Certifikovaný tester základnej úrovne
Učebná osnova

Verzia 2018 v3.1 – Beta 1

International Software Testing Qualifications Board

Czech and Slovak Testing Board

Upozornenie o ochrane autorských práv

História zmien (slovenská verzia)

Verzia	Dátum	Poznámka
ISTQB® CTFL SK 2018 v3.1.1	29. 10. 2022	Oprava pomenovania metodík iteratívneho vývoja na str. 31
ISTQB® CTFL SK 2018 v3.1	6. 4. 2020	Nová verzia podľa ISTQB® CTFL v3.1. Opravy preklepov a formátovania
ISTQB® CTFL SK 2018 v3.1 Beta 1	19. 2. 2020	Aktualizácia verzie 2018 podľa ISTQB® CTFL verzie 3.1. Prispôsobenie verzovania pôvodných ISTQB® materiálov.
ISTQB® CTFL SK 2018 v2.1 Beta 1	3. 2. 2020	Nový slovenský preklad verzie 2018 s grafickými úpravami
ISTQB® CTFL SK 2018 v1.2 Alfa 2	30. 1. 2020	Nový slovenský preklad verzie 2018 s jazykovými korekciami
ISTQB® CTFL SK 2018 v1.1 Alfa 1	12. 1. 2020	Nový slovenský preklad verzie 2018
ISTQB® CTFL SK 2018 v0.3 Beta 3	28. 2. 2019	Certifikovaný tester základnej úrovne – slovenský preklad – Beta 3
ISTQB® CTFL SK 2011 v0.2 Beta 2	9. 10. 2011	Certifikovaný tester základnej úrovne – slovenský preklad – Beta 2
ISTQB® CTFL SK 2011 v0.1 Beta 1	30. 9 . 2011	Certifikovaný tester základnej úrovne – slovenský preklad – Beta 1.
ISTQB® CTFL SK 2007 Beta 2	10. 3. 2008	Certifikovaný tester základnej úrovne – slovenský preklad – Beta 2.
ISTQB® CTFL SK 2007 Beta 1	1. 2. 2008	Certifikovaný tester základnej úrovne – slovenský preklad – Beta 1

História zmien (anglická verzia)

Verzia	Dátum	Poznámka
ISTQB® 2018 V3.1	11. 11. 2019	Certified Tester Foundation Level Syllabus Maintenance Release with minor updates – see Release Notes
ISTQB® 2018	27. 4. 2018	Candidate general release version
ISTQB® 2011	1. 4. 2011	Certified Tester Foundation Level Syllabus Maintenance Release – see Release Notes
ISTQB® 2010	30. 3. 2010	Certified Tester Foundation Level Syllabus Maintenance Release – see Release Notes
ISTQB® 2007	1. 5. 2007	Certified Tester Foundation Level Syllabus Maintenance Release
ISTQB® 2005	1. 7. 2005	Certified Tester Foundation Level Syllabus
ASQF V2.2	July 2003	ASQF Syllabus Foundation Level Version 2.2 “Lehrplan Grundlagen des Software-testens“
ISEB V2.0	25. 2. 1999	ISEB Software Testing Foundation Syllabus V2.0

Obsah

Poďakovanie ISTQB

Tento dokument bol formálne vydaný Valným zhromaždením ISTQB® (11. 11. 2019). Bol spísaný tímom ISTQB v zložení: Klaus Olsen (predseda), Meile Posthuma a Stephanie Ulrich.

Tím ďakuje lokálnym výborom za ich pripomienky týkajúce sa učebnej osnovy základnej úrovne 2018.

Učebná osnova základnej úrovne (The Foundation Syllabus) 2018 bola spísaná tímom ISTQB® v zložení: Klaus Olsen (predseda), Tauhida Parveen (podpredsedníčka), Rex Black (projektový manažér), Debra Friedenberg, Judy McKay, Meile Posthuma, Hans Schaefer, Radoslaw Smilgin , Mike Smith, Steve Toms, Stephanie Ulrichová, Marie Walsh a Eshraka Zakaria.

Tím ďakuje tímu v zložení Rex Black a Dorothy Graham za ich technickú úpravu, revíznym tímom a lokálnym výborom za ich návrhy a vstupy. Nasledujúce osoby sa zúčastnili revízie a hlasovania o týchto učebných osnovách: Tom Adams, Tobias Ahlgren, Xu Aiguo, Chris Van Bael, Katalin Balla, Graham Bath, Gualtiero Bazzana, Arne Becher, Veronica Belcher, Lars Hilmar Bjørstrup, Ralf Bongard, Armin Born, Robert Bornelind, Mette Bruhn-Pedersen, Geza Bujdoso, Earl Burba, Filipe Carlos, Young Jae Choi, Greg Collina, Alessandro Collino, Cui Zhe, Taz Daughtrey, Matthias Daigl, Wim Decoutere, Frans Dijkman, Klaudia Dussa-Zieger, Yonit Elbaz, Ofer Feldman, Mark Fewster, Florian Fieber, David Frei, Debra Friedenberg, Conrad Fujimoto, Pooja Gautam, Thorsten Geiselhart, Chen Geng, Christian Alexander Graf, Dorothy Graham, Michel Grandjean, Richard Green, Attila Gyuri, Jon Hagar, Kobi Halperin, Matthias Hamburg, Zsolt Hargitai, Satoshi Hasegawa, Berit Hatten, Wang Hongwei, Tamás Horváth, Leanne Howard, Chinthaka Indikadahena, J. Jayapradeep, Kari Kakkonen, Gábor Kapros, Beata Karpinska, Karl Kemminger, Kwanho Kim, Seonjoon Kim, Cecilia Kjellman, Johan Klintin, Corne Kruger, Gerard Kruijff, Peter Kunit, Hyeyong Kwon, Bruno Legeard, Thomas Letzkus, Alon Linetzki, Balder Lingegård, Tilo Linz, Hongbiao Liu, Claire Lohr, Ine Lutterman, Marek Majernik, Rik Marselis, Romanos Matthaios, Judy McKay, Fergus McLachlan, Dénes Medzihradszky, Stefan Merkel, Armin Metzger, Don Mills, Gary Mogyorodi, Ninna Morin, Ingvar Nordström, Adam Novak, Avi Ofer, Magnus C Ohlsson, Joel Oliviera, Monika Stocklein Olsen, Kenji Onishi, Francisca Cano Ortiz, Gitte Ottosen, Tuula Pääkkönen, Ana Paiva, Tal Pe'er, Helmut Pichler, Michaël Pilaeten, Horst Pohlmann, Andrew Pollner, Meile Posthuma, Vitalijs Puiso, Salvatore Reale, Stuart Reid, Ralf Reissing, Shark Ren, Miroslav Renda, Randy Rice, Adam Roman, Jan Sabak, Hans Schaefer, Ina Schieferdecker, Franz Schiller, Jianxiong Shen, Klaus Skafte, Mike Smith, Cristina Sobrero, Marco Sogliani, Murian Song, Emilio Soresi, Helder Sousa, Michael Sowers, Michael Stahl, Lucjan Stapp, Li Suyuan, Toby Thompson, Steve Toms, Sagi Traybel, Sabine Uhde, Stephanie Ulrich, Philippos Vakalakis, Erik van Veenendaal, Marianne Vesterdal, Ernst von Düring, Salinda Wickramasinghe, Marie Walsh, Søren Wassard, Hans Weiberg, Paul Weymouth, Hyungjin Yoon, John Young, Surong Yuan, Ester Zabar a Karolina Zmitrowicz.

Tím "International Software Testing Qualifications Board Working Group Foundation Level", verzia 2018: Klaus Olsen (predseda), Tauhida Parveen (podpredseda), Rex Black (projektový manažér), Dani Almog, Debra Friedenberg, Rashed Karim, Johan Klintin, Vipul Kocher, Corne Kruger, Sunny Kwon, Judy McKay, Thomas Müller, Igal Levi, Ebbe Munk, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh, Eshraka Zakaria, a Stevan Zivanovic. Kľúčový tím ďakuje tímu revidujúcim a všetkým lokálnym výborom za návrhy k súčasným učebným osnovám.

Tím "International Software Testing Qualifications Board Working Group Foundation Level", verzia 2011: Thomas Müller (predseda), Debra Friedenberg. Kľúčový tím ďakuje tímu revidujúcim (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquer, Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal) a všetkým lokálnym výborom za návrhy k súčasným učebným osnovám.

Tím "International Software Testing Qualifications Board Working Group Foundation Level", verzia 2010: Thomas Müller (predseda), Rahul Verma, Martin Klonky a Armin Beer. Kľúčový tím ďakuje tímu revidujúcich (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Tuula Pääkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veendendaal) a všetkým lokálnym výborom za ich pripomienky.

Tím "International Software Testing Qualifications Board Working Group Foundation Level", verzia 2007: Thomas Müller (predseda), Dorothy Graham, Debra Friedenberg a Erik van Veendendaal. Kľúčový tím ďakuje tímu revidujúcim (Hans Schaefer, Stephanie Ulrich, Meile posthuma, Anders Pettersson a Wonil Kwon) a všetkým lokálnym výborom za návrhy.

Tím "International Software Testing Qualifications Board Working Group Foundation Level", verzia 2005: Thomas Müller (predseda), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson a Erik van Veendendal. Kľúčový tím ďakuje tímu revidujúcim a všetkým lokálnym výborom za návrhy.

Poďakovanie CaSTB

Preklad do slovenského jazyka v rámci Czech and Slovak Testing Board (CaSTB) bol vytvorený v rámci tímu pracovnej prekladovej skupiny CaSTB s pomocou externých spolupracovníkov.

Preklad do slovenského jazyka, verzia 2018 v3.1: Daniel Poľan, Filip Rechtoris (preklad), Lenka Spodniaková (revízia).

Preklad do slovenského jazyka, nová verzia 2018: Daniel Poľan, Filip Rechtoris (preklad), Karol Frühauf, Miroslav Piter, Lenka Spodniaková (revízia).

Preklad do slovenského jazyka, verzia 2018: Marek Majerník (predseda), Róbert Dankanin, Karol Frühauf, Roman Jurkech, Tamara Kadnárová, Boris Maťko, Katarína Vavreková (preklad), Michal Fecko, Marko Cagalinec, Vanda Hudáková, Milan Domonji, Ivana Svátková, Patrik Maček, Simona Höffner (revízia).

Preklad do slovenského jazyka, verzia 2011: Róbert Dankanin, Marek Majerník, Ľuboš Práznovský (preklad), Daniela Čuvarská, Marcel Veselka (revízia).

Preklad do slovenského jazyka, verzia 2008: Daniela Čuvarská, Róbert Dankanin, Karol Frühauf, Marek Majerník, Ľuboš Práznovský (preklad), Roman Jurkech (revízia).

Hoci bolo snahou prekladateľov docieliť čo najvernejší preklad pôvodného anglického vydania, priestor pre zdokonaľovanie v tak komplexnom texte iste je a stále bude. Postrehy a podnety čitateľov preto radi uvítame na e-mailovej adrese translation@castb.org.

0 Úvodné informácie

0.1 Účel učebnej osnovy

Táto učebná osnova tvorí základ pre medzinárodnú kvalifikáciu testovania softvéru na základnej úrovni. Medzinárodný výbor pre kvalifikáciu testovania softvéru (International Software Testing Qualifications Board, ďalej len ISTQB®) poskytuje túto učebnú osnovu:

1. Členským výborom za účelom prekladu do ich lokálneho jazyka a akreditácie poskytovateľov školení. Členské výbory môžu prispôsobiť osnovu konkrétnym potrebám lokálneho jazyka a pridať odkazy na lokálne publikácie.
2. Certifikačným autoritám za účelom vytvorenia skúškových otázok v lokálnom jazyku uspôsobeným študijným cieľom tejto osnovy.
3. Poskytovateľom školení za účelom prípravy výukových materiálov a určenia vhodných výukových metód.
4. Kandidátom na certifikáciu za účelom pripraviť sa na certifikačnú skúšku a to buď v rámci školenia alebo nezávisle.
5. Medzinárodnej komunite softvérového a systémového inžinierstva za účelom podpory profesie testovania softvéru a systémov a ako základ pre odborné knihy a články.

ISTQB® môže umožniť používať túto učebnú osnovu ďalším subjektom na iné účely za predpokladu, že si tieto subjekty vyžiadajú a získajú písomný súhlas od ISTQB®.

0.2 Certifikovaný tester základnej úrovne pre testovanie softvéru

Základná úroveň je určená pre každého, kto sa zapája do testovania softvéru. Patria sem ľudia v rolách testerov, testovacích analytikov, špecialistov v oblasti testovania, konzultantov, manažérov testovania, akceptačných testerov a vývojárov softvéru. Základná úroveň je tiež vhodná pre každého, kto chce získať základné znalosti o testovaní softvéru, ako vlastníci produktov (product owners), projektoví manažéri, manažéri kvality, manažéri vývoja softvéru, biznis analytici, riaditelia a konzultanti pre oblasť IT. Držitelia certifikátu základnej úrovne sú oprávnení získavať ďalšie vyššie úrovne certifikácie v oblasti testovania softvéru z portfólia ISTQB®.

Prehľad základnej úrovne ISTQB® 2018 ako samostatný dokument, je stále platný pre túto učebnú osnovu základnej úrovne vo verzii 3.1 a obsahuje nasledujúce informácie:

- Profesijné ciele učebných osnov
- Maticu znázorňujúcu vzťah medzi profesijnými a študijnými cieľmi
- Zhrnutie tejto učebnej osnovy

0.3 Študijné ciele a kognitívne úrovne znalostí na preskúšanie

Študijné ciele sledujú profesijné ciele a používajú sa na vytváranie certifikačných skúšok pre základnú úroveň.

Všeobecne platí, že celý obsah tejto učebnej osnovy je možné preskúšať na úrovni K1 (viď nižšie), s výnimkou úvodu a príloh. To znamená, že kandidát môže byť vyzvaný, aby rozpoznal, zapamätal si, alebo si vybavil kľúčové slovo alebo koncept uvedený v niektorej zo šiestich kapitol. Úrovne znalostí konkrétnych študijných cieľov sú uvedené na začiatku každej kapitoly a klasifikujú sa takto:

- K1: Zapamätať si

- K2: Pochopiť
- K3: Aplikovať

Ďalšie podrobnosti a príklady študijných cieľov sú uvedené v Prílohe B.

Všetky pojmy uvedené na začiatku kapitol, pod hlavičkou kapitoly ako kľúčové slová, si má kandidát zapamätať (K1), aj keď nie sú výslovne uvedené v študijných cieľoch.

0.4 Certifikačná skúška základnej úrovne

Certifikačná skúška základnej úrovne je založená na tejto učebnej osnove. Odpovede na jednotlivé skúškové otázky môžu vyžadovať znalosti z jednej alebo viacerých kapitol tejto učebnej osnovy. Všetky časti učebnej osnovy môžu byť súčasťou certifikačnej skúšky, s výnimkou úvodu a príloh. Normy, knihy a ďalšie ISTQB® učebné osnovy sú zahrnuté ako referencie. Ich obsah nie je súčasťou skúšky s výnimkou toho, čo je z ich obsahu priamo zahrnuté v tejto učebnej osnove.

Formát skúšky je test s otázkami s viacerými možnosťami odpovedí (multiple choice). Skúška obsahuje 40 otázok. Na úspešné absolvovanie certifikačnej skúšky je potrebné správne odpovedať aspoň na 65% otázok (t. j. 26 otázok).

Skúšku je možné absolvovať v rámci akreditovaného školenia alebo nezávisle u poskytovateľa certifikačných skúšok. Absolvovanie akreditovaného školenia nie je podmienkou pre účasť na skúške.

0.5 Akreditácia

Členský výbor ISTQB® môže akreditovať poskytovateľov školení, ktorých študijné materiály a kurz je v súlade s touto učebnou osnovou. Poskytovatelia školení môžu získať podmienky akreditácie od daného členského výboru ISTQB®, alebo orgánu ktorý vykonáva túto akreditáciu. Akreditovaný kurz uznaný ako vyhovujúci tejto učebnej osnove, môže zahrnúť aj vykonanie certifikačnej skúšky

0.6 Úroveň detailu

Úroveň detailu tejto učebnej osnovy umožňuje vykonávať medzinárodne štandardizované kurzy a skúšky. V záujme dosiahnutia tohto cieľa sa učebné osnovy skladajú z/zo:

- všeobecných inštruktážnych cieľov opisujúcich zámery osnov základnej úrovne
- zoznamu pojmov, ktoré si študenti musia zapamätať
- študijných cieľov pre každú oblasť znalostí popisujúcich výsledky kognitívneho učenia, ktoré sa majú dosiahnuť
- opisu kľúčových prístupov vrátane odkazov na zdroje, ako je napríklad použitá literatúra a normy

Obsah učebnej osnovy nie je opisom všetkých znalostí z oblasti testovania softvéru. Odráža úroveň detailu, ktorá je obsiahnutá vo vzdelávacích kurzoch pre základnú úroveň. Zameriava sa na koncepciu testovania a testovacie techniky, ktoré sa dajú použiť vo všetkých projektoch softvérového vývoja vrátane agilných. Táto učebná osnova neobsahuje žiadne študijné ciele týkajúce sa konkrétneho životného cyklu alebo metódy vývoja softvéru, ale prinášajú diskusiu o tom, ako sa tieto koncepty uplatňujú v agilných projektoch, alebo iných typoch iteratívnych, inkrementálnych a sekvenčných životných cyklov vývoja softvéru.

0.7 Usporiadanie učebnej osnovy

Súčasťou učebnej osnovy je šesť kapitol s obsahom na preskúšanie. Hlavný nadpis kapitoly tiež obsahuje celkovú dobu výučby pre danú kapitolu v rámci akreditovaných kurzov. Táto hodnota je uvedená len na úrovni celej kapitoly. V prípade akreditovaných vzdelávacích kurzov táto učebná osnova vyžaduje minimálne 16,75 hodín výučby rozdelenej do šiestich kapitol a to nasledovne:

- Kapitola 1: 175 minút Základy testovania
- Kapitola 2: 100 minút Testovanie v rámci životného cyklu vývoja softvéru
- Kapitola 3: 135 minút Statické testovanie
- Kapitola 4: 330 minút Techniky testovania
- Kapitola 5: 225 minút Manažment testovania
- Kapitola 6: 40 minút Nástroje na podporu testovania

1 Základy testovania – 175 minút

Kľúčové slová

pokrytie, ladenie, defekt, chyba, zlyhanie, kvalita, zabezpečenie kvality, koreňová príčina, testovacia analýza, testovacia báza, testovací prípad, dokončenie testovania, testovacia podmienka, riadenie testovania, testovacie dáta, návrh testov, vykonanie testu, , implementácia testov, monitorovanie testovania, testovaný objekt, cieľ testovania, testovacie orákulum (test oracle), plánovanie testovania, testovacia procedúra, proces testovania, testovacia sada, testovanie, testvér, trasovateľnosť, validácia, verifikácia

Študijné ciele pre kapitolu 1 – Základy testovania:

1.1 Čo je testovanie?

FL-1.1.1 (K1) Identifikovať typické ciele testovania

FL-1.1.2 (K2) Rozlišovať medzi testovaním a ladením

1.2 Prečo je testovanie potrebné?

FL-1.2.1 (K2) Pomocou príkladov uviesť, prečo je testovanie potrebné

FL-1.2.2 (K2) Opísať vzťah medzi testovaním a zabezpečovaním kvality a uviesť príklady, ako testovanie prispieva k vyššej kvalite

FL-1.2.3 (K2) Rozlišovať medzi chybou, defektom a zlyhaním

FL-1.2.4 (K2) Rozlišovať medzi koreňovou príčinou defektu a jeho následkami

1.3 Sedem princípov testovania

FL-1.3.1 (K2) Vysvetliť sedem princípov testovania

1.4 Proces testovania

FL-1.4.1 (K2) Vysvetliť aký vplyv má kontext na proces testovania

FL-1.4.2 (K2) Opísať testovacie činnosti a príslušné úlohy v rámci procesu testovania

FL-1.4.3 (K2) Rozlíšiť pracovné produkty, ktoré podporujú proces testovania

FL-1.4.4 (K2) Vysvetliť hodnotu zachovania trasovateľnosti medzi testovacou bázou a pracovnými produktmi testovania

1.5 Psychológia testovania

FL-1.5.1 (K1) Identifikovať psychologické faktory, ktoré ovplyvňujú úspech testovania

FL-1.5.2 (K2) Vysvetliť rozdiel medzi zmýšľaním potrebným pre testovacie činnosti a zmýšľaním potrebným pre vývojové činnosti

1.1 Čo je testovanie?

Softvérové systémy sú neoddeliteľnou súčasťou života. Od biznisových aplikácií (napr. bankovníctva) až po spotrebiteľské produkty (napr. automobily). Väčšina ľudí má skúsenosť so softvérom, ktorý nefungoval podľa očakávaní. Softvér, ktorý nepracuje správne môže viesť k mnohým problémom, vrátane straty peňazí, času, alebo povesti v biznise, a dokonca aj k zraneniam alebo smrti. Testovanie softvéru je spôsob, ako posúdiť kvalitu softvéru a znížiť riziko zlyhania softvéru pri jeho použití.

Pomerne bežné, ale nesprávne chápanie testovania je, že sa skladá len z vykonávania testov, t.z. spustenie softvéru a kontrolovanie výsledkov. Ako je vysvetlené v kapitole 1.4, testovanie softvéru je proces, ktorý zahŕňa mnoho rôznych činností a vykonávanie testov (vrátane kontroly výsledkov) je len jednou z nich. Proces testovania zahŕňa aj činnosti, ako je plánovanie testovania, analýzu, návrh a implementáciu testov, podávanie správy o pokroku a výsledkoch testovania, a vyhodnotenie kvality testovaného objektu.

Niektoré formy testovania zahŕňajú vykonávanie komponentov alebo systémov, ktoré sú predmetom testovania. Takéto testovanie sa nazýva dynamické testovanie. Iné formy testovania vykonávanie komponentov alebo systémov nezahŕňajú. Takéto testovanie sa nazýva statické testovanie. Teda, testovanie zahŕňa aj revíziu (preskúmanie) pracovných produktov, ako sú požiadavky, užívateľské scenáre, a zdrojový kód.

Ďalšie bežné, ale nesprávne chápanie testovania spočíva v tom, že sa zameriava výhradne na verifikáciu požiadaviek, užívateľských scenárov, alebo iných špecifikácií. Testovanie, okrem kontroly či systém spĺňa špecifikované požiadavky, zahŕňa aj validáciu, ktorá kontroluje, či systém plní potreby používateľov a iných zainteresovaných strán vo svojom prevádzkovom prostredí.

Testovacie činnosti sú organizované a vykonávané odlišne v rôznych životných cykloch vývoja softvéru (viď kapitola 2.1).

1.1.1 Typické ciele testovania

Medzi ciele testovania pre každý jeden projekt patrí:

- predchádzať vziku defektov hodnotením pracovných produktov, ako sú požiadavky, užívateľské scenáre, návrh a kód
- verifikovať či boli splnené všetky špecifikované požiadavky
- kontrolovať či je testovaný objekt hotový a validovať, či pracuje ako používatelia a iné zainteresované strany očakávajú
- budovať dôveru v úroveň kvality testovaného objektu
- odhaliť defekty a zlyhania a tým znížiť úroveň rizika neprimeranej kvality softvéru
- poskytnúť dostatočné informácie zainteresovaným stranám, aby mohli prijímať kvalifikované rozhodnutia, najmä pokiaľ ide o úroveň kvality testovaného objektu
- dodržať zmluvné, právne alebo regulatórne požiadavky alebo normy, a/alebo verifikovať zhodu testovaného objektu s takýmito požiadavkami alebo normami

Ciele testovania sa môžu líšiť v závislosti od kontextu testovaného komponentu alebo systému, úrovne testovania a modelu životného cyklu vývoja softvéru. Rozdiely môžu byť napríklad:

- Jedným z cieľov testovania komponentov môže byť nájdenie čo najväčšieho počtu zlyhaní, aby sa ich príčiny odhalili a odstránili čo najskôr. Ďalším cieľom môže byť zvýšenie pokrytia kódu komponentovými testami.

- Jedným z cieľov akceptačného testovania môže byť potvrdenie, že systém funguje podľa očakávaní a spĺňa požiadavky. Ďalším z cieľov tohto testovania môže byť poskytnutie informácií zainteresovaným stranám o úrovni rizika vydania systému v danom čase.

1.1.2 Testovanie a ladenie

Testovanie a ladenie sa líšia. Vykonávanie testov môže odhaliť zlyhania, ktoré sú spôsobené defektmi v softvéri. Ladenie je vývojová činnosť, ktorá nachádza, analyzuje a opravuje takéto defekty. Následné konfirmačné testovanie skontroluje, či v dôsledku týchto opráv došlo k vyriešeniu defektov. V niektorých prípadoch sú testeri zodpovední za prvotné a záverečné potvrdzujúce testovanie, zatiaľ čo vývojári robia ladenie a príslušné testovanie komponentov a integračné testovanie komponentov (v rámci procesu priebežnej integrácie, v agilnom vývoji a v niektorých ďalších životných cykloch vývoju softvéru môžu byť testeri zapojení aj do ladenia a testovania komponentov).

Pre ďalšie informácie o konceptoch testovanie softvéru viď norma *ISO (ISO/IEC/IEEE 29119-1)*.

1.2 Prečo je testovanie potrebné?

Dôsledné testovanie komponentov a systémov, a k nim príslušná dokumentácia môžu pomôcť znížiť riziko zlyhaní v prevádzke. Nachádzanie defektov a ich následná oprava, prispieva k zvyšovaniu kvality týchto komponentov alebo systémov. Okrem toho môže byť testovanie softvéru vyžadované z dôvodu splnenia zmluvných alebo zákonných požiadaviek špecifických pre konkrétne priemyselné odvetvie.

1.2.1 Prínosy testovania k úspechu

V priebehu histórie výpočtovej techniky, bolo pomerne bežné, že softvér a systémy, ktoré boli nasadené do prevádzky, obsahovali defekty, ktoré následne spôsobili zlyhania alebo boli v rozpore s potrebami zúčastnených strán. Používanie vhodných techník testovania však môže znížiť počet takýchto problematických dodávok. To za predpokladu že sa tieto techniky aplikujú s primeranou odbornou znalosťou testovania, t. j. sú použité na príslušných úrovniach testovania a v príslušných fázach životného cyklu vývoja softvéru. Medzi príklady patrí:

- Zapojenie testerov do revízie požiadaviek alebo zlepšovania užívateľských scenárov môže v týchto pracovných produktoch odhaliť defekty. Identifikácia a odstránenie defektov v požiadavkách znižuje riziko vývoja nesprávnej alebo netestovateľnej vlastnosti systému.
- Úzka spolupráca testerov so systémovými návrhármi, počas návrhu systému, umožní obom stranám lepšie porozumieť tomuto návrhu, a ako ho testovať. Toto lepšie porozumenie môže znížiť riziko zásadných defektov v návrhu a podporiť skorú identifikáciu potrebných testov.
- Úzka spolupráca testerov s vývojármi, počas vývoja kódu umožní obom stranám lepšie porozumieť kódu a ako ho testovať. Toto lepšie porozumenie môže znížiť riziko defektov v kóde a v testoch.
- Zapojenie testerov do verifikácie a validácie softvéru pred jeho vydaním môže odhaliť zlyhania, ktoré by inak neboli nájdené, a následne podporiť proces odstránenia defektov, ktoré spôsobili tieto zlyhania (napr. ladenie). Tým sa zvyšuje pravdepodobnosť, že softvér vyhovuje potrebám zúčastnených strán a spĺňa ostatné požiadavky.

Okrem týchto príkladov, dosiahnutie definovaných testovacích cieľov (viď kapitola 1.1.1) prispieva k celkovému úspechu vývoja a údržby softvéru.

1.2.2 Zabezpečovanie kvality a testovanie

Zatiaľ čo ľudia často používajú frázu zabezpečenie kvality (quality assurance, alebo QA) ako odkaz na testovanie, zabezpečenie kvality a testovanie nie je to isté. Sú však prepojené. Oba pojmy sú súčasťou obsiahlejšej disciplíny nazývanej manažment kvality (quality management). Manažment kvality zahŕňa všetky činnosti, ktoré sú zodpovedné za smerovanie a riadenie organizácie, pokiaľ ide o kvalitu. Okrem iných činností, manažment kvality zahŕňa oboje, zabezpečenie kvality a kontrolu kvality (quality control). Zabezpečovanie kvality sa zvyčajne zameriava na dodržiavanie správnych procesov s cieľom zvýšiť dôveru v to, že sa dosiahne primeraná úroveň kvality. Pokiaľ sa procesy vykonávajú správne, pracovné produkty vytvorené týmito procesmi sú vo všeobecnosti vyššej kvality, čo prispieva k prevencii defektov. Použitie analýzy koreňových príčin na zistenie a odstránenie príčin defektov spolu s riadnym uplatňovaním zistení retrospektív na zlepšenie procesov sú tiež dôležité pre účinné zabezpečovanie kvality.

