pro nové uživatele.
- Definice pravidel používání.
- Implementování způsobu získávání informací z jeho aktuálního používání.
- Monitorování používání nástroje a hodnocení jeho výhod.
- Poskytování podpory testovacímu týmu pro daný nástroj.
- Shromažďování ponaučení získaných všemi týmy.

Reference
6.2.2 Buwalda, 2001, Fewster, 1999
6.3 Fewster, 1999

7. Reference

Standardy

ISTQB Glosář pojmů používaných v testování softwaru, verze 3.01.

[CMMI] Chrissis, M.B., Konrad, M. and Shrum, S. (2004) *CMMI, Guidelines for Process Integration and Product Improvement*, Addison Wesley: Reading, MA
Viz kapitola 2.1

[IEEE Std 829-1998] IEEE Std 829™ (1998) IEEE Standard for Software Test Documentation
Viz kapitoly 2.3, 2.4, 4.1, 5.2, 5.3, 5.5, 5.6

[IEEE 1028] IEEE Std 1028™ (2008) IEEE Standard for Software Reviews and Audits
Viz kapitola 3.2

[IEEE 12207] IEEE 12207/ISO/IEC 12207-2008, Software life cycle processes
Viz kapitola 2.1

[ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality
Viz kapitola 2.3

Literatura

[Beizer, 1990] Beizer, B. (1990) Software Testing Techniques (2nd edition), Van Nostrand Reinhold: Boston
Viz kapitoly 1.2, 1.3, 2.3, 4.2, 4.3, 4.4, 4.6

[Black, 2001] Black, R. (2001) Managing the Testing Process (3rd edition), John Wiley & Sons: New York
Viz kapitoly 1.1, 1.2, 1.4, 1.5, 2.3, 2.4, 5.1, 5.2, 5.3, 5.5, 5.6

[Buwalda, 2001] Buwalda, H. et al. (2001) Integrated Test Design and Automation, Addison Wesley: Reading, MA
Viz kapitola 6.2

[Copeland, 2004] Copeland, L. (2004) A Practitioner's Guide to Software Test Design, Artech House: Norwood, MA
Viz kapitoly 2.2, 2.3, 4.2, 4.3, 4.4, 4.6

[Craig, 2002] Craig, Rick D. and Jaskiel, Stefan P. (2002) Systematic Software Testing, Artech House: Norwood, MA
Viz kapitoly 1.4.5, 2.1.3, 2.4, 4.1, 5.2.5, 5.3, 5.4

[Fewster, 1999] Fewster, M. and Graham, D. (1999) Software Test Automation, Addison Wesley: Reading, MA
Viz kapitoly 6.2, 6.3

[Gilb, 1993]: Gilb, Tom and Graham, Dorothy (1993) Software Inspection, Addison Wesley: Reading, MA
Viz kapitoly 3.2.2, 3.2.4

[Hetzel, 1988] Hetzel, W. (1988) Complete Guide to Software Testing, QED: Wellesley, MA
Viz kapitoly 1.3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 4.1, 5.1, 5.3

[Kaner, 2002] Kaner, C., Bach, J. and Pettticord, B. (2002) Lessons Learned in Software Testing, John Wiley & Sons: New York
Viz kapitoly 1.1, 4.5, 5.2

[Myers 1979] Myers, Glenford J. (1979) The Art of Software Testing, John Wiley & Sons: New York
Viz kapitoly 1.2, 1.3, 2.2, 4.3

[van Veenendaal, 2004] van Veenendaal, E. (ed.) (2004) The Testing Practitioner (Chapters 6, 8, 10), UTN Publishers: The Netherlands
Viz kapitoly 3.2, 3.3

8. Příloha A – Pozadí učebních osnov

Historie dokumentu

Tento dokument byl připraven během let 2004–2011 pracovní skupinou skládající se ze jmenovaných členů Mezinárodního výboru pro kvalifikaci testování softwaru (International Software Testing Qualifications Board ISTQB). Nejdřív byl dokument revidován výborem pro revizi a následně představiteli vybranými z mezinárodní komunity testování softwaru. Pravidla použitá při tvorbě tohoto dokumentu jsou uvedena v příloze C.