Kontrola kvality zahŕňa rôzne činnosti vrátane testovacích činností, ktoré podporujú dosiahnutie primeranej úrovne kvality. Testovacie činnosti sú súčasťou celkového procesu vývoja alebo údržby softvéru. Keďže zabezpečenie kvality sa zaoberá riadnym vykonaním celého procesu, zabezpečenie kvality podporuje riadne testovanie. Ako je opísané v kapitolách 1.1.1 a 1.2.1, testovanie prispieva k dosiahnutiu kvality niekoľkými spôsobmi.

1.2.3 Chyby, defekty a zlyhania

Človek môže urobiť chybu (omyl), ktorá vedie k vzniku defektu (vady alebo bugu) v kóde softvéru alebo v niektorých iných súvisiacich pracovných produktoch. Chyba, ktorá vedie ku vzniku defektu v jednom pracovnom produkte môže vyvolať ďalšiu chybu, ktorá vedie ku vzniku defektu v súvisiacich pracovných produktoch. Napríklad, chyba v definícii požiadaviek môže viesť k defektu v požiadavke, ktorý vedie ku chybe pri programovaní a tá následne ku vzniku defektu v kóde.

Ak je defekt v kóde vykonaný, môže spôsobiť zlyhanie, aj keď nie nevyhnutne. Zlyhanie totiž môže závisieť na okolnostiach. Napríklad, niektoré defekty vyžadujú veľmi špecifické vstupy alebo vstupné podmienky na vyvolanie zlyhania, ktoré sa z tohto dôvodu môže vyskytnúť zriedkavo alebo aj nikdy.

Chyby sa môžu vyskytnúť z mnohých dôvodov, napríklad:

- časová tieseň
- ľudská omylnosť
- neskúsení alebo nedostatočne kvalifikovaní účastníci na projekte
- nedorozumenia a nesprávna komunikácia medzi účastníkmi projektu vrátane nesprávnej komunikácie o požiadavkách a návrhu
- zložitosť kódu, návrhu, architektúry, nevyriešené zásadné problémy, a/alebo používané technológie
- nedorozumenia ohľadom vnútorných alebo vonkajších rozhraní systému, najmä v prípade veľkého počtu interakcií na týchto rozhraniach
- nové, neznáme technológie

Okrem zlyhaní spôsobených defektmi v kóde môžu byť zlyhania spôsobené aj podmienkami konkrétneho prostredia. Napríklad žiarenie, elektromagnetické pole a znečistenie môžu spôsobovať chyby vo firmvéri alebo ovplyvňovať výkon softvéru zmenou hardvérových podmienok.

Nie všetky neočakávané výsledky testov sú zlyhania. Falošne-pozitívne (false-positive) výsledky testov sa môžu vyskytnúť v dôsledku chýb pri vykonaní testov, alebo v dôsledku defektov v testovacích dátach, testovacom prostredí, alebo v inom testvéri, či z rôznych iných dôvodov. Opačná situácia môže tiež nastať, keď podobné chyby alebo defekty vedú k falošne-negatívnym (false-negative) výsledkom.

Falošne-negatívne výsledky, vznikajú ak testy neodhalia defekty, ktoré by odhaliť mali; falošne-pozitívne výsledky sú hlásené ako defekty, aj keď nimi v skutočnosti nie sú.

1.2.4 Defekty, ich koreňové príčiny a dôsledky

Koreňové príčiny defektov sú prvotné akcie alebo podmienky, ktoré prispeli k vytvoreniu defektu. Nájdené defekty sa môžu analyzovať s cieľom zistenia ich koreňovej príčiny, a tak znížiť výskyt podobných defektov v budúcnosti. Zameraním sa na najvýznamnejšie koreňové príčiny, môže analýza koreňových príčin viesť k zlepšovaniu procesov a tým zabrániť vzniku významného počtu defektov v budúcnosti.

Predpokladajme napríklad, že nesprávne platby úrokov (z dôvodu jedného chybného riadku kódu) majú za následok sťažnosti zákazníkov. Chybný kód bol napísaný podľa užívateľského scenáru, ktorý bol nejednoznačný, z dôvodu nepochopenia princípu výpočtu úrokov vlastníkom produktu. Ak sa vo výpočtoch úrokov vyskytuje množstvo defektov a tieto defekty majú svoju koreňovú príčinu v podobných nedorozumeniach, vlastníci produktu by mali podstúpiť školenie na tému výpočtu úrokov, aby sa znížil výskyt takýchto defektov v budúcnosti.

V tomto príklade, sú sťažnosti zákazníkov dôsledkami. Nesprávny výpočet úrokových splátok je zlyhanie. Nesprávny výpočet v kóde je defekt, ktorý vznikol z pôvodného defektu vo forme nejednoznačnosti v užívateľskom scenári. Koreňová príčina pôvodného defektu bola v nedostatku znalostí na strane vlastníka produktu,čo viedlo k tomu, že vlastník produktu urobil chybu pri definícii užívateľského scenára. Proces analýzy koreňovej príčiny je vysvetlený v učebných osnovách *ISTQB-CTEL-TM* a *ISTQB-CTEL-ITP.*

1.3 Sedem princípov testovania

V posledných 50 rokoch bola navrhnutá rada princípov, ktoré poskytujú obecné návody spoločné pre všetky typy testovania.

1. Testovanie ukazuje prítomnosť defektov, nie ich neprítomnosť

Testovanie môže ukázať, že defekty sú prítomné, ale nemôže dokázať, že žiadne defekty neexistujú. Testovanie znižuje pravdepodobnosť, že v softvéri zostanú neodhalené defekty, ale keď sa neobjavia žiadne defekty, nie je to dôkazom jeho správnosti.

2. Kompletné testovanie je nemožné

Testovanie všetkého (všetky kombinácie vstupov a vstupných podmienok) nie je možné s výnimkou triviálnych prípadov. Namiesto snahy o kompletné testovanie je lepšie sa zamerať na analýzu rizík, vhodné techniky testovania a prioritizáciu.

3. Včasné testovanie šetrí čas a peniaze

Pre včasné odhalenie defektov by sa mali statické aj dynamické testovacie činnosti začať v životnom cykle vývoja softvéru čo najskôr. Včasné testovanie je niekedy označované ako posun doľava. Testovanie na začiatku životného cyklu vývoja softvéru pomáha znižovať alebo eliminovať nákladné zmeny v budúcnosti (viď kapitola 3.1).

4. Zhlukovanie defektov

Väčšina defektov objavených počas testovania pred vydaním je zvyčajne obsiahnutá v malom počte modulov, alebo je zodpovedná za väčšinu prevádzkových zlyhaní. Predpokladané a skutočne zistené zhluky defektov počas testovania alebo v prevádzke sú dôležitým vstupom do analýzy rizík, ktorá sa používa k vhodnému zameraniu testovacích činností (ako je uvedené v Princípe 2).

5. Vyvarovanie sa pesticídovému paradoxu

Ak sa rovnaké testy opakujú neustále dookola, nakoniec už neodhalia žiadne nové defekty. Na odhalenie nových defektov, môže byť nutné zmeniť existujúce testy a testovacie dáta, prípadne vytvoriť nové testy. (Testy prestanú byť účinné pri hľadaní defektov, rovnako ako pesticídy prestanú byť účinné pri zabíjaní hmyzu.) V niektorých prípadoch, ako je napríklad automatizované regresné testovanie, má pesticídový paradox pozitívny prínos, ktorým je relatívne malý počet regresných defektov.

6. Testovanie je závislé na kontexte

Testovanie sa vykonáva odlišne v rôznych kontextoch. Napríklad priemyselový bezpečnostne kritický softvér sa testuje odlišne ako mobilná aplikácie pre e-commerce. Ako ďalší príklad je rozdiel v testovaní na agilnom projekte a na projekte so sekvenčným životným cyklom vývoja softvéru (viď kapitola 2.1).

7. Neprítomnosť chýb je klam

Niektoré organizácie očakávajú, že testeri dokážu vykonať všetky možné typy testov a nájsť všetky možné defekty, ale Princípy 1 a 2 nám hovoria, že to možné nie je. Ďalším klamom je očakávanie, že len nájdenie a opravenie veľkého počtu defektov zaistí úspech systému. Napríklad: napriek dôkladnému testovaniu všetkých špecifikovaných požiadaviek a odstráneniu všetkých nájdených defektov nezabránime vzniknutiu systému, ktorý je ťažko použiteľný a nespĺňa potreby a očakávania užívateľov, alebo tomu, že je menejcenný v porovnaní s konkurenčnými systémami.

Pre príklady týchto a ďalších testovacích princípov viď *Myers 2011*, *Kaner 2002*, *Weinberg 2008 a Beizer 1990*.

1.4 Proces testovania

Neexistuje žiadny univerzálny proces testovania softvéru, ale existujú bežné sady testovacích činností, bez ktorých testovanie menej pravdepodobne dosiahne stanovené ciele. Tieto sady testovacích činností tvoria proces testovania. Vhodný a konkrétny proces testovania softvéru v danej situácii závisí od mnohých faktorov. Aké testovacie činnosti budú súčasťou procesu testovania, ako sa tieto činnosti vykonávajú a kedy sa tieto činnosti vyskytujú, môže byť predmetom diskusie v rámci testovacej stratégii organizácie.

1.4.1 Proces testovania v kontexte

Kontextové faktory, ktoré ovplyvňujú testovací proces danej organizácie, zahŕňajú, ale nie sú obmedzené len na:

- model životného cyklu vývoja softvéru a metódy použité v projekte
- zvažované úrovne a typy testovania
- projektové a produktové riziká
- sféru biznisu
- prevádzkové obmedzenia zahŕňajú, ale nie sú obmedzené len na:
  - rozpočty a zdroje
  - harmonogramy
  - zložitosti
  - zmluvné a regulatórne požiadavky
- politiky a postupy organizácie
- požadované interné a externé normy

Nasledujúce kapitoly popisujú všeobecné aspekty organizačných procesov testovania z hľadiska nasledujúcich bodov:

- testovacie činnosti a úlohy
- pracovné produkty testovania
- trasovateľnosť medzi testovacou bázou a pracovnými produktmi testovania

Je veľmi užitočné, ak testovacia báza (pre akúkoľvek zvažovanú úroveň alebo typ testovania) má definované merateľné kritériá pokrytia. Kritériá pokrytia môžu byť použité ako kľúčové ukazovatele výkonnosti (KPI – Key Performance Indicators) na riadenie činností, ktoré dokazujú dosiahnutie cieľov softvérového testovania (viď kapitola 1.1.1).

Napríklad pre mobilnú aplikáciu môže testovacia báza obsahovať zoznam požiadaviek a zoznam podporovaných mobilných zariadení. Každá požiadavka ako aj každé podporované zariadenie sú prvkami testovacej bázy. Kritériá pokrytia môžu vyžadovať aspoň jeden testovací prípad pre každý prvok testovacej bázy. Po vykonaní testov, sú zainteresované strany informované o splnení špecifikovaných požiadaviek a o tom či boli zistené nejaké zlyhania na podporovaných zariadeniach.

Pre ďalšie informácie ohľadom testovacích procesov viď norma *ISO (ISO/IEC/IEEE 29119-2)*.

1.4.2 Testovacie činnosti a úlohy

Proces testovania pozostáva z hlavných skupín činností ako:

- plánovanie testovania
- monitorovanie a riadenie testovania
- testovacia analýza
- návrh testov
- implementácia testov
- vykonanie testov
- dokončenie testovania

Každá hlavná skupina činností sa skladá zo základných činností, ktoré budú popísané nižšie v podkapitolách. Každá dielčia činnosť pozostáva z viacerých individuálnych úloh, ktoré sa môžu líšiť v závislosti na konkrétnom projekte alebo vydaní.

Ďalej platí, že hoci mnohé z týchto hlavných skupín činností sa logicky zdajú byť sekvenčnými, často sa vykonávajú iteratívne. Napríklad agilný vývoj pozostáva z malých iterácií návrhu softvéru, zostavenia softvéru a testovania, ktoré prebiehajú nepretržite a za podpory priebežného plánovania. Preto aj testovacie činnosti sú v rámci tohto prístupu k vývoju softvéru vykonávané iteratívne a priebežne. Dokonca aj v sekvenčnom vývoji softvéru bude logická postupnosť hlavných skupín činností obsahovať prekrytia, kombinácie, súbežnosti, prípadne vynechanie niektorých položiek. Preto je nutné prispôsobiť hlavné skupiny činností kontextu systému a projektu.

Plánovanie testovania

Plánovanie testovania zahŕňa činnosti, ktoré definujú ciele testovania a prístup k splneniu testovacích cieľov v rámci obmedzení daného kontextu (napr. špecifikovaním vhodných testovacích techník a úloh a formulovaním testovacieho harmonogramu s ohľadom na dodržanie termínu). Plány testovania môžu byť opätovne revidované na základe spätnej väzby z monitorovacích a kontrolných činností. Plánovanie testovania je ďalej vysvetlené v kapitole 5.2.

Monitorovanie a riadenie testovania

Monitorovanie testovania zahŕňa priebežné porovnávanie aktuálneho stavu oproti plánovanému pokroku pomocou metrík monitorovania testov definovaných v pláne testovania. Riadenie testovania zahŕňa zavedenie opatrení potrebných na splnenie cieľov plánu testovania (ktoré sa môžu časom aktualizovať). Monitorovanie a riadenie testovania sú podporené vyhodnotením výstupných kritérií, ktoré sa v niektorých životných cykloch vývoja softvéru označujú ako definícia hotového (DoD - Definition of Done) (viď učebné osnovy ISTQB-CTFL-AT). Vyhodnotenie výstupných kritérií pre vykonávanie testov v rámci danej úrovne testovania môže zahŕňať:

- kontrolu výsledkov testov a záznamov testovania oproti stanoveným kritériám pokrytia
- posúdenie úrovne kvality komponentu alebo systému na základe výsledkov testov a záznamov testovania
- určenie, či sú potrebné ďalšie testy (napr. ak testy pôvodne určené na dosiahnutie určitej úrovne pokrytia produktových rizík neboli dostačujúce, čo by vyžadovalo vytvorenie a vykonanie dodatočných testov)

Pokrok práce pri testovaní oproti plánu sa oznamuje zainteresovaným stranám v správach o postupe testovania vrátane odchýlok od plánu a informácií potrebných na akékoľvek rozhodnutia o zastavení testovania.

Monitorovanie a riadenie testovania sú ďalej vysvetlené v kapitole 5.3.

Testovacia analýza

Počas testovacej analýzy sa analyzuje testovacia báza za účelom identifikácie testovateľných vlastností a taktiež sa určia odpovedajúce testovacie podmienky. Inými slovami, testovacia analýza určuje "čo testovať", pokiaľ ide o merateľné kritériá pokrytia.

Testovacia analýza pozostáva z nasledujúcich hlavných činností:

- Analýza testovacej bázy vhodná pre zamýšľanú úroveň testovania, ako napríklad:
  - Špecifikácia požiadaviek, ako sú biznis, funkcionálne alebo systémové požiadavky, užívateľské scenáre, epic-y, prípady použitia alebo ďalších podobných pracovných produktov, ktoré určujú požadované funkcionálne a nefunkcionálne správanie komponentu alebo systému.
  - Informácie o návrhu a implementácii, ako sú dokumenty alebo diagramy architektúry systému alebo softvéru, špecifikácie návrhu, grafy tokov volaní, diagramy modelovania (napr. diagramy UML alebo diagramy vzťahov a entít), špecifikácie rozhrania alebo podobné pracovné produkty, ktoré určujú štruktúru komponentu alebo systému.
  - Implementácia komponentu alebo samotného systému, vrátane kódu, databázových dotazov a metadát, a rozhraní
  - Správy o analýze rizík, ktoré môžu brať do úvahy funkcionálne, nefunkcionálne a štrukturálne aspekty komponentu alebo systému
- Hodnotenie testovacej bázy a testovacích položiek s cieľom identifikovať defekty rôznych typov, ako sú:
  - nejednoznačnosti
  - opomenutia
  - nezrovnalosti
  - nepresnosti
  - rozpory
  - nadbytočnosti

- Identifikácia vlastností a súborov vlastností, ktoré sa majú testovať
- Definícia testovacích podmienok a stanovenie ich priorít pre každú vlastnosť založenú na analýze testovacej bázy vzhľadom na funkcionálne, nefunkcionálne a štrukturálne charakteristiky, iné biznis a technické faktory, a úrovne rizík
- Zachytenie obojsmernej trasovateľnosti medzi jednotlivými prvkami testovacej bázy a súvisiacimi testovacími podmienkami (viď kapitola 1.4.3 a 1.4.4)

V procese testovacej analýzy môže byť užitočná aplikácia techník testovania čiernej skrinky, bielej skrinky a testovacích techník založených na skúsenostiach (viď kapitola 4), s cieľom znížiť pravdepodobnosť vynechania dôležitých testovacích podmienok a definovať presnejšie testovacie podmienky.

V niektorých prípadoch testovacia analýza prinesie testovacie podmienky, ktoré môžu byť použité ako ciele testovania v testovacích listinách. Testovacie listiny sú typické pracovné produkty používané pri testovaní založenom na skúsenostiach (viď kapitola 4.4.2). Ak sú tieto ciele testovania sledovateľné k testovacej báze, je možné merať dosiahnuté pokrytie počas testovania založenom na skúsenostiach.

Identifikácia defektov počas testovacej analýzy je dôležitým potenciálnym prínosom, najmä ak sa nepoužíva žiadny iný proces revízie a/alebo proces testovania je úzko spojený s procesom revízie. Tieto činnosti v rámci testovacej analýzy nielen verifikujú, či sú požiadavky konzistentné, správne vyjadrené a úplné, ale tiež validujú, či požiadavky správne zachytávajú potreby zákazníkov, užívateľov a ďalších zúčastnených strán. Existujú techniky ako vývoj riadený správaním (BDD – Behavior Driven Development) a vývoj riadený akceptačnými testami (ATDD – Acceptance Tests Driven Development), ktoré zahrňujú vytváranie testovacích podmienok a testovacích prípadov z užívateľských scenárov a akceptačných kritérií ešte pred začiatkom písania kódu. Tieto techniky okrem toho verifikujú, validujú a nachádzajú defekty v týchto užívateľských scenároch a akceptačných kritériách (viď učebná osnova *ISTQB-CTFL-AT*).

Návrh testov

Počas návrhu testov sa testovacie podmienky spracujú na jednotlivé všeobecné testovacie prípady, ich sady a iný testvér. Kým testovacia analýza odpovedá na otázku "čo testovať?", návrh testov dáva odpoveď na otázku "ako otestovať?"

Návrh testov pozostáva z nasledujúcich hlavných činností:

- navrhovanie a prioritizácia testovacích prípadov a ich sád
- identifikácia potrebných testovacích dát pre testovacie podmienky a testovacie prípady
- návrh testovacieho prostredia, identifikácia potrebnej infraštruktúry a nástrojov
- zachytenie obojsmernej trasovateľnosti medzi testovacou bázou, testovacími podmienkami a testovacími prípadmi (viď kapitola 1.4.4)

Spracovanie testovacích podmienok na testovacie prípady a ich sady počas návrhu testov často zahŕňa použitie testovacích techník (viď kapitola 4).

Rovnako ako pri testovacej analýze, môže návrh testov tiež viesť k identifikácii podobných typov defektov v testovacej báze. Rovnako ako pri testovacej analýze, je identifikácia defektov počas návrhu testov dôležitým potenciálnym prínosom.

Implementácia testov

Počas implementácie testov sa vytvorí a/alebo dokončí testvér potrebný na vykonanie testu, vrátane definície postupnosti testovacích prípadov do testovacích procedúr. Takže, návrh testov odpovedá na otázku "ako otestovať?" a implementácia testov odpovedá na otázku "máme k dispozícii všetko potrebné na spustenie testov?".

Implementácia testov pozostáva z nasledujúcich hlavných činností:

- vývoj a prioritizácia testovacích procedúr a potenciálne vytváranie automatizovaných testovacích skriptov
- vytváranie testovacích sád z testovacích procedúr a prípadne automatizovaných testovacích skriptov
- usporiadanie testovacích sád v harmonograme vykonávania testov spôsobom, aby bolo dosiahnuté efektívne vykonanie testov (viď kapitola 5.2.4)
- vytvorenie testovacieho prostredia (potenciálne, vrátane testovacieho vybavenia, virtualizácie služieb, simulátorov a iných položiek infraštruktúry) a overenie, či je všetko potrebné správne nastavené
- príprava testovacích dát a zabezpečenie ich správnej integrácie do testovacieho prostredia
- overenie a aktualizácia obojsmernej trasovateľnosti medzi testovacou bázou, testovacími podmienkami, testovacími prípadmi, testovacími procedúrami a testovacími sadami (viď kapitola 1.4.4). Jednotlivé úlohy sa pri návrhu testov a implementácií testov často kombinujú.

Pri prieskumnom testovaní a iných typoch testovania založených na skúsenostiach, môže dochádzať k tomu, že návrh testov, ich implementácia a aj ich dokumentovanie je súčasťou vykonávania testov. Prieskumné testovanie môže byť založené na testovacích listinách (vytvorených v rámci testovacej analýzy) a vykonávanie prieskumných testov beží súčasne s jeho návrhom a implementáciou (viď kapitola 4.4.2).

Vykonanie testov

Počas vykonávania testov sa testovacie sady spúšťajú v súlade s harmonogramom vykonávania testov.

Vykonávanie testov pozostáva z nasledujúcich hlavných činností:

- zaznamenávanie identifikátorov (IDs) a verzií testovacích položiek alebo testovaných objektov, testovacích nástrojov a testvéru
- vykonávanie testov buď manuálne, alebo pomocou nástrojov na vykonanie testov
- porovnávanie skutočných výsledkov voči očakávaným výsledkom
- analýza anomálií na zistenie ich pravdepodobných príčin (napr. poruchy sa môžu vyskytnúť v dôsledku defektov v kóde, ale zároveň môže dochádzať k falošne-pozitívnym výsledkom, viď kapitola 1.2.3)
- reportovanie defektov na základe zistených zlyhaní (viď kapitola 5.6)
- zaznamenávanie výsledkov testov na základe ich vykonania (napr. test prešiel, test zlyhal, test je zablokovaný)
- opakovanie testovacích činností buď v dôsledku akcie vykonanej v súvislosti s nejakou anomáliou, alebo ako súčasť plánovaného testovania (napr. vykonanie opraveného testu, konfirmačné testovanie a/alebo regresné testovanie)
- overenie a úprava obojsmernej trasovateľnosti medzi testovacou bázou, testovacími podmienkami, testovacími prípadmi, testovacími procedúrami a výsledkami testov.

Dokončenie testovania

Činnosti dokončenia testovania zhromažďujú údaje z dokončených testovacích činností s cieľom konsolidácie skúseností, testvéru a akýchkoľvek ďalších relevantných informácií. Činnosti dokončenia testovania sa vyskytujú pri dosiahnutí projektových míľnikov. Napríklad ako je vydanie softvérového systému, dokončenie testovacieho projektu (alebo jeho zrušenie), dokončenie iterácie v agilnom projekte, dokončenie testovania na jednej úrovni alebo pri dokončení servisného vydania (maintenance release).

Dokončenie testovania pozostáva z nasledujúcich hlavných činností:

- kontrola, či sú všetky reportované defekty uzavreté, zadanie zmenových požiadaviek alebo položiek produktového backlogu na pokrytie defektov, ktoré ostanú nevyriešené
- vytvorenie súhrnnej správy z testovania, ktorá sa má komunikovať na zainteresované strany
- dokončenie a archivácia testovacieho prostredia, testovacích dát, testovacej infraštruktúry a iného testvéru na neskoršie použitie
- odovzdanie testvéru tímom, ktoré sa starajú o údržbu, iným projektovým tímom a/alebo iným zainteresovaným stranám, ktoré by ho mohli využiť
- analýza poznatkov získaných z dokončených testovacích činností na určenie zmien potrebných pre budúce iterácie, vydania a projekty
- použitie nazhromaždených informácií na zlepšenie zrelosti procesu testovania

1.4.3 Pracovné produkty testovania

Pracovné produkty testovania sú vytvorené v rámci procesu testovania. Rovnako ako sa môže líšiť implementácia procesov testovania pre každú organizáciu, tak sa môžu líšiť pracovné produkty, ktoré sú vytvorené počas procesu testovania. A to v spôsobe ako sú tieto pracovné produkty organizované a spravované, a v pomenovaniach týchto pracovných produktov. Táto učebná osnova sa riadi vyššie uvedeným procesom testovania a pracovnými produktmi popísanými v tejto učebnej osnove a v slovníku ISTQB®. Pre ďalšie informácie ohľadom pracovných produktov testovania viď norma *ISO (ISO/IEC/IEEE 29119-3)*.

Mnohé z pracovných produktov testovania opísaných v tejto kapitole možno zaznamenať a spravovať pomocou nástrojov na manažment testovania a nástrojov na manažment defektov (viď kapitola 6).

Pracovné produkty príslušné k plánovaniu testovania

Pracovné produkty príslušné k plánovaniu testovania zvyčajne zahŕňajú jeden alebo viac plánov testovania. Plán testovania obsahuje informácie o testovacej báze, na ktorú sa budú pomocou informácií o trasovateľnosti viazať ďalšie pracovné produkty testovania (viď kapitola 1.4.4), ako aj výstupné kritériá alebo definícia hotového (DoD - Definition of Done), ktoré sa zasa budú používať na monitorovanie a riadenie testovania. Plány testovania sú opísané v kapitole 5.2.

Pracovné produkty príslušné k monitorovaniu a riadeniu testovania

Pracovné produkty príslušné k monitorovaniu a riadeniu testovania zvyčajne zahŕňajú rôzne typy správ z testovania vrátane správ o pokroku v testovaní tvorených priebežne a/alebo pravidelne a súhrnných správ z testovania tvorených po dosiahnutí určitého míľnika. Všetky správy z testovania by mali poskytovať informácie o pokroku v testovaní relevantne danému publiku k danému dátumu, vrátane zhrnutia výsledkov z vykonania testov v momente, keď budú k dispozícií.

Pracovné produkty príslušné k monitorovaniu a riadeniu testovania by sa mali zaoberať aj otázkami v oblasti riadenia projektov, ako je dokončovanie úloh, prideľovanie a využitie zdrojov, a prácnosť projektu.

Monitorovanie a riadenie testovania a pracovné produkty k nim príslušné vytvorené počas týchto činností sú ďalej vysvetlené v kapitole 5.3 tejto učebnej osnovy.

Pracovné produkty príslušné k testovacej analýze

Pracovné produkty príslušné k testovacej analýze zahŕňajú definované testovacie podmienky so stanovenými prioritami, z ktorých každá je v ideálnom prípade obojsmerne trasovateľná voči špecifickému prvku testovacej bázy, ktorú pokrýva. Pri prieskumnom testovaní (exploratory testing) môže

testovacia analýza zahŕňať vytvorenie testovacích listín. Testovacia analýza môže tiež viesť k odhaleniu a reportovaniu defektov v testovacej báze.

Pracovné produkty príslušné k návrhu testov

Výsledkom návrhu testov sú testovacie prípady a sady testovacích prípadov používaných pri vykonávaní testovacích podmienok definovaných v testovacej analýze. Dobrým zvykom je navrhovať všeobecné testovacie prípady bez konkrétnych hodnôt vstupných dát a očakávaných výsledkov. Takéto všeobecné testovacie prípady sa môžu opakovane použiť vo viacerých testovacích cykloch s rôznymi konkrétnymi hodnotami, pričom ostáva primerane zdokumentovaný rozsah testovacieho prípadu. V ideálnom prípade je každý testovací prípad obojsmerne trasovateľný voči testovacím podmienkam, ktoré pokrýva.

Návrh testov tiež vedie k:

- návrhu a/alebo identifikácii potrebných testovacích dát
- návrhu testovacieho prostredia
- identifikácii infraštruktúry a nástrojov

Rozsah, v akom sú tieto výstupy zdokumentované, sa však môže výrazne líšiť.

Pracovné produkty príslušné k implementácii testov

Pracovné produkty príslušné k implementácii testov zahŕňajú:

- testovacie procedúry a určenie poradia týchto testovacích procedúr
- testovacie sady
- harmonogram vykonania testov

V ideálnom prípade, po dokončení implementácie testov, môže byť splnenie kritérií pokrytia stanovených v pláne testovania preukázané prostredníctvom obojsmernej trasovateľnosti medzi testovacími procedúrami a špecifickými prvkami testovacej báze a to prostredníctvom testovacích prípadov a testovacích podmienok.

V niektorých prípadoch zahŕňa implementácia testov vytvorenie pracovných produktov, ktoré využívajú nástroje (napr. virtualizácia služieb) alebo sú využívané nástrojmi (napr. automatizované testovacie skripty).

Implementácia testov môže tiež viesť k vytvoreniu a overeniu testovacích dát a testovacieho prostredia. Úplnosť dokumentácie o výsledkoch verifikácie dát a/alebo prostredia sa môže výrazne líšiť.

Testovacie dáta slúžia na priraďovanie konkrétnych hodnôt pre vstupy a očakávané výsledky testovacích prípadov. Tieto konkrétne hodnoty, spolu s výslovným inštrukciami k ich použitiu, zmenia všeobecné testovacie prípady (high-level) na konkrétne testovacie prípady (low-level). Ten istý všeobecný testovací prípad sa môže vykonať s rôznymi testovacími dátami na rôznych vydaniach testovaného objektu. Konkrétne očakávané výsledky, ktoré sú viazané na konkrétne testovacie dáta sú určené v testovacom orákulu (test-oracle).

Pri prieskumnom testovaní môžu byť niektoré pracovné produkty súvisiace s návrhom a implementáciou testov vytvorené až počas vykonania testov, avšak rozsah, v akom sú prieskumné testy (a ich trasovateľnosť na konkrétne prvky testovacej bázy) dokumentované, sa môže výrazne líšiť.

Testovacie podmienky definované v testovacej analýze sa môžu ďalej vylepšovať pri implementácii testov.

Pracovné produkty príslušné k vykonávaniu testov

Pracovné produkty príslušné k vykonávaniu testov zahŕňajú:

- dokumentáciu o stave jednotlivých testovacích prípadov alebo testovacích procedúr (napr. pripravený na vykonanie, prešiel, zlyhal, zablokovaný, zámerne vynechaný, atď.)
- správy o defektoch (viď kapitola 5.6)
- dokumentáciu o tom, ktoré testovacie položky, testované objekty, testovacie nástroje a testvér boli zapojené do testovania

V ideálnom prípade je možné v okamihu ukončenia vykonania testov určiť a reportovať stav každého prvku testovacej bázy prostredníctvom obojsmernej trasovateľnosti k pridruženej testovacej procedúre (príp. k testovacím procedúram). Môžeme napríklad udať, ktoré požiadavky prešli všetkými plánovanými testami, ktoré požiadavky zlyhali a/alebo majú pridružené defekty, a u ktorých požiadaviek existujú testy, ktoré ešte čakajú na spustenie. To umožňuje verifikovať, či boli splnené kritériá pokrytia, a tiež reportovanie výsledkov testov spôsobom, ktorý je zrozumiteľný pre zainteresované strany.

Pracovné produkty príslušné k dokončeniu testovania

Pracovné produkty príslušné k dokončeniu testovania zahŕňajú súhrnné správy z testovania, opatrenia na zlepšenie budúcich projektov alebo iterácií, zmenové požiadavky alebo položky v produktovom backlogu, alebo dokončený testvér.

1.4.4 Trasovateľnosť medzi testovacou bázou a pracovnými produktmi testovania

Ako sa uvádza v kapitole 1.4.3, pracovné produkty testovania a názvy týchto pracovných produktov sa výrazne líšia. Bez ohľadu na tieto odlišnosti je pre zavedenie efektívneho monitorovania a riadenia testovania dôležité zabezpečiť vytvorenie a následnú údržbu trasovateľnosti medzi každým prvkom testovacej bázy a rôznymi pracovnými produktmi z testovania patriacich k tomuto prvku a to v priebehu celého procesu testovania, ako je popísané vyššie. Okrem vyhodnotenia pokrytia testovania, správna trasovateľnosť pozitívne podporuje aj:

- analýzu vplyvu zmien
- zaistenie auditovateľnosti testovania
- splnenie kritérií danými správou IT
- zlepšenie zrozumiteľnosti správ o pokroku v testovaní a súhrnných správ z testovania poskytujúcich informácie o stave prvkov testovacej báze (napr. požiadavky, ktoré prešli testami, požiadavky, ktoré zlyhali, a požiadavky, ktoré čakajú na vykonanie príslušných testov)
- prezentáciu technických aspektov testovania zainteresovaným stranám, vo forme, ktorej dokážu porozumieť
- poskytovanie informácií na posúdenie kvality produktu, procesných schopností a pokroku prác na projekte voči biznisovým cieľom

Niektoré nástroje na manažment testovania poskytujú modely pracovných produktov testovania, ktoré zodpovedajú časti alebo všetkým pracovným produktom testovania uvedeným v tejto kapitole. Niektoré organizácie zostavujú svoje vlastné systémy riadenia na správu pracovných produktov a zaistenie informácií o trasovateľnosti na mieru svojim potrebám.

1.5 Psychológia testovania

Na vývoji softvéru, vrátane jeho testovania, sa podieľajú ľudské bytosti, preto má ľudská psychika významný vplyv na testovanie softvéru.

1.5.1 Ľudská psychika a testovanie

Identifikácia defektov počas statického testovania (napr. revízia požiadaviek, alebo schôdzka na spresnenie užívateľských scenárov), alebo identifikácia zlyhaní počas dynamického vykonania testov, môže byť vnímaná ako kritika produktu i jeho autora. Prvkom ľudskej psychológie označovaný pojmom „konfirmačné skreslenie“ (alebo tiež potvrdzovacie skreslenie, angl. confirmation bias) opisuje tendenciu človeka ťažko prijímať informácie, ktoré sú v rozpore s jeho názormi. Napríklad, pretože vývojári očakávajú, že ich kód je správny, majú konfirmačné skreslenie, ktoré sťažuje akceptovať fakt, že kód je nesprávny. Okrem konfirmačného skreslenia existujú aj iné kognitívne predsudky (cognitive biases), ktoré môžu spôsobiť, že ľudia ťažko pochopia alebo prijmú informácie získané testovaním. Ďalším obvyklým ľudským rysom je obviňovanie nositeľa zlých správ, a informácie získané testovaním často obsahujú takéto zlé správy.

V dôsledku týchto psychologických faktorov, môžu niektorí ľudia vnímať testovanie ako deštruktívnu činnosť, napriek tomu že výrazne prispieva k pokroku na projekte a ku kvalite produktu (viď kapitola 1.1 a 1.2). Pri snahe znížiť toto negatívne vnímanie by informácie o defektoch a zlyhaniach mali byť komunikované konštruktívne. Týmto spôsobom je možné znížiť napätie medzi testermi a analytikmi, vlastníkmi produktov, návrhármi a vývojármi. To platí pre statické aj dynamické testovanie.

Testeri a manažéri testovania musia byť zruční v oblasti medziľudských vzťahov, aby mohli účinne komunikovať o defektoch, zlyhaniach, výsledkoch testov, pokroku v testovaní a rizikách, a budovať pozitívne vzťahy s kolegami. Spôsoby dobrej komunikácie zahŕňajú nasledujúce príklady:

- Snažte sa spolupracovať a nevyvolávať konflikty. Pripomeňte každému spoločný cieľ zvýšenia kvality systémov.
- Zdôrazňujte prínosy testovania. Napríklad autorom môžu informácie o defekte pomôcť zlepšiť ich pracovné produkty a zručnosti. Organizácii zasa nájdené a opravené defekty počas testovania ušetria čas a peniaze a znížia celkové riziko vzhľadom na kvalitu produktu.
- Komunikujte výsledky testovania a ďalšie zistenia neutrálne so zameraním na fakty bez kritiky osoby, ktorá chybnú položku vytvorila. Píšte objektívne a faktické správy o defektoch a svoje zistenia overujte.
- Snažte sa pochopiť ako sa druhá osoba cíti, a tiež pochopiť dôvody, ktoré môžu prispieť k negatívnej reakcii na podanú informáciu.
- Uistite sa, že druhá osoba pochopila, čo bolo povedané, a naopak.

Typické ciele testovania boli popísané vyššie (viď kapitola 1.1). Jasne definovaný a správny súbor cieľov testovania má významné psychologické dôsledky. Väčšina ľudí má tendenciu zosúladiť svoje plány a správanie s cieľmi stanovenými tímom, manažmentom a ďalšími zainteresovanými stranami. Je tiež dôležité, aby testeri dodržiavali tieto ciele s minimálnou osobnou zaujatosťou.

1.5.2 Zmýšľanie testera a vývojára

Vývojári a testeri často myslia inak. Primárnym cieľom vývoja je navrhnúť a vybudovať produkt. Ako už bolo uvedené, ciele testovania zahŕňajú verifikáciu a validáciu produktu, nájdenie defektov pred vydaním, a tak ďalej. Ide o rôzne súbory cieľov, ktoré si vyžadujú rôzne zmýšľanie. Prepojením týchto odlišných zmýšľaní možno dosiahnuť vyššiu úroveň kvality produktu.

Zmýšľanie každého jednotlivca odráža predpoklady a preferované metódy pre rozhodovanie a riešenie problémov. Zmýšľanie testera by malo zahŕňať zvedavosť, profesionálny pesimizmus, kritické oko, zmysel pre detail, a motiváciu pre dobrú a pozitívnu komunikáciu a vzťahy. Skúsenosťami sa zmýšľanie testera rozširuje a dozrieva.

Zmýšľanie vývojára môže zahŕňať niektoré prvky zmýšľania testera, avšak úspešní vývojári sa často zaujímajú viac o návrh a tvorbu riešení, ako o uvažovanie, čo by mohlo byť na týchto riešeniach zlé. Okrem toho, konfirmačné skreslenie sťažuje uvedomovanie si vlastných chýb.

So správnym zmýšľaním sú vývojári schopní otestovať aj svoj vlastný kód. Rôzne modely životného cyklu vývoja softvéru majú často rôzne spôsoby organizácie testerov a testovacích činností. Vykonávanie niektorých testovacích činností nezávislými testermi, zvyšuje efektívnosť nachádzania defektov, čo je obzvlášť dôležité pri rozsiahlych, zložitých alebo bezpečnostne kritických systémoch. Nezávislí testeri prinášajú novú perspektívu, ktorá je odlišná od autorov pracovných produktov (t. j. biznis analytikov, vlastníkov produktov, návrhárov a vývojárov), pretože majú odlišné kognitívne predsudky oproti autorom.

2 Testovanie v rámci životného cyklu vývoja softvéru – 100 minút

Kľúčové slová

akceptačné testovanie, alfa testovanie, beta testovanie, testovanie súvisiace so zmenami, krabicový softvér (COTS - commercial off-the-shelf), integračné testovanie komponentov, testovanie komponentov, konfirmačné testovanie, zmluvné akceptačné testovanie, funkcionálne testovanie, analýza dopadu, integračné testovanie, údržbové testovanie, nefunkcionálne testovanie, prevádzkové akceptačné testovanie, regresné testovanie, regulatórne akceptačné testovanie, sekvenčný vývojový model, testovanie systémovej integrácie, systémové testovanie, testovacia báza, testovací prípad, testovacie prostredie, úroveň testovania, testovaný objekt, cieľ testovania, typ testu, užívateľské akceptačné testovanie, testovanie bielej skrinky

Študijné ciele pre kapitolu 2 – Testovanie v rámci životného cyklu vývoja softvéru:

2.1 Modely životného cyklu vývoja softvéru

FL-2.1.1 (K2) Vysvetliť vzťahy medzi vývojovými a testovacími činnosťami v rámci životného cyklu vývoja softvéru

FL-2.1.2 (K1) Identifikovať dôvody, prečo sa modely životného cyklu vývoja softvéru musia prispôsobiť kontextu projektu a charakteristikám produktu

2.2 Úrovne testovania

FL-2.2.1 (K2) Porovnať rôzne úrovne testovania z hľadiska cieľov, testovacej bázy, testovaných objektov, typických defektov a zlyhaní, prístupov a zodpovedností

2.3 Typy testov

FL-2.3.1 (K2) Porovnať funkcionálne testovanie, nefunkcionálne testovanie a testovanie bielej skrinky

FL-2.3.2 (K1) Uvedomiť si, že funkcionálne testovanie, nefunkcionálne testovanie a testovanie bielej skrinky prebieha na akejkoľvek úrovni testovania

FL-2.3.3 (K2) Porovnať účel konfirmačného testovania a regresného testovania

2.4 Údržbové testovanie

FL-2.4.1 (K2) Zhrnúť spúšťače údržbového testovania

FL-2.4.2 (K2) Opísať úlohu analýzy dopadu pri údržbovom testovaní

2.1 Modely životného cyklu vývoja softvéru

Model životného cyklu vývoja softvéru opisuje typy činností vykonávaných v každej etape vývoja softvéru a logickú a chronologickú nadväznosť týchto činností. Existuje množstvo rôznych modelov životného cyklu vývoja softvéru, a každý z nich vyžaduje rôzne prístupy k testovaniu.

2.1.1 Vzťah medzi vývojom a testovaním softvéru

Dôležitou súčasťou role testera je, aby sa zoznámil s bežnými modelmi životného cyklu vývoja softvéru, s cieľom zvoliť vhodné testovacie činnosti.

V každom modeli životného cyklu vývoja softvéru existuje niekoľko charakteristík správneho testovania:

- pre každú vývojovú činnosť existuje zodpovedajúca testovacia činnosť
- každá úroveň testovania má svoje špecifické ciele testovania
- testovacia analýza a návrh pre danú úroveň testovania začína počas zodpovedajúcej vývojovej činnosti
- testeri sa zúčastňujú diskusií s cieľom definovať a spresňovať požiadavky a návrhy a sú zapojení do revízie pracovných produktov (napr. požiadaviek, návrhu, užívateľských scenárov atď.), akonáhle sú k dispozícii ich prvé koncepty

Bez ohľadu na to, aký model životného cyklu vývoja softvéru je použitý, testovacie činnosti by mali začať v počiatočných fázach životného cyklu, aby bol dodržaný princíp včasného testovania.

Táto učebná osnova kategorizuje bežné modely životného cyklu vývoja softvéru ako:

- sekvenčné vývojové modely
- iteratívne a inkrementálne vývojové modely

Sekvenčný vývojový model opisuje proces vývoja softvéru ako lineárny, sekvenčný tok činností. To znamená, že každá fáza v procese vývoja by mala začať po tom, čo je predchádzajúca fáza dokončená. Teoreticky sa fázy neprekrývajú, ale v praxi je výhodné získať spätnú väzbu pre nasledujúcu fázu ešte pred dokončením tej predošlej.

Vo vodopádovom sekvenčnom modeli sú vývojové činnosti (napr. analýza požiadaviek, návrh, implementácia, testovanie) dokončované jedna po druhej. Podľa tohto modelu dochádza k testovacím činnostiam len po dokončení všetkých ostatných vývojových činností.

Oproti tomu V-model integruje proces testovania do celého vývojového procesu na základe princípu včasného testovania. V-model naviac obsahuje úrovne testovania, ktoré sú v súlade s príslušnou vývojovou fázou, čo tiež podporuje princíp včasného testovania (viď kapitola 2.2). V tomto modeli prebieha vykonávanie testov v jednotlivých úrovniach testovania sekvenčne, ale v niektorých prípadoch dochádza k ich prekrývaniu.

Sekvenčné vývojové modely poskytujú softvér, ktorý obsahuje kompletnú sadu vlastností, ale zvyčajne vyžadujú mesiace alebo roky na doručenie zainteresovaným stranám a používateľom.

Inkrementálny vývoj predpokladá stanovenie požiadaviek, návrh, implementáciu a testovanie systému po častiach, čo znamená, že jednotlivé softvérové vlastnosti sú vyvíjané postupne. Veľkosť týchto prírastkov alebo inkrementov sa môže líšiť, niektoré metódy pracujú s väčšími prírastkami a iné zas s menšími. Inkrement môže byť malý ako jediná zmena na užívateľskom rozhraní, alebo využitie nového databázového dotazu.

K iteratívnemu vývoju dochádza, keď sú skupiny vlastností špecifikované, navrhnuté, implementované a testované spolu v sérii cyklov (iterácií), často s pevnou dĺžkou. Iterácie môžu zahŕňať zmeny vo vlastnostiach vyvinutých v predchádzajúcich iteráciách spolu so zmenami v rozsahu projektu ako celku. Každá iterácia dodáva funkčný softvér, ktorý je rastúcou podmnožinou požadovanej sady vlastností a tento rast pokračuje až do vytvorenia konečnej podoby softvéru alebo do ukončenia vývoja.

Medzi príklady metodík iteratívneho vývoja patria:

- Rational Unified Process (RUP): Iterácia býva relatívne dlhá (napr. dva až tri mesiace) a teda prírastky vlastností sú adekvátne veľké, napríklad ako dve alebo tri skupiny súvisiacich vlastností.
- Scrum: Iterácia býva relatívne krátka (napr. hodiny, dni alebo niekoľko týždňov) a prírastky vlastností sú adekvátne malé, napríklad drobné vylepšenia, a/alebo dve, tri nové vlastnosti.
- Kanban: Realizuje sa iteráciami s pevnou dĺžkou alebo úplne bez nich. Iterácie môžu dodať hoci len jedno vylepšenie alebo vlastnosť, prípadne môžu nové vlastnosti zoskupovať a vydať ich naraz neskôr.
- Špirálový model: Spočíva v tvorbe experimentálnych prírastkov, z ktorých niektoré môžu byť zásadne prepracované alebo dokonca úplne zahodené.

Pri vývoji komponentov alebo systémov pomocou týchto metód často dochádza počas vývoja k prekrývaniu a opakovaniu jednotlivých úrovní testovania. V ideálnom prípade je každá vlastnosť testovaná na niekoľkých úrovniach testovania a to až do okamžiku konečnej dodávky. V niektorých prípadoch tímy používajú priebežné dodávanie (continuous delivery) alebo priebežné nasadzovanie (continuous deployment). Oba prístupy vyžadujú veľkú mieru automatizácie na viacerých úrovniach testovania. Vývoj s využitím týchto metód tiež zahrňuje koncept samoorganizovaných tímov, ktoré môžu meniť spôsob organizácie testovania i vzťah medzi testerom a vývojárom.

Tieto metódy vytvárajú postupne rastúci systém, ktorý môže byť dodaný koncovým užívateľom vlastnostiach (feature-by-feature), na báze iterácií, alebo tradičnejším spôsobom obsiahlejším vydaním produktu (major release). Bez ohľadu na to, či sú softvérové prírastky dodávané koncovým užívateľom, postupný rast systému zvyšuje dôležitosť regresného testovania.

Na rozdiel od sekvenčných modelov, iteratívne a inkrementálne modely môžu dodávať použiteľný softvér v priebehu týždňov alebo dokonca aj dní, ale dodanie produktu spĺňajúceho všetky požiadavky môže trvať niekoľko mesiacov alebo dokonca rokov.

Pre viac informácií o testovaní softvéru v kontexte agilného vývoja, viď učebné osnovy *ISTQB-CTFL-AT*, *Black 2017*, *Crispin 2008* a *Gregory 2015*.

2.1.2 Modely životného cyklu vývoja softvéru v kontexte

Modely životného cyklu vývoja softvéru musia byť vyberané a prispôsobované kontextu projektu a charakteristikám produktu. Vhodný model životného cyklu vývoja softvéru by mal byť vyberaný a prispôsobovaný na základe cieľa projektu, typu vyvíjaného produktu, biznis priorít (napr. doba zavedenia produktu na trh - time-to-market) a identifikovaných projektových a produktových rizík. Napríklad vývoj a testovanie menšieho interného administratívneho systému by sa mali odlišovať od vývoja a testovania bezpečnostne kritického systému, ako je brzdový systém automobilov. Iným príkladom môže byť obmedzenie komunikácie medzi členmi tímu z dôvodu organizačných a kultúrnych problémov, ktorá môže brániť iteratívnemu vývoju.

V závislosti od kontextu projektu, môže byť potrebné kombinovať alebo reorganizovať úrovne testovania a/alebo testovacie činnosti. Napríklad v prípade integrácie komerčného krabicového softvéru do väčšieho systému, môže kupujúci vykonať testovanie interoperability (schopnosti spolupráce) na úrovni testovania systémovej integrácie (napr. integrácia do infraštruktúry a iných systémov) a na úrovni akceptačného

testovania (funkcionálne a nefunkcionálne testovanie, spolu s užívateľským akceptačným testovaním a prevádzkovým akceptačným testovaním). Viac informácií nájdete v kapitole 2.2 Úrovne testovania a kapitole 2.3 Typy testov.

Okrem toho modely životného cyklu vývoja softvéru sa môžu kombinovať. Napríklad V-model môže byť použitý pre vývoj a testovanie backend systému a jeho integrácie, zatiaľ čo agilný vývoj model môže byť použitý na vývoj a testovanie front-end užívateľských rozhraní a funkčnosti. Prototypovanie môže byť použité na začiatku projektu a po skončení experimentálnej fázy sa môže nasadiť model inkrementálneho vývoja.

Systémy Internetu vecí (Internet of Things - IoT), ktoré pozostávajú z mnohých rôznych objektov (zariadenia, produkty a služby), zvyčajne uplatňujú rôzne modely životného cyklu vývoja softvéru pre rôzne objekty. To samo o sebe predstavuje veľkú výzvu pre spravovanie vývojových verzií týchto systémov. Navyše životný cyklus vývoja softvéru týchto objektov kladie väčší dôraz na fázy vývoja po zavedení do prevádzky (ako napr. samotná prevádzka, aktualizácie a vyraďovanie z prevádzky).

Dôvody, prečo musia byť modely vývoja softvéru prispôsobené kontextu projetku a produktovým charakteristikám sú napríklad:

- Rozdiely v prokuktových rizikách systémov (komplexné alebo jednoduché projekty)
- Rôzne organizačné jednotky môžu byť súčasťou projektu alebo programu (kombinácie sekvenčného a agilného vývoju)
- Krátka doba na dodanie produktu na trh (zlúčenie testovacích úrovní a/alebo integrácia typov testov do testovacích úrovní)

2.2 Úrovne testovania

Úrovne testovania sú skupiny testovacích činností, ktoré sú organizované a riadené spoločne. Každá úroveň testovania je inštancia procesu testovania pozostávajúca z činností popísaných v kapitole 1.4, vykonaných vo vzťahu so softvérom na danej úrovni vývoja, od jednotlivých jednotiek alebo komponentov až po kompletné systémy a prípadne systémy systémov. Úrovne testovania súvisia aj s inými činnosťami v rámci životného cyklu vývoja softvéru. Úrovne testovania použité v tejto učebnej osnove sú:

- testovanie komponentov
- integračné testovanie
- systémové testovanie
- akceptačné testovanie

Úrovne testovania sú charakterizované nasledujúcimi atribútmi:

- špecifické ciele
- testovacia báza, z ktorej sú odvodené testovacie prípady
- testovací objekt (t. j. čo sa testuje)
- typické defekty a zlyhania
- špecifické prístupy a zodpovednosti

Každá úroveň testovania vyžaduje vhodné testovacie prostredie. Pre akceptačné testovanie je to napríklad prostredie odpovedajúce produkčnému prostrediu, zatiaľ čo pre testovanie komponentov vývojári zvyčajne používajú svoje vlastné vývojové prostredie.

2.2.1 Testovanie komponentov

Ciele testovania komponentov

Testovanie komponentov (známe aj ako jednotkové testovanie alebo testovanie modulov) sa zameriava na komponenty, ktoré sú testovateľné samostatne. Medzi ciele testovania komponentov patria:

- zníženie rizík
- verifikácia, či funkcionálne a nefunkcionálne správanie komponentu zodpovedá návrhu a špecifikácii
- budovanie dôvery v kvalitu komponentu
- hľadanie defektov v komponente
- prevencia preniknutia defektov na vyššie úrovne testovania

V niektorých prípadoch, najmä v inkrementálnych a iteratívnych vývojových modeloch (napr. agilných), v ktorých prebiehajú nepretržité zmeny kódu, zohráva automatizované regresné testovanie komponentov kľúčovú úlohu pri budovaní dôvery, že zmeny nenarušili existujúce komponenty.

Testovanie komponentov sa často vykonáva izolovane od zvyšku systému v závislosti od modelu životného cyklu vývoja softvéru a od systému. Môže vyžadovať imitáciu (mock) objektov, virtuálizáciu služieb, sady testovacieho vybavenia (harnesses), stuby (stubs) a ovládače (drivers). Testovanie komponentov sa môže týkať funknčnosti (napr. správnosti výpočtov), nefunkcionálnych charakteristík (napr. hľadania úniku pamäte) a štrukturálnych vlastností (napr. testovania rozhodnutí).

Testovacia báza

Príklady pracovných produktov, ktoré možno použiť ako testovaciu bázu pre testovanie komponentov, zahŕňajú:

- detailný návrh
- kód
- dátový model
- špecifikácia komponentu

Testované objekty

Typické testované objekty pre testovanie komponentov zahŕňajú:

- komponenty, jednotky alebo moduly
- kód a dátové štruktúry
- triedy
- databázové moduly

Typické defekty a zlyhania

Príklady typických defektov a zlyhaní v rámci testovania komponentov zahŕňajú:

- nesprávnu funkčnosť (napr. iná, ako je popísané v špecifikácií návrhu)
- problémy dátových tokov
- nesprávny kód a logika

Defekty sú typicky opravené hneď ako sú nájdené a to často bez formálneho manažmentu defektu. Avšak, keď vývojári defekt nahlásia, poskytujú tak dôležité informácie pre analýzu koreňovej príčiny a zlepšenie procesu.

Špecifické prístupy a zodpovednosti

Testovanie komponentov zvyčajne vykonáva vývojár, ktorý napísal kód. Testovanie vždy vyžaduje prístup k testovanému kódu. Vývojári môžu pracovať na vývoji komponentov alebo sa zaoberať nachádzaním a opravou defektov. Vývojári často píšu a spúšťajú testy až po napísaní kódu komponentu. Avšak, obzvlášť v Agilnom vývoji môže implementácia automatických testov komponentov predchádzať vlastnej implementácií kódu.

Typickým príkladom takéhoto prístupu je vývoj riadený testovaním (test-driven development, TDD). Vývoj riadený testovaním je vysoko iteratívny prístup. Je založený na cykloch, v ktorých sa vyvíjajú automatizované testovacie prípady, následne sa implementujú a integrujú malé časti kódu, vykonávajú sa testy komponentov a opravujú sa všetky problémy spoločne s prepracovávaním kódu (refactoring). Tento proces pokračuje až do dokončenia všetkých komponentov a úspešného prejdenia všetkých testov. Vývoj riadený testovaním je príkladom prístupu nazývaného testy najskôr (test-first). Zatiaľ čo vývoj riadený testovaním vznikol v rámci extrémneho programovania (eXtreme Programming, XP), rozšíril sa aj do iných foriem agilných a sekvenčných životných cyklov vývoja (viď učebné osnovy *ISTQB-CTFL-AT*).

2.2.2 Integračné testovanie

Ciele integračného testovania

Integračné testovanie sa zameriava na vzájomné interakcie medzi komponentmi alebo systémami. Medzi ciele integračného testovania patrí:

- zníženie rizík
- verifikácia, či funkcionálne a nefunkcionálne správanie rozhraní zodpovedá návrhu a špecifikácii
- budovanie dôvery v kvalitu rozhraní
- hľadanie defektov (ktoré môžu byť v samotných rozhraniach alebo v komponentoch či systémoch)
- prevencia preniknutia defektov na vyššie úrovne testovania

Rovnako ako pri testovaní komponentov, automatizované integračné regresné testy v niektorých prípadoch poskytujú istotu, že zmeny nenarušili funkčnosť existujúcich rozhraní, komponentov alebo systémov.