Tento dokument je učební osnovou pro mezinárodní Základní certifikát v testování softwaru, který je první úrovní mezinárodní kvalifikace odsouhlasenou ISTQB (www.istqb.org).

Cíle kvalifikace Základní certifikát

- Získat uznání testování jako základní a profesionální specializace softwarového inženýrství.
- Poskytnout standardní zázemí pro kariérní rozvoj testerů.
- Umožnit profesionálně kvalifikovaným testerům, aby byli respektováni zaměstnavateli, zákazníky a kolegy, a zároveň zlepšit postavení testerů.
- Prosazovat jednotné a vhodné testovací postupy v rámci všech disciplín softwarového inženýrství.
- Identifikovat oblasti testování, které jsou podstatné a přínosné pro dané odvětví.
- Umožnit dodavatelům softwaru, aby přijímali certifikované testery, a tak získali obchodní výhodu vůči své konkurenci zveřejněním svých pravidel náboru testerů.
- Poskytnout možnost pro testery a další, kteří mají zájem o testování, aby získali mezinárodně uznávanou kvalifikaci v předmětu testování.

Cíle mezinárodní kvalifikace (převzaté z ISTQB setkání v Sollentuna, Listopad 2001)

- Umožnit porovnání znalostí testování v různých zemích.
- Umožnit testerům jednodušší uplatnění v jiných zemích.
- Umožnit nadnárodním/mezinárodním projektům, aby měly jednotné chápání problematiky testování.
- Celosvětově zvyšovat množství kvalifikovaných testerů.
- Jako mezinárodně založená iniciativa mít vyšší vliv a hodnotu než lokální přístup v rámci jednoho regionu/země.
- Rozvíjet společný mezinárodní soubor poznatků a vědomostí o testování pomocí učebních osnov a terminologie a zvyšovat úroveň znalostí o testování u všech zúčastněných.
- Propagovat testování jako profesi v dalších zemích.
- Umožnit testerům, aby získali uznávanou kvalifikaci ve svém rodném jazyce.
- Umožnit sdílení znalostí a zdrojů mezi zeměmi.
- Poskytovat mezinárodní uznání testerů a jejich kvalifikace díky zapojení zemí.

Vstupní požadavky na tuto kvalifikaci

Vstupní kritérium pro získání Základního certifikátu ISTQB® v testování softwaru je zájem kandidátů o testování softwaru. Avšak je silně doporučeno, aby kandidáti taktéž:

- Měli přinejmenším minimální znalosti z vývoje softwaru nebo testování softwaru, například šestiměsíční zkušenost jako tester systémových nebo akceptačních testů, případně jako vývojář softwaru.
- Absolvovali školení, které je akreditováno podle ISTQB® standardů (jedním z uznaných lokálních výborů ISTQB).

Pozadí a historie Základního certifikátu v testování softwaru

Nezávislá certifikace testerů softwaru začala ve Velké Británii ve Zkušebním výboru informačních systémů (Information Systems Examination Board ISEB) založením Výboru pro testování softwaru (Software testing board) v roce 1998 (www.bcs.org.uk/iseb). V roce 2002 začala společnost ASQF v Německu podporovat Německý program pro kvalifikaci testerů (German tester qualification scheme www.asqf.de). Tyto učební osnovy jsou založené na osnovách ISEB a ASQF; zahrnují přestrukturovaný aktualizovaný rozšířený obsah a důraz je kladen na témata, která poskytnou nejlepší praktickou pomoc testerům.

Už existující Základní certifikát v testování softwaru (např. od ISEB, ASQF nebo lokálních výborů uznaných ISTQB), který byl udělen před vydáním tohoto mezinárodního certifikátu, bude považován za ekvivalent mezinárodního certifikátu. Základní certifikát neexpiruje a nebude potřebné, aby byl prodlužován nebo obnovován. Datum vydání se nachází na certifikátu.

V rámci každé zúčastněné země jsou místní aspekty kontrolovány lokálním výborem testování softwaru uznaným ISTQB. Povinnosti lokálních výborů jsou zadány ISTQB, ale jsou implementovány v rámci každé země. V povinnostech lokálních výborů se předpokládá zahrnutí akreditace poskytovatelů školení a nastavení zkoušek.