Táto učebná osnova popisuje dve rôzne úrovne integračného testovania, ktoré môžu byť vykonávané na testovaných objektoch rôznej veľkosti takto:

- Integračné testovanie komponentov sa zameriava na interakcie a rozhrania medzi integrovanými komponentmi. Integračné testovanie komponentov sa vykonáva po testovaní komponentov, a je všeobecne automatizované. V iteratívnom a inkrementálnom vývoji sú integračné testy komponentov zvyčajne zahrnuté v procese priebežnej integrácie.
- Testovanie systémovej integrácie sa zameriava na interakcie a rozhrania medzi systémami, balíkmi a mikroslužbami. Testovanie systémovej integrácie môže zahŕňať aj interakcie a rozhrania poskytované externými organizáciami (napr. webové služby). V tomto prípade, organizácie vyvíjajúce softvér nemajú vplyv na externé rozhrania, čo môže spôsobovať rôzne problémy pri testovaní (napr. zabezpečiť odstránenie defektov blokujúcich testovanie externou organizáciou, zaistenie testovacieho prostredia, atď.). Testovanie systémovej integrácie sa môže vykonať po systémovom testovaní alebo súbežne s prebiehajúcim testovaním systému a to ako v sekvenčnom, tak aj v iteratívnom a inkrementálnom vývoji.

Testovacia báza

Príklady pracovných produktov, ktoré možno použiť ako testovaciu bázu pre integračné testovanie, zahŕňajú:

- softvér a návrh systému
- sekvenčné diagramy
- špecifikácie rozhraní a komunikačných protokolov
- prípady použitia
- architektúru na úrovni komponentov alebo systémov
- pracovné toky
- definície externých rozhraní

Testované objekty

Typické testované objekty pre integračné testovanie sú:

- subsystémy
- databázy
- infraštruktúra
- rozhrania
- API
- mikroslužby

Typické defekty a zlyhania

Príklady typických defektov a zlyhaní pre integračné testovanie komponentov sú:

- nesprávne dáta, chýbajúce dáta alebo nesprávne kódovanie dát
- nesprávne sekvencie alebo časovanie volaní rozhrania
- nesúlad rozhraní
- zlyhania v komunikácii medzi komponentmi
- neodchytené alebo nesprávne ošetrené zlyhania komunikácie medzi komponentmi
- nesprávne predpoklady týkajúce sa významu, jednotiek alebo hraníc dát, predávaných medzi komponentmi

Príklady typických defektov a zlyhaní pre testovanie systémovej integrácie sú:

- nekonzistentné štruktúry správ medzi systémami
- nesprávne dáta, chýbajúce dáta alebo nesprávne kódovanie dát
- nesúlad rozhraní
- zlyhania v komunikácii medzi systémami
- neodchytené alebo nesprávne ošetrené zlyhania komunikácie medzi systémami
- nesprávne predpoklady týkajúce sa významu, jednotiek alebo hraníc dát, predávaných medzi systémami
- nedodržanie povinných bezpečnostných predpisov

Špecifické prístupy a zodpovednosti

Integračné testy komponentov a integračné testy systémov by sa mali sústrediť na samotnú integráciu. Napríklad pri integrácii modulu A s modulom B by sa mali testy zameriavať na komunikáciu medzi modulmi, nie na funkčnosť jednotlivých modulov, ktorá by mala byť pokrytá v rámci testovania komponentov. Ak sa integruje systém X so systémom Y, testy by sa mali zamerať na komunikáciu medzi

systémami, nie na funkčnosť jednotlivých systémov, ktorá by mala byť pokrytá v rámci systémového testovania. V tomto prípade sú použiteľné ako funkcionálne, nefunkcionálne, tak aj štrukturálne testy.

Integračné testovanie komponentov je často zodpovednosťou vývojárov. Testovanie systémovej integrácie je vo všeobecnosti zodpovednosťou testerov. V ideálnom prípade by testeri vykonávajúci testovanie systémovej integrácie mali porozumieť systémovej architektúre a mali by mať slovo pri plánovaní integrácie.

Ak sú integračné testy a integračná stratégia plánované pred vytvorením komponentov alebo systémov, mali by byť komponenty alebo systémy vytvorené v poradí podporujúcom efektívnosť testovania. Systematické integračné stratégie môžu vychádzať zo systémovej architektúry (napr. zhora nadol a zdola nahor), funkcionálnych úloh, sekvencií spracovania transakcií alebo iných aspektov systémov alebo komponentov. S cieľom zjednodušiť izoláciu defektov a detekovanie defektov včas by integrácia mala byť zvyčajne inkrementálna (t. j. po malých prírastkoch v počte komponentov alebo systémov v čase) namiesto "veľkého tresku" (t. j. integrácia všetkých komponentov alebo systémov v jedinom kroku). Analýza rizík z najkomplexnejších rozhraní môže pomôcť so správnym zameraním integračného testovania.

Čím väčší je rozsah integrácie, tým ťažšie sa dajú izolovať defekty na konkrétny komponent alebo systém, čo môže viesť k zvýšenému riziku a času potrebnému na riešenie problémov. To je jeden z dôvodov, prečo sa priebežná integrácia, pri ktorej je softvér integrovaný komponent po komponente (t. j. funkcionálna integrácia), stala bežnou praxou. Takáto priebežná integrácia často zahŕňa automatizované regresné testovanie, v ideálnom prípade na viacerých úrovniach testovania.

2.2.3 Systémové testovanie

Ciele systémového testovania

Systémové testovanie sa zameriava na správanie a schopnosti celého systému alebo produktu. Často zahŕňa end-to-end úlohy, ktoré vie systém vykonať aj ako nefunkcionálne správanie počas vykonávania týchto úloh. Medzi ciele systémového testovania patria:

- zníženie rizík
- verifikácia, či funkcionálne a nefunkcionálne správanie systému zodpovedá návrhu a špecifikácii
- validácia, že systém je kompletný a bude fungovať podľa očakávania
- budovanie dôvery v kvalitu systému ako celku
- hľadanie defektov
- prevencia preniknutia defektov na vyššie úrovne testovania alebo do produkcie

Pre určité systémy môže byť cieľom testovania aj overenie kvality dát. Rovnako ako pri testovaní komponentov a integračnom testovaní, automatizované systémové regresné testy v niektorých prípadoch poskytujú istotu, že zmeny nenarušili existujúce vlastnosti alebo end-to-end schopnosti systému. Systémové testovanie často poskytuje informácie využívané zainteresovanými stranami pri rozhodovaní o vydaní systému do prevádzky. Systémové testovanie môže byť vyžadované z dôvodu zákonných alebo regulatórnych požiadaviek alebo noriem.

Testovacie prostredie by malo v ideálnom prípade zodpovedať konečnému cieľovému alebo prevádzkovému prostrediu.

Testovacia báza

Príklady pracovných produktov, ktoré možno použiť ako testovaciu bázu pre systémové testovanie, zahŕňajú:

- špecifikácie systémových a softvérových požiadaviek (funkcionálnych a nefunkcionálnych)
- správy o analýze rizík
- prípady použitia
- epicy a užívateľské scenáre
- modely správania systému
- diagramy stavov
- systémové a používateľské príručky

Testované objekty

Typické testované objekty pri systémovom testovaní zahŕňajú:

- aplikácie
- hardvérové/softvérové systémy
- operačné systémy
- testovaný systém (SUT - system under test)
- konfiguráciu systému a konfiguračné dáta

Typické defekty a zlyhania

Príklady typických defektov a zlyhaní pre systémové testovanie zahŕňajú:

- nesprávne výpočty
- nesprávne alebo neočakávané funkcionálne a nefunkcionálne správanie systému
- nesprávne riadenie a/alebo dátové toky v rámci systému
- neschopnosť správne a úplne vykonať end-to-end funkcionálne úlohy
- neschopnosť systému riadne fungovať v systémovom prostredí
- neschopnosť systému pracovať tak, ako je opísané v systémových a používateľských príručkách

Špecifické prístupy a zodpovednosti

Systémové testovanie by sa malo zameriavať na celkové, end-to-end správanie systému a to ako funkcionálne, tak nefunkcionálne. Systémové testovanie by malo používať najvhodnejšie techniky (viď kapitola 4) pre aspekty systému, ktoré sa majú testovať. Napríklad sa pomocou rozhodovacej tabuľky môže verifikovať, či je funkcionálne správanie systému také ako je popísané v biznis pravidlách.

Systémové testovanie zvyčajne vykonávajú nezávislí testeri, ktorí sa vo veľkej miere spoliehajú na špecifikácie. Defekty v špecifikáciách (napr. chýbajúce užívateľské scenáre, nesprávne zadané biznis požiadavky, atď.) môžu viesť k nedorozumeniam alebo sporom o očakávanom správaní systému. Takéto situácie môžu viesť k falošne-pozitívnym a falošne-negatívnym výsledkom testovania, ktoré zdržujú testovanie a znižujú účinnosť detekcie defektov. Včasné zapojenie testerov do spresňovania užívateľských scenárov alebo do statického testovania, ako sú rôzne typy revízií, pomáha znižovať výskyt takýchto situácií.

2.2.4 Akceptačné testovanie

Ciele akceptačného testovania

Akceptačné testovanie, podobne ako systémové testovanie, sa zvyčajne zameriava na správanie a schopnosti celého systému alebo produktu. Ciele akceptačného testovania zahŕňajú:

- budovanie dôvery v kvalitu systému ako celku
- validácia, že systém je kompletný a bude fungovať podľa očakávania
- verifikácia, či funkcionálne a nefunkcionálne správanie zodpovedá špecifikácii

Akceptačné testovanie môže poskytnúť informácie na posúdenie pripravenosti systému na nasadenie a používanie zákazníkom (resp. koncovým používateľom). Počas akceptačného testovania môžu byť nájdené defekty, avšak nachádzanie defektov nie je jeho cieľom. Nájdenie veľkého počtu defektov počas akceptačného testovania môže byť v niektorých prípadoch považované za významné projektové riziko. Akceptačné testovanie môže byť vyžadované z dôvodu zákonných alebo regulatórnych požiadaviek alebo noriem.

Medzi bežné formy akceptačného testovanie patria:

- užívateľské akceptačné testovanie
- prevádzkové akceptačné testovanie
- zmluvné a regulatórne akceptačné testovanie
- alfa a beta testovanie

Nasledujúce podkapitoly obsahujú popis týchto foriem akceptačného testovania.

Užívateľské akceptačné testovanie (UAT – User Acceptance Testing)

Užívateľské akceptačné testovanie systému sa zameriava na validáciu vhodnosti systému na použitie plánovanými užívateľmi v reálnom alebo simulovanom prevádzkovom prostredí. Hlavným cieľom je budovanie dôvery užívateľov v to, že systém bude spĺňať ich potreby a požiadavky a umožní bezproblémovo vykonávať biznis procesy s minimálnymi nákladmi a rizikami.

Prevádzkové akceptačné testovanie (OAT – Operational Acceptance Testing)

Akceptačné testovanie systému prevádzkovým personálom alebo správcami systémov sa zvyčajne vykonáva v simulovanom produkčnom prostredí. Testy sa zameriavajú na operačné aspekty a môžu zahŕňať:

- testovanie zálohovania a obnovy systému
- testovanie inštalácie, odinštalácie a aktualizácie
- testovanie zotavenia po havárii
- testovanie správy užívateľov
- testovanie úloh údržby systému
- testovanie načítania a migrácie dát
- kontrolu zraniteľností bezpečnosti (security)
- testovanie výkonnosti

Hlavným cieľom prevádzkových akceptačných testov je budovanie dôvery v to, že operátori a správcovia systému budú schopní zabezpečiť správne fungovanie systému pre používateľov v prevádzkovom prostredí, a to aj za výnimočných alebo ťažkých podmienok.

Zmluvné a regulatórne akceptačné testovanie

Zmluvné akceptačné testovanie je vykonávané oproti zmluvným akceptačným kritériám pre softvér vyvinutý na mieru. Akceptačné kritériá by mali byť definované v dobe odsúhlasenia kontraktu zmluvnými stranami. Zmluvné akceptačné testovanie je často vykonávané užívateľmi systému alebo nezávislými testermi.

Regulatórne akceptačné testovanie sa vykonáva oproti akýmkoľvek predpisom, ktoré musia byť systémom dodržané, ako napríklad vládne, právne, alebo bezpečnostné predpisy. Regulatórne akceptačné testovanie je často vykonávané užívateľmi systému alebo nezávislými testermi, niekedy však na výsledky testov dohliada, prípadne ich reviduje regulatórny orgán.

Hlavným cieľom zmluvných a regulatórnych akceptačných testov je budovanie dôvery v to, že sa dosiahol zmluvný či regulatórny súlad.

Alfa a beta testovanie

Alfa a beta testovanie sú zvyčajne používané vývojármi komerčného krabicového softvéru, ktorí chcú získať spätnú väzbu od potenciálnych alebo existujúcich užívateľov, zákazníkov a/alebo operátorov, než uvedú softvérový produkt na trh. Alfa testovanie sa vykonáva v organizácii, ktorá vyvíja softvér, nie však vývojovým tímom, ale potenciálnymi alebo existujúcimi zákazníkmi, a/alebo operátormi, či tímom nezávislých testerov. Beta testovanie je vykonávané potenciálnymi alebo existujúcimi zákazníkmi, a/alebo operátormi priamo na ich pracoviskách. Beta testovanie môže byť vykonané až po alfa testovaní, ale môže byť vykonané aj bez predchádzajúceho alfa testovania.

Jedným z cieľov alfa a beta testovania je budovanie dôvery medzi potenciálnymi alebo existujúcimi zákazníkmi a/alebo prevádzkovateľmi v to, že budú schopní používať systém za normálnych, každodenných podmienok a v prevádzkovom prostredí tak, aby bezproblémovo dosiahli svoje ciele s minimálnymi ťažkosťami, nákladmi a rizikami. Ďalším cieľom môže byť nájdenie defektov súvisiacich s podmienkami a prostredím, v ktorom sa systém bude používať, najmä ak sú tieto podmienky a prostredia ťažko replikovateľné vývojovým tímom.

Testovacia báza

Príklady pracovných produktov, ktoré možno použiť ako testovaciu bázu pre akúkoľvek formu akceptačného testovania, zahŕňajú:

- biznis procesy
- užívateľské alebo biznis požiadavky
- predpisy, zmluvy a normy
- prípady použitia a/alebo užívateľské scenáre
- systémové požiadavky
- systémovú alebo užívateľskú dokumentáciu
- inštalačné postupy
- správy o analýze rizík

Okrem toho, ako testovaciu bázu na odvodenie testovacích prípadov pre prevádzkové akceptačné testovanie je možné použiť jeden alebo viacero z nasledujúcich pracovných produktov:

- postupy zálohovania a obnovy
- postupy zotavenia po havárii
- nefunkcionálne požiadavky
- prevádzkovú dokumentáciu
- pokyny pre nasadenie a inštaláciu
- výkonnostné ciele
- databázové balíky
- bezpečnostné normy alebo predpisy

Typické testované objekty

Typické testované objekty pre akúkoľvek formu akceptačného testovania zahŕňajú:

- testovaný systém (SUT – System Under Test)
- konfiguráciu systému a konfiguračné dáta
- biznis procesy pre plne integrovaný systém

- systémy obnovy a pracoviská schopné bez prerušenia prevziať funkciu pôvodných pracovísk (pre testovanie kontinuity biznisu a obnovy po havárii)
- prevádzkové a údržbové procesy
- formuláre
- správy (reporty)
- existujúce a konvertované produkčné dáta

Typické defekty a zlyhania

Príklady typických defektov pre akúkoľvek formu akceptačného testovania zahŕňajú:

- systémové pracovné toky nespĺňajúce biznis alebo užívateľské požiadavky
- nesprávne implementované biznis pravidlá
- systém nespĺňa zmluvné alebo regulatórne požiadavky
- zlyhania nefunkcionálnych vlastností systému, ako sú zraniteľnosti bezpečnosti, výkonnosť pri vysokej záťaži alebo nesprávna prevádzka na určitej podporovanej platforme

Špecifické prístupy a zodpovednosti

Akceptačné testovanie je často zodpovednosťou zákazníkov, biznis užívateľov, vlastníkov produktov alebo operátorov systému, ale je možné zapojiť aj ďalšie zainteresované strany.

Akceptačné testovanie sa často považuje za poslednú úroveň testovania v sekvenčnom vývojovom cykle, ale môže sa vyskytnúť aj inokedy napríklad:

- akceptačné testovanie komerčného krabicového softvéru sa dá vykonať v momente, keď je softvér nainštalovaný alebo zintegrovaný
- akceptačné testovanie vylepšenej funkčnosti sa môže vykonať pred systémovým testovaním

V iteratívnom vývoji môžu projektové tímy používať rôzne formy akceptačného testovania počas a na konci každej iterácie. Napríklad akceptačné testy na verifikáciu novej vlastnosti voči akceptačným kritériám a validáciu, že nová vlastnosť vyhovuje potrebám užívateľov. Okrem toho sa môžu vyskytnúť alfa a beta testy, a to buď na konci každej iterácie, po dokončení každej iterácie, alebo po dokončení sérii iterácií. Rovnako užívateľské akceptačné testy, prevádzkové akceptačné testy, regulatórne akceptačné testy a zmluvné akceptačné testy môžu byť vykonávané buď na konci každej iterácie, po dokončení každej iterácie, alebo po dokončení série iterácií.

2.3 Typy testov

Typ testu je skupina testovacích činností zameraných na testovanie špecifických charakteristík softvérového systému alebo jeho časti na základe špecifických testovacích cieľov. Takéto ciele môžu zahŕňať:

- vyhodnotenie funkcionálnych charakteristík kvality, ako je úplnosť, správnosť a vhodnosť
- vyhodnotenie nefunkcionálnych charakteristík kvality, ako je spoľahlivosť, výkonnosť, bezpečnosť, kompatibilita a použiteľnosť
- vyhodnotenie, či je štruktúra alebo architektúra komponentu alebo systému správna, úplná a spĺňa špecifikáciu
- vyhodnotenie dopadov zmien, ako je potvrdenie, že defekty boli opravené (konfirmačné testovanie) a hľadanie neúmyselných zmien v správaní vyplývajúcich zo zmien softvéru alebo zmien prostredia (regresné testovanie)

2.3.1 Funkcionálne testovanie

Funkcionálne testovanie systému zahŕňa testy, ktoré vyhodnocujú funkcie, ktoré by mal systém vykonávať. Funkcionálne požiadavky môžu byť popísané v pracovných produktoch ako sú: špecifikácie biznis požiadaviek, epicoch, užívateľských scenároch, prípadoch použitia alebo funkcionálnej špecifikácie, no môžu byť aj nezdokumentované. Funkcie sú to, "čo" by mal systém robiť.

Funkcionálne testy by sa mali vykonávať na všetkých úrovniach testovania (napr. testy komponentov môžu byť založené na špecifikácii komponentu), i keď sa zameranie na jednotlivých úrovniach líši (viď kapitola 2.2).

Funkcionálne testovanie hodnotí správanie softvéru, preto je možné použiť techniky čiernej skrinky na odvodenie testovacích podmienok a testovacích prípadov pre funkčnosť komponentu alebo systému (viď kapitola 4.2).

Dôkladnosť funkcionálneho testovania sa dá merať rozsahom funkcionálneho pokrytia. Funkcionálne pokrytie je miera, v akej bola určitá funkčnosť preskúmaná testami a je vyjadrené ako percento pokrytia daného prvku testami. Napríklad pri použití trasovateľnosti medzi testami a funkcionálnymi požiadavkami je možné vypočítať percentuálny podiel týchto požiadaviek, ktoré sú predmetom testov a potenciálne identifikovať medzery v pokrytí.

Návrh a vykonávanie funkcionálnych testov môže vyžadovať špeciálne zručnosti alebo znalosti, ako je znalosť konkrétneho biznis problému, ktorý softvér rieši (napr. softvér pre geologické modelovanie pre ropný a plynárenský priemysel).

2.3.2 Nefunkcionálne testovanie

Nefunkcionálne testovanie vyhodnocuje charakteristiky systémov a softvéru, ako je: použiteľnosť, výkonnostná efektivita alebo bezpečnosť. Klasifikáciu charakteristík kvality softvérových produktov rieši norma *ISO (ISO/IEC 25010)*. Nefunkcionálne testovanie je testovanie "ako dobre" sa systém chová.

Na rozdiel od bežných (mylných) predstáv, nefunkcionálne testovanie sa môže a často by sa malo vykonávať na všetkých úrovniach testovania, a to čo najskôr. Neskoré objavenie nefunkcionálneho defektu môže byť veľmi nebezpečné pre úspech projektu.

Na odvodenie testovacích podmienok a prípadov pre nefunkcionálne testovanie sa dajú použiť techniky testovania čiernej skrinky (viď kapitola 4.2). Napríklad, analýza hraničných hodnôt sa môže použiť na definovanie podmienok extrémnej záťaže pre výkonnostné testovanie.

Dôkladnosť nefunkcionálneho testovania sa dá merať prostredníctvom nefunkcionálneho pokrytia. Nefunkcionálne pokrytie je miera, do akej je určitý typ nefunkcionálneho prvku preskúmaný testami a vyjadruje sa ako percento pokrytia testami pre daný typ prvku. Napríklad pomocou trasovateľnosti medzi testami a podporovanými zariadeniami pre mobilnú aplikáciu je možné vypočítať percento zariadení, ktoré sú pokryté testovaním kompatibility a potenciálne identifikovať medzery v pokrytí.

Návrh a vykonávanie nefunkcionálnych testov môže vyžadovať špeciálne zručnosti alebo znalosti, ako je: znalosť slabých stránok návrhu alebo technológie (napr. zraniteľností bezpečnosti spojených s určitými programovacími jazykmi) alebo konkrétne užívateľské základne (napr. persony užívateľov zdravotníckeho systému).

Ďalšie podrobnosti o testovaní nefunkcionálnych charakteristík kvality sú uvedené v učebných osnovách *ISTQB-CTAL-TA*, *ISTQB-CTAL-TTA*, *ISTQB-CTAL-SEC* a v ďalších špecializovaných moduloch ISTQB®.

2.3.3 Testovanie bielej skrinky

Testovanie bielej skrinky odvodzuje testy na základe internej štruktúry alebo implementácie systému. Interná štruktúra môže zahŕňať kód, architektúru, pracovné toky a/alebo dátové toky v rámci systému (viď kapitola 4.3).

Dôkladnosť testovania bielej skrinky možno merať prostredníctvom štrukturálneho pokrytia. Štrukturálne pokrytie je miera, do akej je určitý štrukturálny prvok preskúmaný testami a vyjadruje sa ako percento prvkov daného typu, ktoré bolo pokryté testovaním.

Na úrovni testovania komponentov je pokrytie kódu založené na kóde komponentu, ktorý bol testovaný. Môže sa merať z hľadiska rôznych aspektov kódu (položky pokrytia), ako je procento vykonateľných príkazov testovaných v rámci komponentu, alebo percento výsledkov rozhodnutí, ktoré boli testované. Tieto typy pokrytia sa spoločne nazývajú pokrytie kódu. Na úrovni integračného testovania komponentov môže byť testovanie bielej skrinky založené na architektúre systému, ako sú rozhrania medzi komponentmi a pokrytie štruktúry môže byt merané z hľadiska percentuálneho podielu rozhraní preskúmaných testami.

Návrh a prevedenie testov bielej skrinky môže vyžadovať špeciálne zručnosti alebo znalosti, ako je napríklad spôsob implementácie kódu (napr. pri používaní nástrojov na meranie pokrytia kódu), ako sa ukladajú údaje (napr. na vyhodnotenie možných databázových dotazov) a ako používať samotné nástroje pokrytia a správne interpretovať ich výsledky.

2.3.4 Testovanie súvisiace so zmenami

Po vykonaní zmien v systéme (napr. oprava chyby, doručenie novej alebo zmena existujúcej funkčnosti), je nutné zaistiť testovanie, ktoré potvrdí, že zmena skutočne odstránila defekt alebo implementovala danú funkčnosť správne, a nespôsobila pritom žiadne nepredvídané a nežiaduce následky.

- Konfirmačné testovanie: Po oprave defektu by mal softvér byť otestovaný vykonaním všetkých testovacích prípadov, ktoré zlyhali v dôsledku defektu a to na novej verzii softvéru. Softvér môže byť testovaný aj novými testami, ktoré pokrývajú zmeny nutné na opravu defektu. Minimálne treba na novej verzii softvéru vykonať kroky, ktoré viedli k zlyhaniu spôsobenému defektom. Účelom konfirmačného testu je potvrdiť, či bol pôvodný defekt úspešne opravený.
- Regresné testovanie: Stáva sa, že zmena vykonaná v jednej časti kódu, či už oprava alebo iný typ zmeny, môže náhodne ovplyvniť správanie ostatných častí kódu, či už v rámci toho istého komponentu, v iných komponentoch toho istého systému, alebo dokonca v iných systémoch. Zmeny môžu zahŕňať zmeny v prostredí, ako je napríklad nová verzia operačného alebo databázového systému. Takýmto nežiaducim vedľajším účinkom sa hovorí regresia. Regresné testovanie zahŕňa spúšťanie testov na detekciu takýchto nežiaducich vedľajších účinkov.

Konfirmačné testovanie a regresné testovanie sa vykonávajú na všetkých úrovniach.

Najmä v iteratívnych a inkrementálnych vývojových cykloch (napr. agilnom vývoji) vedie pridávanie nových vlastností, zmeny existujúcich vlastností a prepracovávanie kódu (refactoring), k častým zmenám kódu, ktorý následne vyžaduje testovanie súvisiace so zmenami. Vzhľadom k priebežnému vývoju a rozširovaniu systému je konfirmačné a regresné testovanie veľmi dôležité. A to obzvlášť pre systémy Internetu vecí, kde prebieha častá aktualizácia alebo výmena jednotlivých objektov (napr. zariadení).

Regresné testovacie sady sú spúšťané mnohokrát a vo všeobecnosti sa menia pomaly, takže regresné testovanie je silným kandidátom na automatizáciu. Automatizácia týchto testov by sa mala začať na začiatku projektu (viď kapitola 6).

2.3.5 Typy testov a úrovne testovania

Každý z vyššie uvedených typov testu je možné vykonať na ľubovoľnej úrovni testovania.
Pre znázornenie sú nižšie uvedené príklady funkcionálnych a nefunkcionálnych testov, testov bielej skrinky a testov súvisiacich so zmenami naprieč všetkými testovacími úrovňami pre bankovú aplikáciu.
Príklady funkcionálnych testov pre túto aplikáciu sú:

- Pri testovaní komponentov sú testy navrhnuté na základe toho, ako by mal komponent vypočítať zložený úrok.
- Pri integračnom testovaní komponentov sú testy navrhnuté na základe toho, ako sa informácie o účte získané z užívateľského rozhrania predávajú biznis logike.
- Pri systémovom testovaní sú testy navrhnuté na základe toho, ako môžu držitelia účtu požiadať o nejakú formu úveru k svojmu bežnému účtu.
- Pri testovaní systémovej integrácie sú testy navrhnuté na základe toho, ako systém používa externú mikroslužbu na overenie úverového hodnotenia majiteľa účtu.
- Pre akceptačné testovanie sa testy navrhnutú na základe toho, ako bankár schvaľuje alebo zamieta žiadosti o úver.

Príklady nefunkcionálnych testov sú:

- Pre testovanie komponentov sa výkonnostné testy navrhujú tak, aby dochádzalo k vyhodnoteniu počtu cyklov procesoru potrebných k vykonaniu zložitého výpočtu celkového úroku.
- Pri integračnom testovaní komponentov sa bezpečnostné testy navrhujú tak, aby dochádzalo k vyhodnoteniu zraniteľnosti pretečením vyrovnávacej pamäte kvôli dátam predávaným z užívateľského rozhrania biznis logike.
- Pri systémovom testovaní sú testy prenositeľnosti navrhnuté na preskúšanie, či prezentačná vrstva funguje na všetkých podporovaných prehliadačoch a mobilných zariadeniach.
- Pri testovaní systémovej integrácie sú testy spoľahlivosti navrhnuté tak, aby vyhodnotili robustnosť systému v prípade nedostupnosti mikroslužby úverového hodnotenia.
- Pre akceptačné testovanie sú testy použiteľnosti navrhnuté tak, aby vyhodnotili prístupnosť rozhrania používaného bankárom pre spracovanie úverov klientov so zdravotným postihnutím.

Príklady testov bielej skrinky sú:

- Pri testovaní komponentov sú testy navrhnuté tak, aby dosiahli úplné pokrytie príkazov a rozhodnutí (viď kapitola 4.3) pre všetky komponenty, ktoré vykonávajú finančné výpočty.
- Pri integračnom testovaní komponentov sú testy navrhnuté tak, aby preskúmali spôsob, akým každá obrazovka prehliadača predáva údaje nasledujúcej obrazovke a biznis logike.
- Pri systémovom testovaní sú testy navrhované tak, aby pokrývali rôzne a možné postupnosti stránok, pri spracovaní žiadosti o úver.
- Pri testovaní systémovej integrácie sú testy navrhnuté tak, aby preskúmali všetky možné typy dotazov zasielaných mikroslužbe úverového hodnotenia.
- Pri akceptačnom testovaní sú testy navrhnuté tak, aby pokryli všetky podporované štruktúry súborov finančných údajov a rozsahy hodnôt pre prevody medzi bankami.

Na záver sú uvedené príklady testov súvisiacich so zmenami:

- Pri testovaní komponentov sú automatizované regresné testy implementované pre každý komponent a zaradené do frameworku priebežnej integrácie.
- Pri integračnom testovaní komponentov, sú testy navrhnuté tak, aby potvrdili opravy defektov na rozhraniach a obvykle sú spúšťané automaticky pri vložení kódu do repozitáru.

- Pri systémovom testovaní sa vykonajú všetky testy pre daný pracovný tok znova, a to pri zmene ktorejkoľvek obrazovky v danom pracovnom toku.
- Pri testovaní systémovej integrácie sa denne znovu vykonávajú testy interakcie aplikácie a mikroslužby úverového hodnotenia ako súčasť priebežného nasadzovania tejto mikroslužby.
- V prípade akceptačného testovania sa po oprave defektu zisteného akceptačným testovaním znovu vykonávajú testy, ktoré zlyhali.

Aj keď táto kapitola obsahuje príklady všetkých typov testov cez všetky úrovne, nie je potrebné mať zastúpený každý typ testu na každej úrovni pre každý softvér. Je však dôležité vykonávať vhodné typy testov na každej úrovni a najmä na tej najnižšej úrovni, kde sa daný typ testu môže vykonať.

2.4 Údržbové testovanie

Softvér a systémy musia byť po nasadení do produkčného prostredia naďalej udržiavané. Zmeny dodaného softvéru a systémov sú takmer nevyhnutné. Zmeny môžu predstavovať opravy defektov odhalených v produkčnom prostredí, pridanie novej funkčnosti alebo odstránenie či zmena už dodanej funkčnosti. Údržba je tiež potrebná na zachovanie alebo zlepšenie nefunkcionálnych charakteristík kvality komponentov alebo systému počas jeho životnosti, obzvlášť výkonnostná efektivita, spoľahlivosť, bezpečnosť a prenositeľnosť.

Ak sa v rámci údržby vykonajú akékoľvek zmeny, malo by sa vykonať údržbové testovanie a to na vyhodnotenie úspešnosti, s akou boli zmeny vykonané, a aby sa skontrolovali možné vedľajšie účinky (napr. regresia) v častiach systému, ktoré zostávajú nezmenené (čo je zvyčajne väčšina systému). Údržba môže zahŕňať plánované vydanie (release) alebo neplánované vydanie (hotfix).

Údržbové vydanie (maintenance release) môže vyžadovať údržbové testovanie na viacerých testovacích úrovniach s použitím rôznych typov testov a to v závislosti od rozsahu zmeny. Rozsah údržbového testovania závisí na:

- miere rizika zmeny, ako napríklad miera, do akej zmenená oblasť softvéru komunikuje s inými komponentmi alebo systémami
- veľkosti existujúceho systému
- rozsahu zmeny

2.4.1 Spúšťače údržby

Existuje niekoľko dôvodov prečo sa vykonáva údržba softvéru v podobe plánovaných a neplánovaných zmien.

Spúšťače údržby môžeme klasifikovať takto:

- Zmeny, ako sú plánované vylepšenia funkčnosti (napr. nové vydanie), opravy a zmeny prevádzkového prostredia (napr. plánovaná aktualizácia operačného systému alebo databázy), aktualizácie komerčného krabicového softvéru, a záplaty (patch) pre opravu defektov a zraniteľností
- Migrácia (napr. z jednej platformy na druhú), ktorá môže vyžadovať prevádzkové testy nového prostredia a zmeneného softvéru, alebo testy konverzie dát, pokiaľ budú migrované dáta z inej aplikácie do udržovaného systému
    - Vyradenie aplikácie z prevádzky na konci jej životnosti. Pri vyradení aplikácie alebo systému môže byť nutné otestovať migráciu alebo archiváciu dát, ktoré je nutné uchovať na dlhú dobu.
    - Testovanie postupov pre obnovenie a získavanie dát po dlhodobej archivácií.

- Regresné testovanie, aby sa zabezpečilo, že všetky funkčnosti, ktoré budú naďalej používané ešte stále fungujú.

Údržbové testovanie pre systémy Internetu vecí môže byť iniciované zavedením úplne nových alebo upravených „vecí“, ako sú hardvérové zariadenia alebo softvérová služba do celého systému. Údržbové testovanie takýchto systémov kladie osobitný dôraz na integračné testovanie na rôznych vrstvách (napr. na sieťovej a aplikačnej vrstve) a na testovanie aspektov bezpečnosti, najmä pre systémy, týkajúce sa osobných údajov.

2.4.2 Analýza dopadu na údržbu

Analýza dopadu hodnotí zmeny vykonané v rámci servisného vydania za účelom stanovenia zamýšľaných následkov i očakávaných a možných vedľajších účinkov zmeny a určenie oblastí systému, ktoré budú zmenou postihnuté. Analýza dopadu môže tiež pomôcť identifikovať dopad zmeny na existujúce testy. Vedľajšie účinky a postihnuté oblasti systému je potrebné regresne testovať, ideálne po aktualizácii všetkých existujúcich testov ktoré boli ovplyvnené zmenou.

Analýzu dopadu je možné vykonať pred samotnou zmenou, ako podporný prostriedok pre rozhodovanie, či by sa zmena mala vykonať, a to na základe potenciálnych dôsledkov v ostatných oblastiach systému.

Analýza dopadu môže byť komplikovaná, ak:

- špecifikácie (napr. biznis požiadavky, užívateľské scenáre, architektúra) sú zastarané alebo úplne chýbajú
- testovacie prípady nie sú zdokumentované alebo sú zastarané
- obojsmerná trasovateľnosť medzi testami a testovacou bázou nebola udržovaná
- podpora nástrojov je slabá alebo úplne chýba
- zúčastnené osoby nemajú znalosť biznis domény alebo systému
- nebola venovaná dostatočná pozornosť údržbe softvéru počas jeho vývoja

3 Statické testovanie

Kľúčové slová

ad-hoc revízia, revízia založená na kontrolnom zozname, dynamické testovanie, formálna revízia, neformálna revízia, inšpekcia, čítanie založené na perspektíve, revízia (preskúmanie), revízia založená na rolách, revízia založená na scenároch, statická analýza, statické testovanie, technická revízia (preskúmanie), predvedenie (walkthrough)

Študijné ciele pre kapitolu 3 – Statické testovanie:

3.1 Základy statického testovania

FL-3.1.1 (K1) Rozpoznať typy softvérového pracovného produktu, ktoré môžu byť preverené použitím rôznych techník statického testovania

FL-3.1.2 (K2) Príkladmi popísať význam statického testovania

FL-3.1.3 (K2) Vysvetliť rozdiel medzi statickými a dynamickými technikami, berúc do úvahy ciele, typy identifikovaných defektov a význam týchto techník v rámci životného cyklu vývoja softvéru

3.2 Proces revízie

FL-3.2.1 (K2) Zhrnúť činnosti procesu revízie pracovných produktov

FL-3.2.2 (K1) Rozpoznať rôzne role a zodpovednosti pri formálnej revízií

FL-3.2.3 (K2) Vysvetliť rozdiely medzi rôznymi typmi revízií: neformálna revízia, predvedenie, technická revízia a inšpekcia

FL-3.2.4 (K3) Použiť techniku revízie pracovného produktu s cieľom nájsť defekty

FL-3.2.5 (K2) Vysvetliť faktory, ktoré prispievajú k úspešnej revízií

3.1 Základy statického testovania

Na rozdiel od dynamického testovania, ktoré vyžaduje spustenie testovaného softvéru sa statické testovanie opiera o manuálne skúmanie pracovných produktov (t. j. revízie) alebo hodnotenie kódu alebo iných pracovných produktov pomocou nástrojov (t.j. statická analýza). Obidva typy statického testovania posudzujú daný pracovný produkt alebo kód bez toho, aby došlo k jeho spusteniu.

Statická analýza je dôležitá pre bezpečnostne kritické počítačové systémy (napr. softvér pre letecký, zdravotnícky alebo jadrový priemysel), ale stala sa dôležitou a bežnou súčasťou testovania aj v iných oblastiach. Napríklad statická analýza je dôležitou súčasťou testovania bezpečnosti. Statická analýza je tiež často súčasťou automatizovaných systémov na zostavovanie a dodávanie softvéru, napríklad ako systémy na priebežné dodávanie a priebežné nasadzovanie v agilnom vývoji.

3.1.1 Pracovné produkty, ktoré možno preveriť statickým testovaním

Takmer akýkoľvek pracovný produkt možno preveriť pomocou statického testovania (revízií a/alebo statických analýz), ako napríklad:

- špecifikácie vrátane biznis, funkcionálnych a bezpečnostných požiadaviek
- epicy, užívateľské scenáre a akceptačné kritériá
- špecifikácie architektúry a návrhu
- kód
- testvér, vrátane plánu testovania, testovacích prípadov, testovacích procedúr a automatizovaných testovacích skriptov
- užívateľské príručky
- webové stránky
- zmluvy, projektové plány, harmonogramy a plánovanie rozpočtu
- nastavenie konfigurácie a infraštruktúry
- modely, ako sú diagramy činností, ktoré môžu byť použité pre testovanie založené na modeloch (viď učebné osnovy *ISTQB-CTFL-MBT a Kramer 2016*).

Revízie môžu byť použité na akýkoľvek pracovný produkt, ktorý účastníci procesu revízie vedia prečítať a porozumieť mu. Statická analýza môže byť účinne aplikovaná na akýkoľvek pracovný produkt s formálnou štruktúrou (typicky kód alebo modely), pre ktorý existuje vhodný statický analytický nástroj. Statickú analýzu je dokonca možné aplikovať pomocou nástrojov, ktoré vyhodnocujú pracovné produkty napísané v prirodzenom jazyku, ako sú požiadavky (napr. kontrola pravopisu, gramatiky a čitateľnosti).

3.1.2 Výhody statického testovania

Techniky statického testovania poskytujú rôzne výhody. Pri aplikovaní na začiatku životného cyklu vývoja softvéru, statické testovanie umožňuje včasné zistenie defektov pred tým, ako sa vykoná dynamické testovanie (napr. pri revízii požiadaviek alebo návrhu, upresnení backlogu atď.). Obecne platí, že je oveľa lacnejšie odstrániť defekty nájdené skôr, ako defekty nájdené neskôr v životnom cykle. A to najmä pri porovnaní s defektmi nájdenými až po nasadení softvéru a jeho aktívnom používaní. Používanie techniky statického testovania pre nachádzanie a následnú rýchlu opravu defektov, je pre organizáciu takmer vždy oveľa lacnejšie, ako po použití dynamického testovania. Najmä pri zvážení ďalších nákladov spojených s aktualizáciou iných pracovných produktov a vykonaním konfirmačných a regresných testov.

Medzi ďalšie výhody statického testovania môže patriť:

- efektívnejšia detekcia defektov a ich možná oprava pred vykonaním dynamického testovania

- identifikácia defektov, ktoré nie sú ľahko odhaliteľné dynamickým testovaním
- prevencia vzniku defektov v návrhu alebo pri kódovaní a to odhalením nezrovnalostí, nejednoznačností, nepresností a redundancií v požiadavkách
- zvýšenie produktivity vývoja (napr. z dôvodu lepšieho návrhu alebo lepšie udržiavateľným kódom)
- skrátenie času a nákladov na vývoj
- skrátenie času a nákladov na testovanie
- zníženie celkových nákladov na kvalitu počas celej životnosti softvéru v dôsledku menšieho počtu zlyhaní v rámci životného cyklu alebo po dodaní do prevádzky
- zlepšenie komunikácie medzi členmi tímu v prípade ich účasti na revíziách

3.1.3 Rozdiely medzi statickým a dynamickým testovaním

Statické testovanie a dynamické testovanie môžu mať rovnaké ciele (viď kapitola 1.1.1), ako je posúdenie kvality pracovných produktov a včasná identifikácia defektov. Statické a dynamické testovanie sa navzájom dopĺňajú, tým že nachádzajú odlišné typy defektov.

Jedným z hlavných rozdielov je, že statické testovanie nachádza defekty priamo v pracovných produktoch, kým dynamické testovanie identifikuje zlyhania spôsobené defektmi pri spustení softvéru. Defekt môže byť prítomný v pracovnom produkte veľmi dlhú dobu bez toho, aby spôsobil zlyhanie. Vetva kde sa defekt nachádza môže byť vykonávaná zriedkavo alebo byť ťažko dosiahnuteľná. Takže vytvoriť a spustiť dynamický test, ktorý tento defekt objaví nebude jednoduché. Statické testovanie môže nájsť takýto defekt s oveľa menším úsilím.

Ďalším rozdielom je, že statické testovanie možno použiť na zlepšenie konzistencie a vnútornej kvality pracovných produktov, zatiaľ čo dynamické testovanie sa zvyčajne zameriava na externe pozorovateľné správanie.

V porovnaní s dynamickým testovaním, medzi typické defekty, ktoré sú jednoduchšie a lacnejšie na odhalenie a opravu prostredníctvom statického testovania, patria:

- defekty v požiadavkách (napr. nezrovnalosti, nejednoznačnosti, rozpory, opomenutia, nepresnosti a redundancie)
- defekty v návrhu (napr. neefektívne algoritmy alebo databázové štruktúry, vysoká duplicita, nízka súdržnosť)
- defekty pri kódovaní (napr. premenné s nedefinovanou hodnotou, deklarované, ale nepoužité premenné, nedosiahnuteľný kód, duplicitný kód)
- odchýlky od noriem (napr. nedostatočné dodržiavanie noriem pre kódovanie)
- nesprávna špecifikácia rozhraní (napr. použitie rôznych typov jednotiek volajúcim a volaným systémom)
- bezpečnostné zraniteľnosti (napr. náchylnosť na pretečenie vyrovnávacej pamäte)
- nedostatky alebo nepresnosti v trasovateľnosti alebo pokrytí testovacej báze (napr. chýbajúce testy pre niektoré akceptačné kritérium)

Okrem toho, väčšinu defektov v udržateľnosti softvéru, možno nájsť len statickým testovaním (napr. nesprávnu modularizáciu, nízku znovupoužiteľnosť komponentov, ťažko analyzovateľný a upravovateľný kód bez toho, aby bol zavedený nový defekt).

3.2 Proces revízie

Revízie sa od seba líšia stupňom formálnosti od neformálnych až po formálne. Neformálne revízie sa vyznačujú tým, že nie je dodržiavaný žiaden proces a nemajú formálne zdokumentované výstupy.

Formálne revízie sú charakterizované účasťou tímu, zdokumentovanými výsledkami revízie a zdokumentovanými postupmi vykonávania revízie. Formalita procesu revízie súvisí s faktormi, ako je model životného cyklu vývoja softvéru, zrelosť vývojového procesu, zložitosť revidovaných pracovných produktov, akékoľvek zákonné alebo regulatórne požiadavky a/alebo potreba zachovania auditných stôp.

Zameranie revízie závisí na dohodnutých cieľoch (napr. zistenie defektov, pochopenie pracovného produktu, vzdelávanie účastníkov revízie, ako sú testeri a noví členovia tímu, alebo diskusie a rozhodovanie konsenzom).

Podrobnejší popis procesu revízie vrátane pracovných produktov, rolí a techník revízie viď norma *ISO (ISO/IEC 20246)*.

3.2.1 Proces revízie pracovných produktov

Medzi hlavné činnosti procesu revízie partia:

Plánovanie

- definovanie rozsahu revízie zahŕňajúci jej účel, dokumenty alebo časti dokumentov na revíziu, a kvalitatívne charakteristiky na vyhodnotenie
- odhad prácnosti a harmonogram
- identifikácia charakteristík revízie, ako je typ revízie, role, činnosti a kontrolné zoznamy
- výber ľudí k účasti na revízii a prideľovanie rolí
- definovanie vstupných a výstupných kritérií pre formálnejšie typy revízie (napr. inšpekcie)
- kontrola splnenia vstupných kritérií (pre formálnejšie typy revízie)

Zahájenie revízie

- distribúcia pracovných produktov (fyzicky alebo elektronicky) a ďalších materiálov, ako sú formuláre pre zaznamenanie problémov, kontrolné zoznamy a súvisiace pracovné produkty
- vysvetlenie rozsahu, cieľov, procesov, rolí a pracovných produktov účastníkom revízie
- zodpovedanie otázok od účastníkov revízie

Individuálna revízia (t. j. Individuálna príprava)

- revízia celého pracovného produktu alebo jeho časti
- zaznamenanie možných defektov, odporúčaní a otázok

Komunikácia a analýza problému

- komunikovanie identifikovaných potenciálnych defektov (napr. na revíznej schôdzi)
- analýza potenciálnych defektov, priraďovanie vlastníka a stavu k defektom
- vyhodnotenie a zdokumentovanie kvalitatívnych charakteristík
- vyhodnotenie výsledkov revízie vzhľadom na výstupné kritériá s cieľom rozhodnúť o jej výsledku (zamietnutie, nutnosť veľkej zmeny, akceptovanie, prípadne akceptovanie s menšími zmenami)

Opravy a podávanie správ

- vytváranie správ o defektoch pre tie zistenia, ktoré vyžadujú zmeny v pracovnom produkte
- opravy nájdených defektov (zvyčajne vykonávané autorom) v revidovanom pracovnom produkte
- komunikovanie defektov príslušnej osobe alebo tímu (ak sa defekt nachádza v pracovnom produkte súvisiacom s revidovaným produktom)
- zaznamenanie aktualizovaných stavov defektov (vo formálnej revízii), vrátane prípadného odsúhlasenia autora komentáru

- zber metrík (pre formálnejšie typy revízie)
- kontrola splnenia výstupných kritérií (pre formálnejšie typy revízie)
- akceptácia pracovného produktu pri dosiahnutí výstupných kritérií

Výsledky revízie pracovného produktu sa líšia v závislosti od typu a úrovne formálnosti, ako je popísané v kapitole 3.2.3.

3.2.2 Role a zodpovednosti pri formálnej revízii

Typická formálna revízia bude obsahovať nasledujúce role:

Autor

- vytvorí revidovaný pracovný produkt
- opravuje defekty v revidovanom pracovnom produkte (pokiaľ je to potrebné)

Manažér (zástupca manažmentu)

- zodpovedá za plánovanie revízie
- rozhoduje o vykonaní revízie
- priraďuje ľudí, rozpočet a čas
- priebežne sleduje efektivitu nákladov
- rozhoduje o ďalšom postupe v prípade nedostatočných výsledkov

Moderátor

- zabezpečuje efektívny priebeh revíznych schôdzí (pokiaľ sa konajú)
- sprostredkováva vyjednávanie medzi rôznymi pohľadmi (pokiaľ je to potrebné)
- je často osobou, na ktorej závisí úspech revízie

Vedúci revízie

- preberá celkovú zodpovednosť za revíziu
- rozhoduje o tom, kto bude zapojený do revízie a organizuje, kedy a kde sa bude revízia konať

Revidujúci

- môže sa jednať o odborníkov na danú oblasť, osoby pracujúce na projekte, osoby zainteresované v danom pracovnom produkte a/alebo osoby s konkrétnym technickým alebo biznis pozadím.
- identifikuje potenciálne defekty v posudzovanom pracovnom produkte
- môže reprezentovať rôzne pohľady (napr. testera, vývojára, užívateľa, operátora, biznis analytika, experta na použiteľnosť, atď.)

Zapisovateľ

- zhromažďuje potenciálne defekty zistené počas individuálnej revízie
- zaznamenáva nové potenciálne defekty, otvorené body a rozhodnutia z revíznych schôdzí (pokiaľ sa konajú)

U niektorých typov revízie môže jedna osoba zastupovať viac ako jednu rolu a činnosti spojené s každou rolou sa môžu líšiť aj na základe typu revízie. Okrem toho, s príchodom nástrojov na podporu procesu revízie, najmä zaznamenávanie defektov, otvorených bodov a rozhodnutí, rola zapisovateľa už nebýva potrebná.

V revíziách sú možné aj ďalšie detailnejšie role, viď norma *ISO (ISO/IEC 20246)*.

3.2.3 Typy revízie

Aj keď sa dá použiť proces revízie na rôzne účely, jedným z hlavných cieľov je odhaliť defekty. Všetky typy revízie môžu pomôcť pri detekcii defektu. Vybraný typ revízie by mimo iných kritérií výberu mal vychádzať z potrieb projektu, dostupných zdrojov, typu produktu, identifikovaných rizík, alebo sféry biznisu a firemnej kultúry spoločnosti.

Jeden pracovný produkt môže byť predmetom viac ako jedného typu revízie. Ak sa používa viac ako jeden typ revízie, ich poradie sa môže líšiť. Napríklad, neformálna revízia sa môže vykonať pred technickou revíziou, aby bolo zabezpečené, že pracovný produkt je pripravený na technickú revíziu.

Typy vyššie popísaných revízií, možno vykonať ako vzájomnú revíziu (peer review), t. j. sú vykonané kolegami na podobnej organizačnej úrovni.

Typy odhalených defektov počas revízie sa líšia v závislosti od pracovných produktov, ktoré sú predmetom revízie. Príklady defektov, ktoré možno nájsť na základe revízie v rôznych pracovných produktoch sú uvedené v kapitole 3.1.3. Viac informácií o formálnych inšpekciách môžete nájsť v *Gilb 1993*.

Revízie môžu byť klasifikované podľa rôznych atribútov. Nasledujúci zoznam obsahuje štyri najbežnejšie typy revízií a ich súvisiace atribúty.

Neformálna revízia (napr. revízia kolegom, párovanie, párová revízia)

- Hlavný účel: detekcia potenciálnych defektov
- Ďalšie možné účely: tvorba nových nápadov alebo riešení, rýchle riešenie menších problémov
- Nie je založená na formálnom (zdokumentovanom) procese
- Nemusí zahŕňať revíznu schôdzu
- Môže byť vykonaná kolegom autora (buddy check) alebo viacerými osobami
- Výsledky môžu byť zdokumentované
- Užitočnosť je rôzna a závislá od revidujúcich
- Použitie kontrolných zoznamov je nepovinné
- Veľmi často je používaná v agilnom vývoji

Predvedenie

- Hlavné účely: detekcia potenciálnych defektov, vylepšenie softvérového produktu, zváženie alternatívnych implementácií, vyhodnotenie zhody s normami a špecifikáciami
- Ďalšie možné účely: výmena nápadov ohľadom použitia techník alebo štýlov spracovania, školenie účastníkov, dosiahnutie konsenzu
- Individuálna príprava pred revíznou schôdzou je nepovinná
- Revízna schôdza je zvyčajne vedená autorom pracovného produktu
- Rola zapisovateľa musí byť zastúpená
- Použitie kontrolných zoznamov je nepovinné
- Môže mať podobu scenárov, spustenia na nečisto (dry run), alebo simulácie
- Môžu byť vytvorené záznamy o potenciálnych defektoch a revízne správy
- V praxi sa môže líšiť od pomerne neformálneho až po veľmi formálne prevedenie

Technická revízia

- Hlavné účely: získanie konsenzu, detekcia potenciálnych defektov

- Ďalšie možné účely: hodnotenie kvality a budovanie dôvery v pracovný produkt, tvorba nových nápadov, motivácia autorov a umožnenie im zlepšiť budúce pracovné produkty, zváženie alternatívnych implementácií
- Revidujúci by mali byť rovnako technicky zdatní kolegovia autora a odborníci v rovnakých alebo iných technických disciplínach
- Individuálna príprava pred revíznou schôdzou je povinná
- Revízna schôdza je voliteľná, v ideálnom prípade však vedená vyškoleným moderátorom (typicky nie autorom)
- Rola zapisovateľa musí byt zastúpená, v ideálnom prípade nie autorom
- Použitie kontrolných zoznamov je nepovinné
- Môžu byť vytvorené záznamy o potenciálnych defektoch a revízne správy

Inšpekcia

- Hlavné účely: detekcia potenciálnych defektov, hodnotenie kvality a budovanie dôvery v pracovný produkt, predchádzanie podobným chybám v budúcnosti prostredníctvom vzdelávania autora a analýzy koreňových príčin
- Ďalšie možné účely: motivácia autorov a umožnenie im zlepšiť budúce pracovné produkty a proces vývoja softvéru, dosiahnuť konsenzus
- Je dodržiavaný definovaný proces s formálne zdokumentovanými výstupmi na základe pravidiel a kontrolných zoznamov
- Využíva jasne definované role (viď kapitola 3.2.2), ktoré sú povinné a môžu zahŕňať špecializovaného čitateľa (ktorý počas schôdze číta pracovný produkt nahlas a často parafrázuje pomocou vlastných slov)
- Individuálna príprava pred revíznou schôdzou je povinná
- Revidujúci by mali byť kolegovia autora a odborníci v technických disciplínach, ktoré sú relevantné pre pracovný produkt
- Sú špecifikované vstupné a výstupné kritériá
- Rola zapisovateľa musí byť zastúpená
- Revízna schôdza je vedená vyškoleným moderátorom (nie autorom)
- Autor nemôže vystupovať ako vedúci revízie, čitateľ, alebo zapisovateľ
- Môžu byť vytvorené záznamy o potenciálnych defektoch a revízne správy
- Metriky sa zhromažďujú a používajú na zlepšenie celého procesu vývoja softvéru vrátane procesu inšpekcie

3.2.4 Použitie techník revízie

Existuje celá rada techník revízie, ktoré sa dajú použiť pri činnostiach konkrétnej revízie (napr. pri individuálnej príprave) s cieľom odhaliť defekty. Tieto techniky môžu byť použité v rámci všetkých vyššie uvedených typov revízií. Efektívnosť techník sa môže líšiť v závislosti od typu použitej revízie. Príklady rozdielnych techník individuálnej revízie pre rôzne typy revízie sú uvedené nižšie.

Ad hoc

V ad hoc revízií, sú revidujúcim k dispozícii minimálne, prípade žiadne pokyny o spôsobe vykonania revízie. Revidujúci často čítajú pracovný produkt sekvenčne a identifikujú a dokumentujú problémy, keď na nich narazia. Ad hoc revízia je bežne používaná technika, ktorá vyžaduje malú prípravu. Táto technika je vysoko závislá na zručnosti revidujúcich a môže viesť k duplicitám v nahlásených problémoch od rôznych revidujúcich.

Revízia založená na kontrolnom zozname

Revízia založená na kontrolnom zozname je systematická technika, pri ktorej revidujúci zisťujú problémy pomocou kontrolných zoznamov, ktoré sú distribuované na začiatku revízie (napr. moderátorom). Kontrolný zoznam pre revíziu obsahuje súbor otázok mieriacich na potenciálne defekty je tvorený na základe skúseností. Kontrolné zoznamy by mali byť špecifické pre daný typ revidovaného pracovného produktu a mali by sa pravidelne udržiavať s cieľom pokryť problémy, ktoré boli vynechané v predchádzajúcich revíziách. Hlavnou výhodou techniky založenej na kontrolných zoznamoch je systematické pokrytie typických typov defektov. Pri individuálnych revíziách je potrebné dbať na to, aby sa nespracovávali len otázky z kontrolného zoznamu, ale aj hľadali defekty nepokryté kontrolným zoznamom.

Scenáre a spustenia na nečisto

Pri revízií založenej na scenári sú revidujúcim poskytnuté štruktúrované pravidlá na čítanie pracovného produktu. Revízia založená na scenároch, podporuje revidujúcich pri spustení pracovného produktu na nečisto na základe jeho predpokladaného využitia (ak je pracovný produkt zdokumentovaný vo vhodnom formáte, ako sú napríklad prípady použitia). Tieto scenáre poskytujú revidujúcim lepšie pokyny na identifikovanie konkrétnych typov defektov než jednoduché otázky kontrolného zoznamu. Rovnako ako pri revíziách na základe kontrolného zoznamu, by sa revidujúci nemal obmedzovať len na zdokumentované scenáre z dôvodu rizika vynechania iných typov defektov (napr. chýbajúce vlastnosti).

Revízia založená na perspektíve

Pri technike čítania založeného na perspektíve (perspective-based reading), podobne ako pri revízií založenej na rolách sa revidujúci pri individuálnych revíziách pozerajú na revidovaný produkt očami rôznych zainteresovaných strán. Napríklad pohľadom koncového používateľa, marketingového tímu, špecialistu na návrh, testera alebo zástupcu prevádzky. Použitie pohľadov rôznych zainteresovaných strán vedie k väčšej hĺbke v rámci individuálnej revízie s nižšou duplicitou problémov naprieč revidujúcimi.