9. Příloha B – Studijní cíle / kognitivní úroveň znalostí

Pro tyto osnovy jsou použity následující studijní cíle. Každé téma v učebních osnovách bude prozkoušeno podle jemu odpovídajícímu studijnímu cíli.

Úroveň 1: Zapamatovat si (Z1)

Kandidát pochopí, zapamatuje si a vzpomene si na výraz, pojem nebo koncept.
Klíčová slova: zapamatovat si, získat znalost, vzpomenout si, rozeznat, umět.

Příklad
Rozezná definici selhání jako:

- "Nedodání služby koncovému uživateli nebo jiné zainteresované osobě", nebo
- "Odchylka komponenty nebo systému od jeho očekávané dodávky, služby nebo výsledku".

Úroveň 2: Pochopit (Z2)

Kandidát umí označit příčiny nebo vysvětlení u výroků týkajících se daného tématu a umí shrnout, porovnat, klasifikovat, rozdělit a uvést příklady pro koncepty testování.
Klíčová slova: shrnout, zevšeobecnit, abstrahovat, klasifikovat, porovnat, namapovat, odlišit, doložit příkladem, vysvětlit, přeložit, znázornit, odvodit, vyvodit, kategorizovat, vytvořit model.

Příklady
Umí vysvětlit důvod, proč by měly být testy vytvořeny, jakmile je to možné:

- Najít defekty, když jsou levněji odstranitelné.
- Najít nejdůležitější defekty jako první.

Umí vysvětlit podobnosti a rozdíly mezi integračním a systémovým testováním:

- Podobnosti: testování více než jedné komponenty, mohou být testovány nefunkcionální aspekty.
- Rozdíly: integrační testování se zaměřuje na rozhraní a interakce; a testování systému se zaměřuje na aspekty celého systému, jako je proces od začátku do konce (end-to-end).

Úroveň 3: Použít (Z3)

Kandidát umí vybrat správnou aplikaci konceptu nebo techniky a aplikovat ji v daném kontextu.
Klíčová slova: Zavést, vykonat, používat, sledovat postup, uplatnit postup.

Příklad

- Umí identifikovat hraniční hodnoty pro platné a neplatné sekce.
- Umí vybrat testovací případy pro daný diagram přechodu stavů s pokrytím všech přechodů.

Úroveň 4: Analyzovat (Z4)

Kandidát umí rozdělit informace vztahující se k proceduře nebo technice na základní části za účelem lepšího pochopení. Dokáže taktéž rozlišit fakta od logických závěrů. Typickým uplatněním je analýza dokumentů, softwarové nebo projektové situace a návrh vhodných kroků při řešení problému nebo úlohy.
Klíčová slova: Analyzovat, organizovat, najít souvislosti, integrovat, naznačit, rozložit, strukturovat, přisoudit, dekonstruovat, rozlišovat, odlišovat, rozeznat, zaměřit, vybrat.

Příklad

- Analyzujte produktová rizika a navrhněte preventivní a nápravné aktivity za účelem snížení rizik.
- Popište, které části záznamu o incidentu jsou skutečné a které jsou vyvozené z výsledků.

Reference

(Pro úrovně poznání studijních cílů)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon

10. Příloha C – Pravidla používaná pro Základní učební osnovy ISTQB

Zde uvedená pravidla byla použita při vývoji a revidování těchto osnov. (ZNAČKA je zobrazována po každém pravidlu jako zkrácený název pravidla.)

10.1.1 Všeobecná pravidla

SP1. Učební osnovy by měly být srozumitelné a absorbovatelné lidmi s žádnou až šestiměsíční (nebo delší) zkušeností v testování. (ŠESTIMĚSÍČNÍ).
SP2. Učební osnovy by měly být spíše praktické než teoretické. (PRAKTICKÝ)
SP3. Učební osnovy by měly být pro jim určené čtenáře jasné a jednoznačné. (JASNÝ)
SP4. Učební osnovy by měly být srozumitelné lidem z různých zemí a lehce přeložitelné do jiných jazyků. (PŘELOŽITELNÝ)
SP5. Učební osnovy v originálu by měly používat americkou angličtinu. (AMERICKÁ ANGLIČTINA)