Okrem toho, čítanie založené na perspektíve vyžaduje, aby sa revidujúci pokúsil využiť revidovaný pracovný produkt na vytvorenie nového produktu, odvodený z toho pôvodného. Napríklad, tester by sa pokúsil vytvoriť návrh akceptačných testov počas čítania špecifikácie požiadaviek, aby zistil, či boli zahrnuté všetky potrebné informácie. Okrem toho sa pri čítaní založenom na perspektíve očakáva použitie kontrolných zoznamov.

Empirické štúdie ukázali, že čítanie založené na perspektíve je najúčinnejšou všeobecnou technikou na revíziu požiadaviek a technických pracovných produktov. Kľúčovým faktorom úspechu je primerané zohľadňovanie rôznych pohľadov zainteresovaných strán na základe rizík. Pre viac detailov o čítaní založenom na perspektíve viď *Shul 2000* a pre viacej detailov o účinnosti rôznych techník revízie viď *Sauer 2000*.

Revízia založená na rolách

Revízia založená na rolách je technika, pri ktorej revidujúci hodnotia pracovný produkt z pohľadu jednotlivých zainteresovaných rolí. Typické role zahŕňajú špecifické typy koncových používateľov (skúsený, neskúsený, senior, dieťa, atď.) a konkrétne role v organizácií (správca užívateľov, správca systému, tester výkonnosti, atď.). Platia obdobné princípy ako u revízie založenej na perspektíve lebo role sú podobné.

3.2.5 Faktory úspechu pri revíziách

Pre úspešnú revíziu je treba zvážiť vhodný typ revízie a techniky, ktoré budú použité. Okrem toho existuje rada ďalších faktorov, ktoré budú mať vplyv na výsledok revízie.

Medzi faktory úspechu z hľadiska organizácie patrí:

- Každá revízia má jasné ciele, ktoré sú definované počas plánovania revízie a používajú sa ako merateľné výstupné kritériá
- Používané typy revízií sú vhodné pre dosiahnutie cieľov a zároveň musia byť vhodné pre typ a úroveň softvérových pracovných produktov a účastníkov revízie
- Akékoľvek použité techniky revízie, ako revízia založená na kontrolných zoznamoch alebo rolách, sú vhodné pre účinnú identifikáciu defektov v revidovanom pracovnom produkte
- Využívané kontrolné zoznamy sa týkajú hlavných rizík a sú aktuálne
- Rozsiahle dokumenty sú vytvárané a revidované po malých častiach, takže kvalita je zaisťovaná tým, že autor dostáva spätnú väzbu včas a často
- Účastníci majú dostatok času na prípravu
- Revízie sú naplánované s dostatočným predstihom
- Manažment podporuje proces revízie (napr. alokovaním primeraného času na revízne činnosti v projektových plánoch)
- Revízie sú súčasťou firemných politík kvality a/alebo testovania

Medzi faktory úspechu z hľadiska ľudí patrí:

- Pre dosiahnutie cieľov revízie, sú zapojení tí správni ľudia, ako napríklad ľudia s rôznymi zručnosťami, perspektívami pohľadu, alebo tí, ktorí môžu používať revidovaný dokument ako pracovný vstup.
- Testeri sú vnímaní ako hodnotní revidujúci, ktorí prispievajú k revízii a zároveň študujú pracovný produkt, čo im umožňuje pripraviť efektívnejšie testy v predstihu
- Účastníci venujú dostatok času a pozornosti detailom
- Revízie sa vykonávajú po malých častiach, takže revidujúci nestrácajú koncentráciu počas individuálnej revízie a/alebo revíznej schôdze (pokiaľ sa konajú)
- Zistené defekty sú rešpektované, oceňované a objektívne spracované
- Schôdza je vedená správne, takže ju účastníci považujú za užitočne využitý čas
- Revízia prebieha v atmosfére dôvery; výsledok sa nepoužíva na hodnotenie účastníkov
- Účastníci sa dokážu vyhýbať nevhodnej reči tela a správaniu, ktoré by mohlo naznačovať nudu, podráždenie, alebo nepriateľstvo voči ostatným účastníkom
- Účastníkom je poskytnuté vhodné odborné školenie, najmä pri formálnejších typoch revízie ako je inšpekcia
- Organizácia podporuje kultúru učenia a zlepšovania procesu

Pre viac informácii o úspechu revízií viď *Gilb 1993*, *Wiegers 2002*, a *Van Veenendaal 2004*.

4 Techniky testovania – 330 minút

Kľúčové slová

technika testovania čiernej skrinky, analýza hraničných hodnôt, revízia založená na kontrolných zoznamoch, pokrytie, pokrytie rozhodnutí, testovanie podľa rozhodovacej tabuľky, odhadovanie chýb, rozdelenie tried ekvivalencie, technika testovania založená na skúsenostiach, prieskumné testovanie (exploratory testing), testovanie prechodov stavov, pokrytie príkazov, technika testovania, testovanie prípadov použitia, technika testovania bielej skrinky

Študijné ciele pre kapitolu 4 – Techniky testovania:

4.1 Kategórie techník testovania

FL-4.1.1 (K2) Vysvetliť charakteristiky, spoločné znaky a rozdiely medzi technikami testovania čiernej skrinky, technikami testovania bielej skrinky a technikami testovania založených na skúsenostiach

4.2 Techniky testovania čiernej skrinky

FL-4.2.1 (K3) Použiť techniku rozdelenia tried ekvivalencie na odvodenie testovacích prípadov z daných požiadaviek

FL-4.2.2 (K3) Použiť techniku analýzy hraničných hodnôt na odvodenie testovacích prípadov z daných požiadaviek

FL-4.2.3 (K3) Použiť techniku testovania podľa rozhodovacej tabuľky na odvodenie testovacích prípadov z daných požiadaviek

FL-4.2.4 (K3) Použiť techniku testovania prechodov stavov na odvodenie testovacích prípadov z daných požiadaviek

FL-4.2.5 (K2) Vysvetliť spôsob odvodenia testovacích prípadov z prípadov použitia

4.3 Techniky testovania bielej skrinky

FL-4.3.1 (K2) Vysvetliť techniku pokrytia príkazov

FL-4.3.2 (K2) Vysvetliť techniku pokrytia rozhodnutí

FL-4.3.3 (K2) Vysvetliť hodnotu pokrytia príkazov a pokrytia rozhodnutí

4.4 Techniky testovania založené na skúsenostiach

FL-4.4.1 (K2) Vysvetliť techniku odhadovania chýb

FL-4.4.2 (K2) Vysvetliť techniku prieskumného testovania

FL-4.4.3 (K2) Vysvetliť techniku testovania založeného na kontrolných zoznamoch

4.1 Kategórie techník testovania

Účelom techník testovania, vrátane tých, ktoré sú popísané v tejto kapitole, je pomôcť pri identifikácii testovacích podmienok, testovacích prípadov a testovacích dát.

Výber vhodných techník testovania, ktoré budú použité, závisí na rade faktorov, napr:

- zložitosti komponentu alebo systému
- regulatórnych noriem
- zákazníckych alebo zmluvných požiadaviek
- úrovne a typy rizík
- dostupnej dokumentácie
- znalostí a zručností testerov
- dostupných nástrojov
- času a rozpočtu
- modelu životného cyklu vývoja softvéru
- typu očakávaných defektov pre daný komponent alebo systém

Niektoré techniky sú lepšie použiteľné pre určité situácie a úrovne testov; zatiaľ čo iné sú použiteľné na všetkých úrovniach testov. Pri vytváraní testovacích prípadov, testeri vo všeobecnosti kombinujú techniky testovania na dosiahnutie najlepších výsledkov vzhľadom k prácnosti testovania.

Použitie techník testovania pri testovacej analýze, návrhu testov a implementácii testov môže byť od veľmi neformálneho (minimum alebo žiadna dokumentácia) až po veľmi formálne. Príslušná úroveň formality závisí na kontexte testovania vrátane zrelosti testovacích a vývojových procesov, časových obmedzení, bezpečnostných alebo regulatórnych požiadaviek, znalostí a zručností zúčastnených osôb a modeli životného cyklu vývoja softvéru.

4.1.1 Kategórie techník testovania a ich charakteristiky

V tejto učebnej osnove sú techniky testovania rozdelené na techniky testovania bielej skrinky, techniky testovania čiernej skrinky a techniky testovania založených na skúsenostiach.

Techniky testovania čiernej skrinky (tiež nazývané behaviorálne alebo techniky založené na správaní) sú založené na analýze príslušnej testovacej báze (napr. formálnej dokumentácie požiadaviek, špecifikácií, prípadov použitia, užívateľských scenárov alebo biznis procesov). Tieto techniky sú použiteľné pri funkcionálnom aj nefunkcionálnom testovaní. Techniky testovania čiernej skrinky sa sústreďujú na vstupy a výstupy testovaného objektu bez ohľadu na jeho vnútornú štruktúru.

Techniky testovania bielej skrinky (tiež nazývané štrukturálne techniky alebo techniky založené na štruktúre) sú založené na analýze architektúry, podrobnom návrhu, vnútornej štruktúre, alebo kóde testovaného objektu. Na rozdiel od techník testovania čiernej skrinky, techniky testovania bielej skrinky sa sústreďujú na štruktúru a spracovanie v rámci testovaného objektu.

Techniky testovania založené na skúsenostiach využívajú skúsenosti vývojárov, testerov a užívateľov navrhovať, implementovať a vykonávať testy. Tieto techniky sú často kombinované s technikami testovania čiernej alebo bielej skrinky.

Medzi bežné charakteristiky techník testovania čiernej skrinky patria:

- testovacie podmienky, testovacie prípady a testovacie dáta sú odvodené z testovacej bázy, ktorá môže zahŕňať softvérové požiadavky, špecifikácie, prípady použitia a užívateľské scenáre
- testovacie prípady sa môžu použiť na odhalenie nedostatkov medzi požiadavkami a implementáciou požiadaviek, alebo na odchýlky od požiadaviek
- pokrytie sa meria na základe otestovaných položiek testovacej bázy a techniky použitej v rámci testovacej bázy

Medzi bežné charakteristiky techník testovania bielej skrinky patria:

- testovacie podmienky, testovacie prípady a testovacie dáta sú odvodené z testovacej bázy, ktorá môže zahŕňať kód, softvérovú architektúru, detailný návrh alebo akýkoľvek iný zdroj informácií o štruktúre softvéru
- pokrytie sa meria na základe položiek otestovaných v rámci vybranej štruktúry (napr. kódu alebo rozhraní) a použitej techniky testovania

Medzi bežné charakteristiky techník testovania založených na skúsenostiach patria:

- testovacie podmienky, testovacie prípady a testovacie dáta sú odvodené z testovacej bázy, ktorá môže zahŕňať znalosti a skúsenosti testerov, vývojárov, používateľov a iných zainteresovaných strán

Tieto znalosti a skúsenosti zahŕňajú očakávané používanie softvéru, prostredia, pravdepodobné defekty a distribúciu týchto defektov.

Viac informácií o technikách testovania a im odpovedajúcim metrikám pokrytia viď norma *ISO/IEC/IEEE 29119-4*, *Craig 2002* a *Copeland 2004*.

4.2 Techniky testovania čiernej skrinky

4.2.1 Rozdelenie tried ekvivalencie

Rozdelenie tried ekvivalencie rozdeľuje dáta do tried (tiež známych ako triedy ekvivalencie) spôsobom, že všetky prvky v danej triede budú spracované rovnakým spôsobom (viď *Kaner 2013* a *Jorgensen 2014*). Triedy ekvivalencie existujú pre platné aj neplatné hodnoty.

- Platné hodnoty sú hodnoty, ktoré by mal komponent alebo systém akceptovať. Trieda ekvivalencie obsahujúca platné hodnoty sa nazýva "platná trieda ekvivalencie".
- Neplatné hodnoty sú hodnoty, ktoré by mal komponent alebo systém odmietnuť. Trieda ekvivalencie obsahujúca neplatné hodnoty sa nazýva "neplatná trieda ekvivalencie".
- Triedy ekvivalencie možno identifikovať pre akúkoľvek dátovú entitu súvisiacu s testovaným objektom vrátane vstupov, výstupov, interných hodnôt, časových hodnôt (napr. pred alebo po udalosti) a pre parametre rozhrania (napr. integrované komponenty testované počas integračného testovania).
- V prípade potreby je možné akúkoľvek triedu ekvivalencie rozdeliť na podtriedy.
- Každá hodnota musí patriť do jednej a práve jednej triedy ekvivalencie.
- Ak sa v testovacích prípadoch používajú neplatné triedy ekvivalencie, každá takáto trieda by mala byť testovaná jednotlivo, t. j. nie v kombinácii s inými neplatnými triedami ekvivalencie, aby sa zabezpečilo, že nedôjde k maskovaniu zlyhaní. Zlyhania môžu byť maskované, keď sa vyskytne niekoľko zlyhaní v rovnakom čase, ale len jedno z nich je viditeľné, čo znamená, že iné zlyhania ostanú nezistené.

Na dosiahnutie 100% pokrytia s touto technikou, testovacie prípady musia pokrývať všetky identifikované triedy (vrátane neplatných tried) pomocou minimálne jednej hodnoty z každej triedy. Pokrytie sa meria ako počet tried ekvivalencie otestovaných aspoň jednou hodnotou, vydelený celkovým počtom identifikovaných tried ekvivalencie. Zvyčajne sa vyjadruje ako percentuálny podiel. Rozdelenie tried ekvivalencie sa uplatňuje na všetkých testovacích úrovniach.

4.2.2 Analýza hraničných hodnôt

Analýza hraničných hodnôt (Boundary Value Analysis - BVA) je rozšírenie rozdelenia tried ekvivalencie. Môže sa však použiť iba v prípade tried so zoradenými číselnými alebo sekvenčnými dátami. Minimálne a maximálne hodnoty (alebo prvé a posledné hodnoty) triedy sú jej hraničné hodnoty (viď *Beizer 1990*).

Predpokladajme napríklad, že textové pole na vstupe akceptuje jednomiestnu kladnú číselnú hodnotu, ktorá sa zadáva pomocou číselnej klávesnice, takže nie je možné zadať hodnotu nečíselnú. Platný rozsah číslic je od 1 do 5, vrátane. Takže existujú tri triedy ekvivalencie: neplatné (príliš nízke), platné, neplatné (príliš vysoké). Pre platnú triedu ekvivalencie sú hraničné hodnoty 1 a 5. Pre neplatnú (príliš vysokú) triedu, je hraničná hodnota 6. Pre neplatnú (príliš nízku) triedu je len jedna hraničná hodnota a to 0, pretože ide o triedu s iba jedným členom.

Vo vyššie uvedenom príklade identifikujeme dve hraničné hodnoty pre každú hranicu. Hranica medzi neplatnou (príliš nízkou) a platnou triedou sú hraničné hodnoty 0 a 1. Hranica medzi platnou a neplatnou (príliš vysokou) triedou sú hraničné hodnoty 5 a 6. Niektoré varianty tejto techniky identifikujú tri hraničné hodnoty na hranicu: hodnoty pred hranicou, na hranici a tesne za hranicou. Pri použití trojbodových hraničných hodnôt pre predchádzajúci príklad, hraničné hodnoty pre dolnú hranicu sú 0, 1 a 2 a hraničné hodnoty pre hornú hranicu sú 4, 5 a 6 (viď *Jorgensen 2014*).

Pravdepodobnosť nesprávneho správania na hraniciach tried ekvivalencie je vyššia ako správanie v rámci triedy ekvivalencie. Je dôležité si uvedomiť, že ako špecifikované (predpokladané), tak implementované (skutočné) hranice môžu byť presunuté na pozície nad, alebo pod ich zamýšľanými pozíciami, môžu byť úplne vynechané alebo môžu byť doplnené o ďalšie nežiaduce hranice. Analýza hraničných hodnôt a ich testovanie odhalí takmer všetky takéto defekty tým, že núti softvér ukázať správanie z inej triedy než je tá, do ktorej by hraničná hodnota mala patriť.

Analýza hraničných hodnôt môže byť použitá na všetkých testovacích úrovniach. Táto technika sa všeobecne používa na testovanie požiadaviek s číselným oborom hodnôt (vrátane dátumov a časov). Pokrytie hraničných hodnôt pre triedy ekvivalencie sa meria ako počet testovaných hraničných hodnôt, vydelený celkovým počtom identifikovaných hraničných testovateľných hodnôt. Zvyčajne sa vyjadruje ako percentuálny podiel.

4.2.3 Testovanie podľa rozhodovacej tabuľky

Rozhodovacie tabuľky sú dobrým spôsobom, ako zaznamenať zložité biznis pravidlá, ktoré systém musí implementovať. Pri vytváraní rozhodovacích tabuliek, tester identifikuje podmienky (často vstupy) a výsledné akcie (často výstupy) systému. Tieto tvoria riadky tabuľky, zvyčajne s podmienkami na vrchu a akciami v dolnej časti tabuľky. Každý stĺpec zodpovedá rozhodovaciemu pravidlu, ktoré definuje jedinečnú kombináciu podmienok, ktoré vedú k výkonu akcie spojenej s týmto pravidlom. Hodnoty podmienok a akcií sa zvyčajne zobrazujú ako boolovské hodnoty (pravda alebo nepravda) alebo diskrétne hodnoty (napr. červená, zelená, modrá), ale môžu to byť aj čísla alebo rozsahy čísel. Tieto rôzne typy podmienok a akcií možno nájsť spoločne v jednej tabuľke.

Bežný zápis rozhodovacích tabuliek je nasledovný:

Pre podmienky:

- Y ako Yes znamená, že podmienka je pravdivá (tiež môže byť zobrazená použitím T ako True alebo 1)
- N ako No znamená, že podmienka je nepravdivá (tiež môže byť zobrazená použitím F ako False alebo 0)

- — znamená, že na hodnote podmienky nezáleží (tiež môže byť uvedená použitím N/A ako Not Available)Pre akcie:
- X znamená, že akcia by mala nastať (tiež môže byť zobrazená použitím Y, T alebo 1)
- Prázdna hodnota znamená, že akcia by nastať nemala (tiež môže byť zobrazená pomocou —, N, F alebo 0)

Úplná rozhodovacia tabuľka má dostatok stĺpcov (testovacích prípadov) na pokrytie každej kombinácie podmienok. Odstránením stĺpcov, ktoré neovplyňujú výsledok, môžeme značne zredukovať počet testovacích prípadov. Napríklad odtránením nedostiahnuteľných kombinácií podmienok. Pre viac informácií o redukovaní rozhodovacích tabuliek, viď *ISTQB-CTAL-TA*.

Obecnou normou minimálneho pokrytia pre testovanie podľa rozhodovacej tabuľky je mať aspoň jeden testovací prípad na každé rozhodovacie pravidlo v tabuľke. To zvyčajne znamená, že sú pokryté všetky kombinácie podmienok. Pokrytie sa meria ako počet rozhodovacích pravidiel otestovaných aspoň jedným testovacím prípadom, vydelený celkovým počtom rozhodovacích pravidiel. Zvyčajne sa vyjadruje ako percentuálny podiel.

Sila testovania podľa rozhodovacej tabuľky je v tom, že pomáha identifikovať všetky dôležité kombinácie podmienok, z ktorých by niektoré mohli byť prehliadnuté. Pomáha tiež pri hľadaní nedostatkov v požiadavkách. Môže sa aplikovať na všetky situácie, v ktorých správanie softvéru závisí od kombinácie podmienok na akejkoľvek testovacej úrovni.

4.2.4 Testovanie prechodov stavov

Komponenty alebo systémy môžu reagovať odlišne na udalosti v závislosti od aktuálnych podmienok alebo predošlých udalostí (napr. udalosti, ku ktorým došlo od inicializácie systému). Predošlé udalosti môžu byť vyjadrené pomocou stavov. Diagram prechodov stavov zobrazuje možné stavy softvéru, ako aj spôsob, akým softvér do nejakého stavu vstupuje, vystupuje z neho, a ako medzi týmito stavmi prechádza. Prechod medzi stavmi je inicializovaný udalosťou (napr. užívateľ vloží hodnotu do textového poľa). Výsledkom udalosti je prechod. Rovnaká udalosť môže viesť k dvom alebo viacerým rôznym prechodom z rovnakého stavu. Zmena stavu môže viesť k tomu, že vykoná nejakú akciu (napr. zobrazenie výsledku výpočtu alebo chybového hlásenia).

Tabuľka prechodov stavov zobrazuje všetky platné prechody a potenciálne neplatné prechody medzi stavmi, ako aj udalosti a výsledné akcie pre platné prechody. Diagramy prechodov stavov zvyčajne zobrazujú iba platné prechody a vynechávajú neplatné.

Testy môžu byť navrhnuté tak, aby pokryli typickú postupnosť stavov, všetky stavy, všetky prechody, konkrétne sekvencie prechodov, alebo otestovali neplatné prechody.

Testovanie prechodov stavov sa používa pre aplikácie založené na menu a často sa využíva v oblasti vstavaného softvéru (embedded software). Táto technika je vhodná aj pre modelovanie biznis scenárov s konkrétnymi stavmi alebo na testovanie navigácie po obrazovke. Pojem „stav“ je abstraktný – môže predstavovať niekoľko riadkov kódu alebo celý biznis proces.

Pokrytie sa obvykle meria ako počet identifikovaných a otestovaných stavov alebo prechodov, vydelený celkovým počtom identifikovaných stavov alebo prechodov v testovanom objekte. Zvyčajne sa vyjadruje ako percentuálny podiel. Pre viac informácií o kritériách pokrytia pre testovanie prechodov stavov, viď *ISTQB-CTAL-*.

4.2.5 Testovanie prípadov použitia

Testy môžu byť odvodené z prípadov použitia, ktoré reprezentujú špecifický spôsob navrhovania interakcií so softvérovými položkami, ktoré zahŕňajú požiadavky na funkcie softvéru. Prípady použitia sú späté s aktérmi (užívateľmi, externým hardvérom, komponentmi alebo systémami) a predmetom testovania (komponent alebo systém, na ktorý sa aplikuje prípad použitia).

Každý prípad použitia definuje určité správanie, ktoré môže predmet testovania vykonávať v spolupráci s jedným alebo viacerými aktérmi (*UML 2.5.1 2017*). Prípad použitia možno opísať interakciami a činnosťami, ako aj vstupnými a výstupnými podmienkami a v určitých prípadoch aj prirodzeným jazykom. Interakcie medzi aktérmi a predmetom testovania môžu viesť k zmenám stavu predmetu. Interakcie môžu byť vyjadrené graficky podľa pracovných tokov, diagramov činností alebo modelov biznis procesov.

Prípad použitia môže zahŕňať aj možné odchýlky od jeho základného správania, a to vrátane mimoriadneho správania (výnimka) a spracovania chýb (reakcia systému a obnovenie po programátorskej, aplikačnej alebo komunikačnej chybe, ktorá môže mať za následok napr. chybové hlásenie). Testy sú navrhnuté tak, aby vykonávali definované správanie (základné, mimoriadne alebo alternatívne a spracovanie chýb). Pokrytie možno merať ako počet otestovaného správania definovaného pomocou prípadu použitia, vydelené celkovým počtom správaní na základe prípadu použitia. Zvyčajne sa vyjadruje ako percentuálny podiel.

Pre viac informácií o kritériách pokrytia pre testovanie prípadov použitia, viď *ISTQB-CTAL-TA*.

4.3 Techniky testovania bielej skrinky

Testovanie bielej skrinky je založené na vnútornej štruktúre testovaného objektu. Techniky testovania bielej skrinky môžu byť použité na všetkých testovacích úrovniach, ale dve s kódom súvisiace techniky popísané v tejto časti sú najčastejšie používané na úrovni testovania komponentov. Existujú pokročilejšie techniky, ktoré sa používajú v niektorých bezpečnostne kritických prostrediach, mission-critical prostrediach, alebo prostrediach s vysokou integritou na dosiahnutie dokonalého pokrytia, avšak v tejto kapitole popísané nie sú. Pre viac informácií o týchto technikách, viď *ISTQB-CTAL-TTA*.

4.3.1 Testovanie a pokrytie príkazov

Testovanie príkazov preveruje potenciálne spustiteľné príkazy v kóde. Pokrytie sa meria ako počet príkazov vykonaných testami vydelený celkovým počtom vykonateľných príkazov v testovanom objekte. Zvyčajne sa vyjadruje ako percentuálny podiel.

4.3.2 Testovanie a pokrytie rozhodnutí

Testovanie rozhodnutí preveruje rozhodnutia v kóde a testuje kód, ktorý sa vykonáva na základe výsledkov rozhodnutia. Testovacie prípady preto vychádzajú z riadiacich tokov, ktoré sú definované rozhodovacími bodmi. Napr. pre príkaz IF je jeden tok pre pravdivý a jeden pre nepravdivý výsledok. Pre príkaz CASE by bolo nutné navrhnúť testovacie prípady pre všetky možné výsledky vrátane štandardnej (default) vetvy.

Pokrytie sa meria ako počet výsledkov rozhodnutia vykonaných testami vydelenými celkovým počtom výsledkov rozhodnutia v testovanom objekte. Zvyčajne sa vyjadruje ako percentuálny podiel.

4.3.3 Hodnota testovania príkazov a testovania rozhodnutí

V prípade dosiahnutia 100% pokrytia príkazov je zabezpečené, že všetky spustiteľné príkazy v kóde boli testované aspoň raz, ale nie je zabezpečené otestovanie všetkej logiky rozhodovania. Z dvoch techník testovania bielej skrinky vysvetlených v tejto učebnej osnove môže testovanie príkazov poskytnúť menšiu mieru pokrytia ako testovanie rozhodnutí.

V prípade dosiahnutia 100% pokrytia rozhodnutí je zaistené, že boli preverené všetky výsledky rozhodnutia (výsledky TRUE aj FALSE). To platí aj v prípade, že príkaz pre vetvu FALSE nie je explicitne definovaný (napr. neexistencia vetvy ELSE v príkaze IF). Pokrytie príkazov pomáha nájsť defekty v kóde, ktorý nebol vykonávaný inými testami. Pokrytie rozhodnutí pomáha nájsť defekty v kóde v prípadoch, kde by iné testy neoverili obe vetvy (TRUE aj FALSE) rozhodnutia.

Dosiahnutie 100% pokrytia rozhodnutí zaručuje 100% pokrytie príkazov (ale nie naopak).

4.4 Techniky testovania založené na skúsenostiach

Pri použití techník testovania založených na skúsenostiach, sú testovacie prípady odvodené na základe zručností a intuície testerov, ako aj na ich skúsenostiach s podobnými aplikáciami a technológiami. Tieto techniky môžu byť užitočné pri definícii testov, ktoré by nebolo ľahké identifikovať inými systematickými technikami. V závislosti na prístupe a skúsenostiach testera môžu tieto techniky dosiahnuť rôzny stupeň pokrytia a efektivity. V niektorých prípadoch posúdiť mieru pokrytia môže byť pomerne ťažké. Dokonca, pri použití týchto techník, môže byť miera pokrytia nemerateľná.

Bežne používané techniky založené na skúsenostiach sú popísané v nasledujúcich podkapitolách.

4.4.1 Odhadovanie chýb

Odhadovanie chýb je technika používaná na predvídanie výskytu chýb, defektov a zlyhaní založená na znalostiach testera, vrátane:

- ako aplikácia pracovala v minulosti
- aké typy chýb sa zvyčajne vyskytujú
- aké zlyhania sa vyskytli v iných aplikáciách

Metodický prístup k technike odhadovania chýb spočíva vo vytvorení zoznamu možných chýb, defektov a zlyhaní a navrhnutia testov, ktoré by mohli tieto zlyhania a defekty odhaliť. Zoznamy týchto chýb, defektov a zlyhaní môžu byť vytvorené na základe skúseností, dát o defektoch a zlyhaniach, alebo z obecných znalostí o tom, prečo softvér zlyháva.

4.4.2 Prieskumné testovanie

Prieskumné testovanie je založené na neformálnych (nie vopred definovaných) testoch, ktoré sa navrhujú, vykonávajú, zaznamenávajú a vyhodnocujú dynamicky počas vykonávania testov. Výsledky testov sa používajú na učenie a získavanie informácií o komponente alebo systéme a na vytvorenie testov pre oblasti, ktoré môžu vyžadovať väčší rozsah testovania.

Prieskumné testovanie môže byť vykonávané formou testovacieho sedenia za účelom vykonávať túto činnosť štruktúrovane. Počas testovacieho sedenia sa prieskumné testovanie vykonáva v rámci definovaného časového úseku, kedy tester používa testovaciu listinu obsahujúcu ciele testovania, ktorých dosiahnutie je cieľom tohto testovania. Tester môže v rámci testovacieho sedenia dokumentovať vykonávané kroky a zistenia.

Prieskumné testovanie je najužitočnejšie v prípadoch s nedostatočnou alebo chýbajúcou špecifikáciou alebo v prípadoch výrazného časového tlaku na testovanie. Prieskumné testovanie je užitočné aj ako doplnok viac formálnych testovacích techník.

Prieskumné testovanie je silne spojené s reaktívnymi testovacími stratégiami (viď kapitola 5.2.2). Prieskumné testovanie môže zahŕňať použitie techník testovania čiernej skrinky, bielej skrinky a techník založených na skúsenostiach.

4.4.3 Revízia založená na kontrolných zoznamoch

Pri testovaní na základe kontrolného zoznamu testeri navrhujú, implementujú a vykonajú testy na pokrytie testovacích podmienok, ktoré sú obsahom kontrolného zoznamu. Testeri v rámci analýzy tvoria nový kontrolný zoznam, alebo rozširujú existujúci kontrolný zoznam. Môžu však použiť už aj existujúci kontrolný zoznam bez zmeny. Tieto kontrolné zoznamy môžu byť vytvorené na základe skúseností, znalosti o tom, čo je dôležité pre užívateľa, alebo pochopenia toho, prečo a ako zlyháva softvér.

Kontrolné zoznamy môžu byť vytvorené na podporu rôznych typov testov, vrátane funkcionálneho a nefunkcionálneho testovania. Pri absencii podrobných testovacích prípadov môže testovanie na základe kontrolného zoznamu poskytnúť návod na testovanie a zaistiť určitý stupeň konzistentnosti testovania. Keďže sa jedná o všeobecné zoznamy (high-level) je pravdepodobné, že sa v testovaní môže objaviť určitá variabilita, čo môže prispieť k väčšiemu pokrytiu, ale tiež k menšej miere opakovateľnosti testu.

5 Manažment testovania – 225 minút

Kľúčové slová

konfiguračný manažment, manažment defektov, správa o defekte, vstupné kritériá, výstupné kritériá, produktové riziko, projektové riziko, riziko, úroveň rizika, testovanie založené na rizikách, prístup k testovaniu, riadenie testovania, odhad testovania, manažér testovania, monitorovanie testovania, plán testovania, plánovanie testovania, správa o pokroku v testovaní, testovacia stratégia, súhrnná správa z testovania, tester

Študijné ciele pre kapitolu 5 – Manažment testovania:

5.1 Organizácia testovania

FL-5.1.1 (K2) Vysvetliť výhody a nevýhody nezávislého testovania

FL-5.1.2 (K1) Identifikovať úlohy pre manažéra testovania a testera

5.2 Organizácia testovania

FL-5.2.1 (K2) Zhrnúť účel a obsah plánu testovania

FL-5.2.2 (K2) Rozlišovať medzi rôznymi testovacími stratégiami

FL-5.2.3 (K2) Uviesť príklady potenciálnych vstupných a výstupných kritérií

FL-5.2.4 (K3) Použiť znalosti o prioritách a závislostiach (technických a logických) pri vytváraní harmonogramu vykonávania testov pre danú sadu testovacích prípadov

FL-5.2.5 (K1) Identifikovať faktory, ktoré ovplyvňujú prácnosť testovania

FL-5.2.6 (K2) Vysvetliť rozdiel medzi technikou odhadu založenej na metrikách a technikou založenou na znalostiach

5.3 Monitorovanie a riadenie testovania

FL-5.3.1 (K1) Zapamätať si metriky používané pri testovaní

FL-5.3.2 (K2) Zhrnúť účel, obsah a cieľovú skupinu správ z testovania

5.4 Konfiguračný manažment

FL-5.4.1 (K2) Zhrnúť ako konfiguračný manažment podporuje testovanie

5.5 Riziká a testovanie

FL-5.5.1 (K1) Definovať úroveň rizika pomocou pravdepodobnosti a dopadu

FL-5.5.2 (K2) Rozlišovať medzi projektovými a produktovými rizikami

FL-5.5.3 (K2) Pomocou príkladov opísať, ako môže analýza produktových rizík ovplyvniť dôkladnosť a rozsah testovania

5.6 Manažment defektov

FL-5.6.1 (K3) Vytvoriť správu o defekte, ktorá zahŕňa defekty odhalené počas testovania

5.1 Organizácia testovania

5.1.1 Nezávislé testovanie

Testovacie úlohy môžu vykonávať ľudia v konkrétnej testovacej role, alebo ľudia v iných rolách (napr. zákazníci). Určitý stupeň nezávislosti často robí testerov účinnejšími pri hľadaní defektov, čo je spôsobené rozdielnymi kognitívnymi predsudkami autora a testera (viď kapitola 1.5). Nezávislosťou však nejde nahradiť znalosť systému. Príkladom sú vývojári, ktorí môžu efektívne nájsť veľa defektov vo svojom vlastnom kóde.

Stupne nezávislosti pri testovaní zahŕňajú nasledujúce úrovne (od nízkej úrovne nezávislosti až po vysokú):

- Bez nezávislých testerov; jedinou formou testovania je testovanie vlastného kódu vývojármi.
- Nezávislí vývojári alebo testeri v rámci vývojových tímov alebo projektového tímu; testovanie môžu vykonávať vývojári, ktorí testujú produkty svojich kolegov.
- Nezávislý testovací tím alebo oddelenie v rámci organizácie, ktoré zodpovedá projektovému alebo výkonnému manažmentu.
- Nezávislí testeri z biznisu alebo užívateľskej komunity, alebo so špecializáciami na špecifické typy testov, ako je použiteľnosť, bezpečnosť, výkonnosť, súlad s regulatórnymi požiadavkami alebo prenositeľnosť.
- Nezávislí testeri mimo organizácie, pracujúci v mieste vývoja produktu (on-site, in-house) alebo mimo miesta vývoja (off-site, outsourcing).

Pre väčšinu typov projektov je vhodné mať viac úrovní testovania, pričom niektoré z týchto úrovní sú vykonávané nezávislými testermi. Vývojári by sa mali podieľať na testovaní, a to najmä na nižších úrovniach, aby mali kontrolu nad kvalitou vlastnej práce.

Spôsob, akým je implementovaná nezávislosť testovania, sa líši v závislosti od modelu životného cyklu vývoja softvéru. Napríklad v agilnom vývoji môžu byť testeri súčasťou vývojového tímu. V niektorých organizáciách, ktoré používajú agilné prístupy, môžu byť títo testeri považovaní za súčasť väčšieho nezávislého testovacieho tímu. Okrem toho, v takýchto organizáciách, môžu vlastníci produktov (product owner) vykonávať akceptačné testovanie na konci každej iterácie za účelom validácie užívateľských scenárov.

Medzi potenciálne výhody (+) nezávislého testovania patria:

- (+) Nezávislí testeri dokážu s väčšou pravdepodobnosťou rozpoznať rôzne druhy zlyhaní v porovnaní s vývojármi a to z dôvodu odlišného zázemia, technickej perspektívy a predsudkov.
- (+) Nezávislý tester môže overiť, spochybniť alebo vyvrátiť predpoklady, ktoré mali zainteresované strany v dobe prípravy špecifikácie a implementácie systému
- (+) Nezávislí testeri dodávateľa môžu otvorene a objektívne podať správu o testovanom systéme bez (politického) tlaku od spoločnosti, ktorá ich najala

Medzi potenciálne nevýhody (-) nezávislého testovania patria:

- (-) Izolácia od vývojového tímu môže viesť k nedostatočnej spolupráci, oneskoreniu pri poskytovaní spätnej väzby vývojovým tímom, alebo k nepriateľským vzťahom s vývojovým tímom
- (-) Vývojári môžu stratiť pocit zodpovednosti za kvalitu
- (-) Nezávislí testeri môžu byť vnímaní ako problémový článok vývojového procesu
- (-) Nezávislí testeri môžu mať nedostatok dôležitých informácií (napr. o testovanom objekte)

Mnohé organizácie sú schopné úspešne dosiahnuť prínosov nezávislosti testovania a pritom eliminovať jeho nevýhody.

5.1.2 Úlohy manažéra testovania a testera

V tejto učebnej osnove sú pokryté dve testovacie role, manažéri testovania a testeri. Činnosti a úlohy vykonávané týmito dvoma rolami závisia od kontextu projektu a produktu, zručnosti ľudí v týchto rolách a od organizácie.

Manažér testovania má celkovú zodpovednosť za proces testovania a úspešné riadenie testovacích činností. Manažment testovania môže vykonávať profesionálny manažér testovania, alebo projektový manažér, manažér vývoja alebo manažér zabezpečenia kvality. Vo väčších projektoch alebo organizáciách sa môže niekoľko testovacích tímov zodpovedať manažérovi testovania, koučovi testovania alebo koordinátorovi testovania, pričom každý tím je vedený vedúcim testovania alebo hlavným testerom.

Typické úlohy manažéra testovania môžu zahŕňať:

- vypracovať alebo revidovať politiku testovania a testovaciu stratégiu organizácie
- plánovať testovacie činnosti s ohľadom na kontext a pochopenie cieľov testovania a rizík to môže zahŕňať výber prístupov k testovaniu, odhad času, prácnosti a nákladov na testovanie, získavanie zdrojov, definovanie úrovní testovania a testovacích cyklov a plánovanie manažmentu defektov
- zostaviť a aktualizovať testovací plán(y)
- koordinovať plán(y) testovania s projektovými manažérmi, vlastníkmi produktov a ďalšími
- zdielať perspektívy testovania s inými projektovými činnosťami, ako je plánovanie integrácií
- iniciovať analýzu, návrh, implementáciu a vykonávanie testov, monitorovať priebeh a výsledky testov a kontrolovať stav plnenia výstupných kritérií, prípadne definíciu hotového (Definition of Done - DoD) a napomáhať v aktivitách dokončenia testovania
- pripraviť a dodať správy o pokroku v testovaní a súhrnné správy z testovania na základe zhromaždených informácií
- prispôsobiť plánovanie na základe výsledkov testov a pokroku testovania (niekedy zdokumentovaných v správach o pokroku v testovaní a/alebo v súhrnných správach z testovania pre iné, v projekte už dokončené, testy) a vykonať všetky opatrenia potrebné na riadenie testovania
- podporovať nastavenie nástroja na manažment defektov a adekvátneho konfiguračného manažmentu testvéru
- zaviesť vhodné metriky na meranie pokroku testovania, a hodnotenie kvality testovania a produktu
- podporovať výber a zavádzanie nástrojov na podporu procesu testovania, vrátane odporúčaní týkajúcich sa rozpočtu na výber nástrojov (a prípadne nákupu a/alebo podpory), rozvrhnúť čas a prácnosť na pilotné projekty a poskytovať priebežnú podporu pri využívaní nástrojov
- rozhodovať o implementácii testovacích prostredí
- propagovať a hájiť testerov, testovací tím a profesiu testovania v rámci organizácie
- rozvíjať zručnosti testerov a podporovať ich kariérny rozvoj (napr. prostredníctvom vzdelávacích plánov, hodnotenia výkonnosti, koučovaním, atď.)

Spôsob, akým sa vykonáva rola manažéra testovania, sa líši v závislosti od životného cyklu vývoja softvéru. Napríklad v agilnom vývoji sú niektoré z vyššie uvedených úloh riešené agilným tímom. Najmä úlohy, ktoré sa týkajú každodenného testovania v rámci tímu, sú často vykonávané testerom pracujúcim v danom tíme. Niektoré z úloh, ktoré sa týkajú viacerých tímov alebo celej organizácie, alebo úlohy

súvisia s personálnym manažmentom môžu byť vykonávané manažérmi testovania mimo vývojového tímu, ktorí sa zvyknú nazývať koučami testovania. Pre viac o riadení procesu testovania viď *Black 2009*.

Typické úlohy testera môžu zahŕňať:

- prispievať k plánom testovania a revidovať ich
- analyzovať, revidovať a posudzovať požiadavky, užívateľské scenáre, akceptačné kritériá, špecifikácie a modely (t. j. testovacia báza) s ohľadom na testovateľnosť
- identifikovať a zdokumentovať testovacie podmienky a zachytiť trasovateľnosť medzi testovacími prípadmi, testovacími podmienkami a testovacou bázou
- navrhnúť, nastaviť a overiť testovacie prostredia, často v spolupráci so systémovými administrátormi a správcami siete
- navrhnúť a implementovať testovacie prípady a testovacie procedúry
- pripraviť a získavať testovacie dáta
- vytvoriť podrobný harmonogram vykonávania testov
- vykonať testy, vyhodnotiť výsledky a zdokumentovať odchýlky od očakávaných výsledkov
- použiť vhodné nástroje na uľahčenie procesu testovania
- automatizovať testy podľa potreby (s podporou zo strany vývojárov alebo expertov na automatizáciu testov)
- vyhodnotiť nefunkcionálne charakteristiky ako je výkonnostná efektivita, spoľahlivosť, použiteľnosť, bezpečnosť, kompatibilita a prenositeľnosť
- revidovať testy vytvorené ostatnými

Ľudia, ktorí pracujú na testovacej analýze, návrhu testov, realizácií špecifických typov testov alebo automatizácií testov môžu byť odborníkmi v týchto rolách. V závislosti na produktových a projektových rizikách a zvolenom modele životného cyklu vývoja softvéru, môžu rôzni ľudia prevziať rolu testera na rôznych testovacích úrovniach. Napríklad na úrovni testovania komponentov a na úrovni integračného testovania komponentov je rola testera často vykonávaná vývojármi. Na úrovni akceptačného testovania rolu testera často vykonávajú biznis analytici, odborníci na danú problematiku a užívatelia. Na úrovni systémového testovania a na úrovni testovania systémovej integrácie je rola testera často vykonávaná nezávislým testovacím tímom. Na úrovni prevádzkového akceptačného testovania je rola testera často vykonávaná prevádzkovými a/alebo systémovými administrátormi.

5.2 Plánovanie a odhadovanie testovania

5.2.1 Účel a obsah plánu testovania

Plán testovania popisuje testovacie činnosti pre projekty vývoja a údržbové projekty. Plánovanie je ovplyvnené politikou testovania a testovacou stratégiou organizácie, životným cyklom vývoja a použitými metódami (viď kapitola 2.1), rozsahom testovania, cieľmi, rizikami, obmedzeniami, kritickosťou, testovateľnosťou a dostupnosťou zdrojov.

S postupom projektu a postupom plánovania testovania je k dispozícii viac informácií a detailov, ktoré môžu byť zahrnuté do plánu testovania. Plánovanie testovania je nepretržitá činnosť a vykonáva sa počas celého životného cyklu produktu. Je však nutné si uvedomiť, že životný cyklus produktu môže presahovať rozsah projektu, aby zahrnul aj fázu údržby. Spätná väzba poskytovaná prostredníctvom testovacích činností by sa mala použiť na rozpoznanie meniacich sa rizík a zohľadniť ich pri úprave plánovania. Plánovanie môže byť zdokumentované v hlavnom pláne testovania a v samostatných plánoch testovania pre jednotlivé úrovne testovania (ako je systémové testovanie a akceptačné testovanie), alebo pre jednotlivé typy testov (ako je testovanie použiteľnosti a testovanie výkonnosti).

Plánovanie testovania môže zahŕňať nasledujúce činnosti, z ktorých niektoré môžu byť zdokumentované v pláne testovania:

- určenie rozsahu, cieľov a rizík testovania
- definovanie celkového prístupu k testovaniu
- integrácia a koordinácia testovacích činností do činností životného cyklu softvéru
- rozhodovanie o tom, čo testovať, ktorí ľudia a ďalšie zdroje sú potrebné na vykonávanie rôznych testovacích činností a ako budú vykonané testovacie činnosti
- plánovanie testovacej analýzy, návrhu, implementácie, vykonávania testov a činností pre vyhodnocovanie a to buď v konkrétnych dátumoch (napr. v sekvenčnom vývoji) alebo v kontexte každej iterácie (napr. v iteratívom vývoji)
- výber metrík na monitorovanie a riadenie testovania
- určenie rozpočtu pre testovacie činnosti
- určenie úrovne detailov a štruktúry testovacej dokumentácie (napr. poskytnutím šablón alebo vzorového dokumentu)

Obsah plánov testovania sa líši a môže presahovať vyššie uvedené témy. Štruktúru vzorového plánu testovania a vzorový plán testovania možno nájsť v norme *ISO/IEC/IEEE 29119-3*.

5.2.2 Testovacia stratégia a prístup k testovaniu

Testovacia stratégia poskytuje obecný popis procesu testovania, zvyčajne na produktovej úrovni alebo na úrovni organizácie. Medzi obecné typy testovacích stratégií patria:

- **Analytické:** Tento typ testovacej stratégie je založený na analýze určitých faktorov (napr. požiadavky alebo rizika). Testovanie založené na rizikách je príkladom analytického prístupu, pri ktorom sú testy navrhnuté a prioritizované na základe úrovne rizika.
- **Založené na modeli:** U tohoto typu testovacej stratégie sú testy navrhnuté na základe modelov požadovaného aspektu produktu, ako sú napríklad funkcie, biznis procesy, vnútorná štruktúra alebo nefunkcionálna charakteristika (napr. spoľahlivosť). Príkladmi takýchto modelov sú modely biznis procesov, stavové modely a modely rastu spoľahlivosti.
- **Metodické:** Tento typ testovacej stratégie sa opiera o systematické použitie niektorých vopred definovaných testov alebo testovacích podmienok, ako je napríklad taxonómia bežných alebo pravdepodobných typov zlyhaní, zoznam dôležitých charakteristík kvality, alebo organizačné štandardy pre jednotný vzhľad (company-wide look-and-feel standard) pre mobilné aplikácie alebo webové stránky.
- **Založené na zhode s procesom (alebo normou):** Tento typ stratégie testovania zahŕňa analýzu, návrh a implementáciu testov založených na externých pravidlách a štandardoch. Tie môžu byť špecifikované normami pre jednotlivé odvetvia, dokumentáciou procesov, dôslednou identifikáciou a použitím testovacej bázy, alebo akýmkoľvek procesom alebo štandardom, ktorý organizácia definuje alebo je v organizácií nadefinovaný.
- **Riadené (alebo konzultačné):** Tento typ testovacej stratégie je založený predovšetkým na radách, konzultáciách a pokynoch zainteresovaných strán, odborníkov v danej oblasti alebo technických expertov, ktorí môžu byť mimo testovacieho tímu alebo mimo samotnej organizácie.
- **Regresne averzné:** Tento typ testovacej stratégie je motivovaný snahou vyhnúť sa regresii existujúcich schopností produktu. Táto testovacia stratégia zahŕňa opätovné použitie existujúceho testvéru (najmä testovacie prípady a testovacie dáta), rozsiahlu automatizáciu regresných testov a štandardné testovacie sady.

- **Reaktívne:** Pri tomto type testovacej stratégie je priebeh testovania reakciou na správanie testovaného komponentu alebo systému. Zohľadňuje udalosti, ku ktorým dochádza behom vykonávania testov bez toho, aby bolo testovanie vopred naplánované (ako je to u predchádzajúcich stratégiách). Testy sú navrhované, implementované a môžu byť vykonávané v okamžitej reakcii na poznatky získané z výsledkov predošlých testov. Prieskumné testovanie je obvyklou technikou používanou v reaktívnych stratégiách.

Vhodná testovacia stratégia je často tvorená kombináciou niekoľkých typov testovacích stratégií. Napríklad testovanie založené na rizikách (analytická stratégia) možno kombinovať s prieskumným testovaním (reaktívna stratégia), kde sa vzájomne dopĺňajú a pri spoločnom použití môžu dosiahnuť efektívnejšie testovanie.

Zatiaľ čo testovacia stratégia poskytuje všeobecný popis procesu testovania, prístup k testovaniu upravuje a prispôsobuje testovaciu stratégiu pre konkrétny projekt alebo vydanie. Prístup k testovaniu tvorí základ pre výber techník testovania, úrovní testovania, typov testov a na definovanie vstupných kritérií a výstupných kritérií (prípadne pre definíciu pripravenosti a definíciu hotového). Prispôsobenie obecnej stratégie je založené na rozhodnutiach vykonaných v súvislosti s komplexnosťou a cieľmi projektu, povahou vyvíjaného produktu a analýzou produktových rizík. Zvolený prístup závisí od kontextu a môže brať do úvahy faktory, ako sú riziká, bezpečie (safety), dostupné zdroje a zručnosti, technológie, povahu systému (napr. systém na zákazku verzus krabicový softvér), ciele testovania a predpisy.

5.2.3 Vstupné a výstupné kritériá (definícia pripravenosti a definícia hotového)

Aby bolo možné zaistiť efektívne riadenie kvality softvéru a testovania, odporúča sa stanoviť si kritériá, ktoré definujú, kedy sa má daná testovacia činnosť začať a kedy je daná činnosť dokončená. Vstupné kritériá (v agilnom vývoji typicky nazývané ako definícia pripravenosti) definujú vstupné podmienky pre vykonanie danej testovacej činnosti. Ak nie sú splnené vstupné kritériá je pravdepodobné, že činnosť bude ťažšia na vykonanie, časovo náročnejšia, nákladnejšia a rizikovejšia. Výstupné kritériá (v agilnom vývoji typicky nazývané ako definícia hotového) definujú, aké podmienky musia byť dosiahnuté, aby bolo možné prehlásiť úroveň testovania alebo sadu testov za dokončené. Vstupné a výstupné kritériá by mali byť definované pre každú úroveň testovania a typ testu. Tieto kritériá sa budú líšiť na základe cieľov testovania.

Typické vstupné kritériá zahŕňajú:

- dostupnosť testovateľných požiadaviek, užívateľských scenárov a/alebo modelov (napr. pri testovacej stratégii založenej na modeloch)
- dostupnosť testovacích položiek, ktoré splnili výstupné kritériá pre predchádzajúce úrovne testovania
- dostupnosť testovacieho prostredia
- dostupnosť potrebných testovacích nástrojov
- dostupnosť testovacích dát a iných potrebných zdrojov

Typické výstupné kritériá zahŕňajú:

- vykonanie plánovaných testov
- dosiahnutie stanovenej úrovne pokrytia (napr. pokrytie požiadaviek, užívateľských scenárov, akceptačných kritérií, rizík, kódu)
- počet nevyriešených defektov je v rámci dohodnutého limitu
- počet odhadovaných zostávajúcich defektov je dostatočne nízky

- hodnotené úrovne spoľahlivosti, výkonnostnej efektivity, použiteľnosti, bezpečnosti a iných relevantných charakteristík kvality sú dostatočné

Je bežné, že testovacie činnosti sú zredukované alebo úplne ukončené bez splnenia výstupných kritérií a to v dôsledku vyčerpania rozpočtu, vypršania plánovaného času alebo z dôvodu tlaku na to, aby bol produkt uvedený na trh načas. Za takýchto okolností môže byť ukončenie testovania prijateľné, ak zainteresované strany a vlastníci biznisu v rámci projektu preskúmali a akceptovali riziká spojené s vydaním produktu bez ďalšieho testovania.

5.2.4 Harmonogram vykonania testov

Po vytvorení rôznych testovacích prípadov a testovacích procedúr (s potencionálne automatizovanými testovacími procedúrami) a ich zostavení do testovacích sád môžu byť testovacie sady usporiadané do harmonogramu vykonania testov, ktorý definuje poradie vykonávania. Harmonogram vykonania testov by mal zohľadňovať faktory ako je stanovenie priorít, závislosti, konfirmačné testovanie, regresné testy a najefektívnejšia sekvencia na vykonávanie testov.

V ideálnom prípade by mali byť testovacie prípady zoradené podľa priorít a testovacie prípady s najvyššou prioritou vykonané ako prvé. Toto nemusí platiť v prípadoch, keď medzi testovacími prípadmi alebo testovanými vlastnosťami systému existujú závislosti. Ak je testovací prípad s vyššou prioritou závislý na testovacom prípade s nižšou prioritou, tak sa musí najprv vykonať testovací prípad s nižšou prioritou. Podobne, ak existujú závislosti medzi testovacími prípadmi je nutné ich zoradiť vhodným spôsobom bez ohľadu na ich relatívne priority. Konfirmačné a regresné testy musia byť tiež zoradené podľa priorít, a to na základe dôležitosti rýchlej spätnej väzby na zmeny, aj keď i tu môžu existovať isté závislosti.

V niektorých prípadoch môže vykonanie rôznych sekvencií testov priniesť rozdielnu účinnosť testovania. V takýchto prípadoch dochádza ku kompromisom medzi efektivitou vykonania testu v porovnaní s dodržiavaním stanovených priorít.

5.2.5 Faktory ovplyvňujúce prácnosť testovania

Odhad prácnosti testovania zahŕňa predpovedanie množstva práce, ktoré bude potrebné na splnenie cieľov testovania konkrétneho projektu, vydania alebo iterácie. Medzi faktory ovplyvňujúce prácnosť testovania patria charakteristiky produktu, vývojového procesu, ľudí a výsledky testov.

Charakteristiky produktu

- riziká spojené s produktom
- kvalita testovacej bázy
- veľkosť produktu
- zložitosť produktovej domény (oblasti, pre ktorú je produkt určený)
- požiadavky na charakteristiky kvality (napr. bezpečnosť, spoľahlivosť)
- požadovaná úroveň detailu testovacej dokumentácie
- požiadavky na súlad s právnymi a regulatórnymi normami

Charakteristiky vývojového procesu

- stabilita a zrelosť organizácie
- použitý model vývoja softvéru
- prístup k testovaniu
- použité nástroje

- proces testovania
- časová tieseň

Charakteristiky ľudí

- zručnosti a skúsenosti zúčastnených osôb, najmä s podobnými projektmi a produktmi (napr. doménová znalosť)
- tímová súdržnosť a vedenie tímu

Výsledky testov

- počet a závažnosť nájdených defektov
- množstvo potrebných zmien

5.2.6 Techniky odhadovania testovania

Existuje celá rada techník odhadovania, ktoré slúžia na určenie prácnosti potrebnej na primerané testovanie. Dve z najčastejšie používaných techník sú:

- Technika založená na metrikách: odhad prácnosti testovania založený na metrikách predošlých podobných projektov alebo na základe typických hodnôt
- Technika založená na expertoch: odhad prácnosti testovania na základe skúseností vlastníkov testovacích úloh alebo odborníkov

V agilnom vývoji sú príkladom prístupov založených na metrikách grafy burn-down, kedy sa zostávajúca prácnosť najprv zaznamená a reportuje. Následne sa táto informácia používa ako podklad pre určenie rýchlosti tímu (team velocity) a tým vieme odhadnúť množstvo práce, ktoré tím zvládne dodať v ďalšej iterácii. Príkladom prístupu založenom na expertoch je plánovací poker, kde členovia tímu na základe ich skúseností odhadujú prácnosť potrebnú na dodanie vlastnosti (viď učebná osnova *ISTQB-CTFL-AT*.

Pri sekvenčných projektoch je príkladom prístupu založeného na metrikách odhadovanie pomocou modelov odstraňovania defektov. V týchto prípadoch sa zaznamenáva a reportuje množstvo defektov a čas na ich odstránenie, čo poskytuje podklad pre odhady budúcich projektov podobného charakteru. Príkladom prístupu založenom na expertoch je technika odhadu Wideband Delphi, pri ktorej skupiny expertov poskytujú odhady na základe ich skúseností (viď učebná osnova *ISTQB-CTAL-TM*).

5.3 Monitorovanie a riadenie testovania

Účelom monitorovania testovania je zhromažďovať informácie a poskytovať spätnú väzbu o testovacích činnostiach ako aj ich zviditeľňovanie. Informácie, ktoré treba monitorovať, môžu byť zhromažďované manuálne alebo automaticky. Získané informácie by sa mali použiť na posúdenie postupu prác pri testovaní a na meranie toho, či sú splnené výstupné kritériá definované pre testovanie, ako je napríklad splnenie cieľov pre pokrytie produktových rizík, požiadaviek alebo akceptačných kritérií. V prípade agilných projektov sa jedná o splnenie testovacích činností týkajúcich sa definície hotového.

Riadenie testovania označuje akékoľvek usmerňujúce alebo nápravné akcie prijaté v dôsledku zhromaždených informácií a metrík. Akcie môžu pokrývať akúkoľvek testovaciu činnosť a môžu ovplyvniť akúkoľvek inú činnosť v rámci životného cyklu softvéru.