10.1.2 Aktuální obsah

SO1. Učební osnovy by měly zahrnovat současné koncepty testování a měly by odrážet současné nejlepší praktiky v testování softwaru tam, kde je to všeobecně odsouhlaseno. Osnovy jsou předmětem revize každých tři až pět let. (AKTUÁLNÍ)
SO2. Učební osnovy by měly minimalizovat obsah záležitostí měnících se časem, jako jsou například aktuální tržní podmínky, aby mohly mít životnost tři až pět let. (ŽIVOTNOST)

10.1.3 Studijní cíle

SC1. Studijní cíle by měly rozlišovat mezi body na rozpoznání/zapamatování (úroveň znalostí Z1), body, kterým by kandidáti měli rozumět koncepčně (Z2), body, které by kandidáti měli být schopni používat v praxi (Z3) a body, které by měli být kandidáti schopni použít k analýze dokumentů, softwaru nebo situace na projektu v daném kontextu (Z4). (ZNALOSTNÍ ÚROVEŇ)
SC2. Popis obsahu by měl být konzistentní s cíli studia. (KONSISTENTNÍ S SC)
SC3. K objasnění studijních cílů by měly být vydány spolu s učební osnovou vzorové zkušební otázky pro každou hlavní část. (OTÁZKY K SC)

10.1.4 Celková struktura

SS1. Struktura osnov by měla být jasná a měla by dovolovat křížové odkazy na jiné části a též se na ně odkazovat zpětně z jiných částí, referencí ze zkušebních otázek a z dalších relevantních dokumentů. (KŘÍŽOVÉ ODKAZY)
SS2. Překrývání mezi částmi osnov by mělo být minimalizováno. (PŘEKRYTÍ)
SS3. Všechny části osnov by měly mít stejnou strukturu. (KONZISTENCE STRUKTURY)
SS4. Osnovy by měly obsahovat verzi, datum vydání a číslo stránky na každé straně. (VERZE)
SS5. Osnovy by měly zahrnovat směrnice pro množství času, které je potřebné věnovat každé části (za účelem vyjádřit úměrnou důležitost každého tématu). (STRÁVENÝ ČAS)

Reference

SR1. Zdroje a reference pro koncepty budou uvedeny v osnovách, aby tak pomohly poskytovatelům školení najít dostatek dalších informací o tématu. (REFERENCE)

SR2. V případě, kdy nejsou připraveny identifikované a jasné zdroje, mělo by být v osnovách uvedeno více detailů. Například definice jsou v glosáři, takže v osnovách jsou uvedeny pouze pojmy. (DETAILY BEZ REFERENCÍ)

Zdroje informací

Pojmy použité v osnovách jsou definovány ve Výkladovém slovníku ISTQB používaných při testování softwaru. Aktuální verze glosáře je dostupná v ISTQB (anglický jazyk) nebo CaSTB (český jazyk).

Seznam doporučené literatury o testování softwaru je také vydaný paralelně s těmito osnovami. Hlavní seznam literatury je uvedený v části Reference.

11. Příloha D – Upozornění poskytovatelům školení

Každému hlavnímu nadpisu v osnovách je přidělen vyznačený čas v minutách. Účelem je doporučit proporcionální rozdělení času pro každou část v rámci akreditovaného školení a zároveň určit přibližné minimum času pro výuku daných částí.
Poskytovatelé školení mohou dané části věnovat více času, než se uvádí v osnovách, a taktéž kandidáti mohou věnovat více času čtení a studiu. Studijní plán školení nemusí přesně kopírovat pořadí uvedené v osnovách.

Osnovy obsahují reference na platné standardy, které musí být použity při přípravě školícího materiálu. Každý použitý standard musí být ve verzi specifikované v aktuální verzi těchto osnov. Další publikace, šablony nebo standardy, na které nejsou reference v těchto učebních osnovách, mohou být také použity, ale nebudou předmětem zkoušení.

Všechny studijní cíle Z3 a Z4 vyžadují, aby byla do studijních materiálů zahrnuta praktická cvičení.

12. Příloha E – Poznámky k vydaným verzím Učebních osnov

Vydání z roku 2011

1. První vydání v českém jazyce

13. Index