Príklady akcií v rámci riadenia testovania zahŕňajú:

- prehodnotenie priorít testov v prípade, že nastane identifikované riziko (napr. neskoré dodanie softvéru)

- zmena harmonogramu testovania z dôvodu dostupnosti alebo nedostupnosti testovacieho prostredia alebo iných zdrojov
- prehodnotenie, či testovacia položka spĺňa vstupné alebo výstupné kritériá z dôvodu vykonaných zmien

5.3.1 Metriky používané pri testovaní

Metriky sa môžu zbierať počas a na konci testovacích činností s cieľom posúdiť:

- pokrok prác voči plánovanému harmonogramu a rozpočtu
- aktuálnu kvalita testovaného objektu
- vhodnosť zvoleného prístupu k testovaniu
- efektívnosť testovacích činností v súvislosti s cieľmi

Obecné testovacie metriky zahŕňajú:

- percentuálny podiel plánovanej a vykonanej práce v rámci prípravy testovacích prípadov (alebo percentuálny podiel plánovaných a implementovaných testovacích prípadov)
- percentuálny podiel plánovanej a vykonanej práce pri príprave testovacieho prostredia
- pokrok vo vykonávaní testovacích prípadov (napr. počet vykonaných/nevykonaných testovacích prípadov, testovacích prípadov alebo testovacích podmienok, ktoré prešli/zlyhali)
- informácie o defektoch (napr. hustota defektov, zistené a opravené defekty, miera zlyhania a výsledky konfirmačného testovania)
- mieru pokrytia požiadaviek, užívateľských scenárov, akceptačných kritérií, rizík alebo kódu testami
- mieru dokončenia úloh, alokácia a využitie zdrojov, prácnosť
- náklady na testovanie vrátane pomeru nákladov a prínosov zistenia ďalšieho defektu alebo pomeru nákladov a prínosov vykonania ďalšieho testu

5.3.2 Účel, obsah a publikum správ z testovania

Účelom správ z testovania je zhrnúť a zdielať informácie o testovacej činnosti počas jej priebehu a na jej konci (napr. pre danú úroveň testovania). Správa z testovania vypracovaná počas testovacej činnosti je označovaná ako správa o pokroku v testovaní, zatiaľ čo správa vypracovaná na konci testovacej činnosti je označovaná ako súhrnná správa z testovania.

Počas monitorovania a kontroly testovania je úlohou manažéra testovania pravidelne podávať správy o pokroku v testovaní zainteresovaným stranám. Okrem obsahu spoločného pre správy o pokroku v testovaní a súhrnné správy z testovania môžu typické správy o pokroku v testovaní zahŕňať aj:

- stav jednotlivých testovacích činností a pokrok oproti plánu testovania
- faktory, ktoré bránia pokroku v práci
- plánované testovanie, ktoré bude pokryté ďalšími správami z testovania
- kvalita testovaného objektu

V momente dosiahnutia výstupných kritérií, manažér testovania vydá súhrnnú správu z testovania. Táto správa poskytuje zhrnutie vykonaných testov na základe poslednej správy o pokroku v testovaní a akékoľvek ďalšie relevantné informácie.

Typická súhrnná správa z testovania môže zahŕňať:

- zhrnutie vykonaných testov

- informácie o tom, čo sa dialo počas obdobia testovania
- odchýlky od plánu vrátane odchýlok od harmonogramu, odchýlky v trvaní alebo v prácnosti testovacích činností
- stav testovania a kvality produktu vzhľadom na výstupné kritériá alebo definíciu hotového (DoD)
- faktory, ktoré zablokovali alebo stále blokujú pokrok prác
- metriky defektov, testovacích prípadov, pokrytia, pokroku prác a čerpania zdrojov (napr. ako je vysvetlené v kapitole 5.3.1)
- reziduálne (zbytkové) riziká (viď kapitola 5.5)
- pracovné produkty testovania vhodné pre opakované použitie

Obsah správy z testovania sa bude líšiť v závislosti od projektu, organizačných požiadaviek a životného cyklu vývoja softvéru. Napríklad komplexný projekt s množstvom zainteresovaných strán alebo projekt, ktorý podlieha regulatórnym nariadeniam, môžu vyžadovať podrobnejšie a prísnejšie reportovanie oproti projektom, ktoré zaisťujú drobnú aktualizáciu softvéru. V agilnom vývoji môže byť správa o pokroku v testovaní začlenená do tabuliek úloh (task boards), prehľadu defektov alebo burn-down grafov. Tie môžu byť následne diskutované počas denných schôdzok (stan-up meetings). Pre viac informácií viď učebná osnova *ISTQB-CTFL-AT*.

Správy z testovania by mali byť prispôsobené kontextu projektu, ale aj publiku resp. cieľovej skupine správ. Typ a množstvo informácií, ktoré by mali byť zahrnuté pre technické publikum alebo testovací tím sa môžu líšiť od toho, čo by bolo zahrnuté do súhrnnej správy pre vedie firmy. V prvom prípade môžu byť dôležité podrobné informácie o typoch defektu a ich trendoch. V druhom prípade môže byť vhodnejšie poskytnúť obecnú (high-level) správu (napr. súhrnný stav defektov podľa priority, aktuálny stav čerpania rozpočtu a harmonogramu, testovacie podmienky, ktoré prešli/neprešli, prípadne neboli testované)

Norma *ISO/IEC/IEEE 29119-3* popisuje dva typy správ z testovania: správa o pokroku v testovaní a správa o dokončení testovania (v tejto učebnej osnove nazývaná ako súhrnná správa z testovania) a zároveň obsahuje doporučenú štruktúru a príklady.

5.4 Konfiguračný manažment

Účelom konfiguračného manažmentu je vytvoriť a udržiavať integritu komponentu alebo systému, testvéru a ich vzájomných vzťahov behom projektového a produktového životného cyklu.

Konfiguračný manažment by mal z pohľadu podpory kvality testovania zaistiť nasledujúce:

- Všetky testovacie položky sú unikátne identifikované, ich verzie sú riadené, zmeny v nich sú sledované a medzi sebou majú definovaný vzťah.
- Všetky položky testvéru sú unikátne identifikované, ich verzie sú riadené, zmeny v nich sú sledované a medzi sebou majú definovaný vzťah. Zároveň je definovaný vzťah aj s verziami testovacích položiek, aby bola zachovaná trasovateľnosť počas celého procesu testovania.
- Všetky identifikované dokumenty a softvérové položky sú jednoznačne odkazované v testovacej dokumentácii.

Počas plánovania testovania by sa mali identifikovať a implementovať postupy konfiguračného manažmentu a infraštruktúra (nástrojov).

5.5 Riziká a testovanie

5.5.1 Definícia rizík

Riziko je udalosť, ktorá môže nastať v budúcnosti a má negatívne následky. Úroveň rizika je určená pravdepodobnosťou, že udalosť nastane, a dopadom (škodami), ktorý má po tom, čo udalosť nastane.

5.5.2 Projektové a produktové riziká

Produktové riziko sa týka možnosti, že pracovný produkt (napr. špecifikácia, komponent, systém alebo test) môže neuspokojiť oprávnené potreby svojich užívateľov alebo zainteresovaných strán. Ak sú produktové riziká spojené so špecifickými charakteristikami kvality produktu (napr. funkcionálna vhodnosť, spoľahlivosť, výkonnostná efektivita, použiteľnosť, bezpečnosť, kompatibilita, udržateľnosť a prenositeľnosť), tak sa tieto produktové riziká môžu označovať ako riziká kvality. Príklady produktových rizík zahŕňajú:

- softvér nemusí plniť svoje zamýšľané funkcie podľa špecifikácie
- softvér nemusí vykonávať plánované funkcie podľa potrieb užívateľov, zákazníkov a/alebo zainteresovaných strán.
- architektúra systému nemusí adekvátne podporovať niektoré nefunkcionálne požiadavky
- za určitých okolností môžu byť konkrétne výpočty vykonané nesprávne
- riadiace štruktúry cyklu môžu byť naprogramované nesprávne
- doba odozvy môže byť nedostatočná pre systém spracovania transakcií, ktorý vyžaduje vysoký výkon
- užívateľský zážitok (user experience - UX) nemusí napĺňať očakávania produktu

Projektové riziká zahŕňajú situácie, ktoré pokiaľ by nastali, by mohli mať negatívny dopad na dosiahnutie projektových cieľov. Príklady projektových rizík zahŕňajú:

- Projektové problémy:
  - oneskorenia pri dodaní alebo dokončení úloh, splnenie výstupných kritérií alebo definície hotového
  - nepresné odhady, prerozdelenie finančných prostriedkov na projekty s vyššou prioritou alebo všeobecné zníženie rozpočtu naprieč organizáciou, môže viesť k nedostatočnému financovaniu
  - zmeny v neskorých fázach projektu môžu vyžadovať dodatočné náklady na prepracovanie
- Organizačné problémy:
  - nedostatočné zručnosti, školenia a počet zamestnancov
  - osobné problémy môžu spôsobiť konflikty a narušovať medziľudské vzťahy
  - užívatelia, zástupcovia biznisu alebo odborníci nemusia byť k dispozícii v dôsledku protichodných priorít biznisu
- Politické problémy:
  - testeri nemusia primerane komunikovať svoje potreby a/alebo výsledky testov
  - vývojári a/alebo testeri nemusia zohľadniť informácie získané pri testovaní a počas revízie (napr. nebudú zlepšovať vývojové a testovacie postupy)
  - môže existovať nevhodný postoj k testovaniu alebo neprimerané očakávania od testovania (napr. nedocenenie hodnoty zistenia defektov počas testovania)

- Technické problémy:
  - požiadavky nemusia byť dostatočne dobre definované
  - požiadavky nemusia byť splnené vzhľadom na existujúce obmedzenia
  - testovacie prostredie nemusí byť pripravené včas
  - konverzia dát, plánovanie migrácie a ich podpora nástrojmi môže byť opozdená
  - nedostatky vo vývojovom procese môžu mať vplyv na konzistenciu alebo kvalitu projektových pracovných produktov, ako sú návrh, kód, konfigurácia, testovacie dáta a testovacie prípady
  - zlý manažment defektov a problémy tomu podobné môžu mať za následok nahromadenie defektov ako aj iné prvky technického dlhu
- Problémy s dodávateľom:
  - tretia strana nemusí dodať potrebný produkt alebo službu, alebo dokonca môže skrachovať
  - zmluvné problémy môžu spôsobiť problémy v projekte

Projektové riziká môžu ovplyvniť ako vývojové tak aj testovacie činnosti. V niektorých prípadoch sú projektoví manažéri zodpovední za riešenie všetkých projektových rizík, ale nieje neobvyklé, keď je zodpovednosť za projektové riziká súvisiace s testovaním delegovaná na manažérov testovania.

5.5.3 Testovanie založené na rizikách a kvalita produktu

Riziká sú často využívané na lepšie zacielenie testovacích činností. Používajú sa na rozhodovanie o tom, kde a kedy začať testovať a na identifikovanie oblastí, ktoré vyžadujú väčšiu pozornosť. Testovanie sa používa na zníženie pravdepodobnosti výskytu nežiadúcej udalosti alebo na zníženie jej dopadu. Testovanie sa používa ako činnosť na zmierňovanie rizík, na poskytovanie informácií k identifikovaným rizikám, ako aj na poskytovanie informácií o zostatkových (nevyriešených) rizikách.

Prístup k testovaniu založený na rizikách poskytuje proaktívne príležitosti na zníženie úrovne produktových rizík. Zahŕňa analýzu produktových rizík, ktorá zahŕňa identifikáciu produktových rizík a posúdenie pravdepodobnosti a dopadu každého rizika. Výsledné informácie o produktových rizikách sa používajú na usmernenie plánovania testovania, špecifikáciu, prípravu a vykonanie testovacích prípadov, monitorovanie a riadenie testovania. Včasná analýza produktových rizík prispieva k úspechu projektu.

V prístupe založenom na rizikách sa výsledky analýzy produktových rizík používajú na:

- určenie techník testovania, ktoré sa majú použiť
- určenie konkrétnych úrovní a typov testovania, ktoré sa majú vykonať (napr. testovanie bezpečnosti, testovanie prístupnosti)
- určenie rozsahu testovania, ktoré sa má vykonať
- prioritizáciu testovania s cieľom nájsť kritické defekty čo najskôr
- určenie, či môžu byť na zníženie rizika použité iné činnosti ako testovanie (napr. zabezpečenie školení pre neskúsených návrhárov)

Testovanie založené na rizikách stavia pri analýze projektových rizík na kolektívnych znalostiach a prehľade zainteresovaných strán projektu. Pre minimalizáciu pravdepodobnosti zlyhania produktu zabezpečujú funkcie riadenia rizík systematický prístup pre:

- analýzu (a pravidelné prehodnocovanie) toho, čo sa môže pokaziť (riziká)
- určenie rizík, ktoré je nutné riešiť
- zavedenie opatrení na zmiernenie týchto rizík
- uskutočnenie pohotovostných plánov na riešenie rizík pre prípad, že skutočne nastanú

Testovanie môže identifikovať nové riziká, pomáha určiť, ktoré riziká by mali byť zmiernené, a tiež prispieva k zníženiu neistoty ohľadom rizík.

5.6 Manažment defektov

Keďže jedným z cieľov testovania je nájdenie defektov, musia byť defekty nájdené počas testovania zaznamenané. Spôsob, akým sa zaznamenávajú defekty sa môže líšiť v závislosti od kontextu komponentu alebo systému, úrovne testovania a modelu životného cyklu vývoja softvéru. Všetky zistené defekty by mali byť preskúmané a sledované od ich odhalenia a klasifikácie, až po ich vyriešenie (napr. oprava defektu a úspešné konfirmačné testovanie, odloženie riešenia do následujúceho vydania, prijatie ako trvalé obmedzenie produktu, atď.). Z toho dôvodu musí organizácia vytvoriť proces manažmentu defektov, ktorého cieľom je vyriešenie všetkých defektov, zahŕňajúc pracovný tok (workflow) a pravidlá pre klasifikáciu. Tento proces musí byť odsúhlasený všetkými, ktorí sa zúčastňujú na manažmente defektov, vrátane architektov, návrhárov, vývojárov, testerov a vlastníkov produktov. V niektorých organizáciách môže byť zaznamenávanie a sledovanie defektov veľmi neformálne.

Počas procesu manažmentu defektov sa niektoré záznamy defektov môžu ukázať ako falošne-pozitívne, a nie ako ozajstné zlyhania spôsobené defektmi. Napríklad k zlyhaniu testu môže dôjsť v prípade výpadku siete alebo vypršaní časového limitu pre operáciu. Toto správanie nie je výsledkom defektu v testovanom objekte, ale je to anomália, ktorá vyžaduje preskúmanie. Testeri by sa mali pokúsiť minimalizovať počet nahlásených falošne-pozitívnych defektov.

Defekty môžu byť hlásené počas kódovania, statickej analýzy, revízií, dynamického testovania alebo počas používania softvérového produktu. Defekty môžu byť hlásené v prípade problémov v kóde, v bežiacich systémoch alebo v akomkoľvek type dokumentácie vrátane požiadaviek, užívateľských scenárov, akceptačných kritérií, vývojových dokumentov, testovacích dokumentov, návodov alebo inštalačných príručiek. V záujme efektívneho a účinného procesu manažmentu defektov môžu organizácie definovať normy pre atribúty, klasifikáciu a pracovné toky defektov.

Typicky majú správy o defekte tieto ciele:

- poskytnúť vývojárom a iným stranám informácie o akejkoľvek nežiaducej udalosti, ktorá nastala, aby mohli identifikovať špecifické dopady, izolovať problém s čo najjednoduchším reprodukujúcim testom a opraviť potenciálne defekty podľa potreby, alebo inak vyriešiť daný problém
- poskytnúť manažérom testovania prostriedok na sledovanie kvality pracovného produktu a vplyvu na testovanie (napr. ak je hlásených veľa defektov, testeri strávia veľa času ich hlásením namiesto vykonávania testov, a zároveň bude viac konfirmačného testovania)
- poskytnúť nápady pre zlepšenie procesu vývoja a testovaniaSpráva o defekte vytvorená počas dynamického testovania typicky zahŕňa:
- identifikátor
- názov a krátky súhrn defektu, ktorý sa reportuje
- dátum správy o defekte, organizácia a autor, ktorý defekt reportujú
- identifikáciu testovanej položky (testovaná konfiguračná položka) a prostredia
- fázu životného cyklu vývoja softvéru, v ktorej bol defekt nájdený
- popis defektu umožňujúci reprodukciu a vyriešenie, vrátane protokolov (logs), záloh databáze alebo obrazový záznam (ak sa defekt zistí počas vykonávania testu)
- očakávané a skutočné výsledky
- rozsah alebo stupeň dopadu (závažnosti) defektu na záujmy zúčastnených strán
- naliehavosť/priorita opravy

- stav defektu (napr. otvorený, odložený, duplikát, čakajúci na opravu, čakajúci na konfirmačné testovanie, opätovné otvorený, uzatvorený)
- závery, odporúčania a záznamy o schvaľovaní
- globálne problémy, ako napríklad iné oblasti, ktoré môžu byť ovplyvnené zmenou vyplývajúcou z defektu
- história zmien, ako napríklad postupnosť akcií prijatých členmi projektového tímu v súvislosti s izoláciu defektu, opravy a potvrdenia jej korektnosti
- odkazy, vrátane testovacieho prípadu, ktorý odhalil problém

Niektoré z týchto detailných informácií môžu byť automaticky vkladané a/alebo spravované použitím nástrojov na manažment defektov. A to napríklad automatické priraďenie identifikátora, priradenie a aktualizácia stavu správy o defekte v priebehu svojho životného cyklu, atď. Defekty zistené počas statického testovania, najmä revízií, sa zvyčajne dokumentujú iným spôsobom, napríklad v zápise z revíznej schôdze.

Príklad obsahu správy o defekte možno nájsť v norme *ISO (ISO/IEC/IEEE 29119-3)* (ktorá nazýva správy o defekte ako správy o incidente).

6 Nástroje na podporu testovania – 40 minút

Kľúčové slová

testovanie riadené dátami, testovanie riadené kľúčovými slovami, automatizácia testov, nástroj na vykonávanie testov, nástroj na manažment testovania

Študijné ciele pre kapitolu 6 – Nástroje na podporu testovania:

6.1 Zvažovanie testovacích nástrojov

FL-6.1.1 (K2) Klasifikovať testovacie nástroje podľa ich účelu a podpory testovacích činností

FL-6.1.2 (K1) Identifikovať výhody a riziká automatizácie testov

FL-6.1.3 (K1) Zapamätať si špecifické ohľady týkajúce sa nástrojov na vykonávanie testov a nástrojov na manažment testovania

6.2 Efektívne využívanie nástrojov

FL-6.2.1 (K1) Identifikovať hlavné zásady pre výber nástroja

FL-6.2.2 (K1) Zapamätať si ciele využívania pilotných projektov pri zavádzaní nástrojov

FL-6.2.3 (K1) Identifikovať faktory úspechu pre vyhodnotenie, implementáciu, nasadenie a priebežnú podporu testovacích nástrojov v rámci organizácie

6.1 Zvažovanie testovacích nástrojov

Testovacie nástroje môžu byť použité na podporu jednej alebo viacerých testovacích činností. Základné kategórie nástrojov zahŕňajú:

- nástroje, ktoré sa priamo používajú pri testovaní, ako sú nástroje na vykonávanie testov a nástroje na prípravu testovacích dát
- nástroje, ktoré pomáhajú spravovať požiadavky, testovacie prípady, testovacie procedúry, automatizované testovacie skripty, výsledky testov, testovacie dáta, defekty a nástroje na reportovanie a monitorovanie vykonania testu
- nástroje, ktoré sa používajú na analýzu a hodnotenie
- akýkoľvek nástroj, ktorý pomáha pri testovaní (v tomto prípade je testovací nástroj aj tabuľkový softvér)

6.1.1 Klasifikácia testovacích nástrojov

V závislosti od kontextu môžu testovacie nástroje napĺňať jeden alebo viacero z nasledujúcich účelov:

- zlepšiť účinnosť testovacích činností automatizáciou opakovaných úloh alebo úloh, ktoré pri manuálnom vykonávaní vyžadujú nezanedbateľné zdroje (napr. vykonanie testu, regresné testovanie)
- zlepšiť účinnosť testovacích činností podporou manuálneho testovania počas celého procesu testovania (viď kapitola 1.4)
- zlepšiť kvalitu testovacích činností tým, že umožní konzistentnejšie testovanie a zvýši reprodukovateľnosť defektov
- automatizovať činnosti, ktoré nie je možné vykonať manuálne (napr. rozsiahle testovanie výkonnosti)
- zvýšiť spoľahlivosť testov (napr. pomocou automatizácie porovnávania veľkých dát alebo automatizácie simulácií správania)

Nástroje možno klasifikovať na základe viacerých kritérií, ako je účel, cena, licenčný model (napr. komerčný alebo open source) a použitá technológia. V tejto učebnej osnove sú nástroje klasifikované podľa testovacích činností, ktoré podporujú.

Niektoré nástroje podporujú výhradne alebo prevažne jednu činnosť. Iné môžu podporovať viac ako jednu činnosť, ale sú klasifikované pre činnosť, s ktorou sú najviac spojené. Nástroje od jediného dodávateľa, najmä tie, navrhnuté tak, aby spolupracovali, môžu byť poskytované ako integrovaný balík.

Niektoré typy testovacích nástrojov môžu ovplyvňovať výsledok (intrusive tools), teda ich použitie má dopad na výsledok testu. Napríklad skutočná doba odozvy aplikácie môže byť odlišná vzhľadom na dodatočné inštrukcie, ktoré sú spustené pomocou nástroja na testovanie výkonnosti. Alebo miera dosiahnutého pokrytia kódu môže byť skreslená práve z dôvodu použitia nástroja na meranie pokrytia. Dôsledky použitia intruzívnych nástrojov sa nazývajú efekt meracej sondy (probe efect).

Niektoré nástroje sú vhodnejšie pre vývojárov ako pre testerov (napr. nástroje, ktoré sa používajú počas testovania komponentov a integračného testovania). Tieto nástroje sú v nasledujúcich kapitolách označené symbolom D (ako developer) .

Nástroje na podporu manažmentu testovania a správu testvéru

Tieto nástroje môžu byť použité v rámci všetkých testovacích činností a počas celého životného cyklu vývoja softvéru. Medzi nástroje, ktoré podporujú riadenie testovania a správu testvéru patria:

- nástroje na manažment testovania a nástroje na správu životného cyklu aplikácií (ALM – Application Lifecycle Management)
- nástroje na správu požiadaviek (napr. trasovateľnosť na testované objekty)
- nástroje na manažment defektov
- nástroje na konfiguračný manažment
- nástroje na priebežnú integráciu (D)

Nástroje na podporu statického testovania

Nástroje na podporu statického testovania sú spojené s činnosťami a výhodami uvedenými v kapitole 3. Medzi tieto nástroje patria:

- nástroje na statickú analýzu

Nástroje na podporu návrhu a implementácie testov

Nástroje na návrh testov pomáhajú pri tvorbe dobre udržiavateľných pracovných produktov vrátane testovacích prípadov, testovacích procedúr a testovacích dát. Medzi tieto nástroje patria:

- nástroje na testovanie založenom na modeli
- nástroje na prípravu testovacích dát

V niektorých prípadoch môžu nástroje na podporu návrhu a implementácie testov tiež podporovať vykonávanie a zaznamenávanie testov, alebo týmto nástrojom na vykonávanie a zaznamenávanie testov môžu priamo poskytnúť svoje výstupy.

Nástroje na podporu vykonávania a zaznamenávania testov

Existuje mnoho nástrojov na podporu a zlepšenie vykonávania a zaznamenávania vykonávania testov. Medzi tieto nástroje patria:

- nástroje na vykonávanie testov (napr. na spustenie regresných testov)
- nástroje na meranie pokrytia (napr. požiadaviek, kódu) (D)
- testovacie vybavenie (D)

Nástroje na podporu merania výkonnosti a dynamickej analýzy

Nástroje na meranie výkonnosti a dynamickú analýzu sú nevyhnutné na podporu činností v oblasti testovania výkonnosti a záťažové testovanie, keďže tieto činnosti nie je možné účinne vykonávať manuálne. Medzi tieto nástroje patria: nástroje na testovanie výkonnosti

- nástroje na dynamickú analýzu (D)

Nástroje na podporu špecifických testovacích činností

Okrem nástrojov, ktoré podporujú obecný testovací proces, existuje mnoho ďalších nástrojov, ktoré podporujú konkrétnejšie testovacie činnosti nefunkcionálneho testovania.

6.1.2 Výhody a riziká automatizácie testov

Získanie nástroja nie je garanciou úspechu. Každý nový nástroj zavedený do organizácie bude vyžadovať dodatočné úsilie na dosiahnutie reálnych a trvalých prínosov. S použitím alebo zavedením nového nástroja sú spojené potenciálne prínosy, ale aj riziká. Platí to najmä pre nástroje na vykonávanie testov, ktoré sa často označujú ako nástroje na automatizáciu testov.

Medzi potenciálne prínosy použitia nástrojov na podporu vykonávania testov patria:

- zníženie opakujúcich sa manuálnych činností, čím dochádza k úspore času (napr. vykonávanie regresných testov, nastavenie alebo vyčistenie testovacieho prostredia, opätovné zadávanie rovnakých testovacích dát alebo kontrola voči normám kódovania).
- vyššia konzistencia a opakovateľnosť (napr. testovacie dáta sa vytvárajú jednotným spôsobom, testy sú spustené pomocou nástroja v rovnakom poradí a s rovnakou frekvenciou, testy sú dôsledne odvodzované z požiadaviek)
- objektívnejšie vyhodnotenie (napr. metriky statických metód a miery pokrytia)
- ľahší prístup k informáciám o testovaní (napr. štatistiky a grafy o pokroku v testovaní, o výskyte defektov alebo výkonnostných charakteristikách)

Medzi potenciálne riziká použitia nástrojov na podporu vykonávania testov patria:

- nerealistické očakávania od nástroja (vrátane funkčnosti a jednoduchosti použitia)
- podcenenie času, nákladov a prácnosti na počiatočné zavedenie nástroja (vrátane školení a nákladov na externých špecialistov)
- podcenenie času a prácnosti potrebnej na dosiahnutie trvalých prínosov (vrátane potrebných zmien v procese testovania a priebežného zlepšovania spôsobu, akým sa nástroj používa)
- podcenenie prácnosti potrebnej k údržbe testvéru
- prehnaná dôvera v nástroj (vnímaná ako náhrada práce testera pri navrhovaní a vykonávaní testov, alebo využitie automatizovaného testovania v prípadoch, kde by bolo manuálne testovanie vhodnejšie)
- zanedbanie správy verzií testvéru
- podcenenie problémov v rámci vzťahov a interoperability medzi kritickými nástrojmi, ako sú nástroje na správu požiadaviek, nástroje na konfiguračný manažment, nástroje na manažment defektov a nástroje od viacerých výrobcov
- dodávateľ nástroja môže ukončiť obchodnú činnosť, prestať poskytovať podporu alebo nástroj predať inému dodávateľovi
- dodávateľ môže poskytovať nízku úroveň podpory, aktualizácií a opravy chýb
- pozastavenie open source projektu
- nástroj nemusí podporovať nové platformy alebo technológie
- nejasné vlastníctvo nástroja v organizácií (napr. nie je jasné, kto zabezpečuje školenia alebo aktualizácie)

6.1.3 Špecifické ohľady týkajúce sa nástrojov na vykonávanie testov a manažment testovania

Pre hladkú a úspešnú implementáciu nástrojov na vykonávanie testov a nástrojov na manažment testovania v rámci organizácií je treba zvážiť radu faktorov.

Nástroje na vykonávanie testov

Nástroje na vykonávanie testov ovládajú testované objekty pomocou automatizovaných testovacích skriptov. Tento typ nástrojov často vyžaduje značnú prácnosť na dosiahnutie významných prínosov.

- **Prístup zachytávania testov:** Zachytenie testov pomocou nahrávania manuálnych akcií testera nástrojom typu capture/playback sa môže zdať atraktívne, ale tento prístup je zle škálovateľný pri veľkom počte testovacích skriptov. Každý zaznamenaný skript je lineárnou reprezentáciou akcií s konkrétnymi dátami, ktoré sú neoddeliteľnou súčasťou každého skriptu. Takto zaznamenaný typ skriptu môže byť v prípade neočakávaných udalostí nestabilný a stále vyžaduje priebežnú údržbu z dôvodu zmien užívateľského rozhrania v čase.

- **Prístup k testovaniu riadený dátami**: Tento prístup vyčleňuje vstup testu a očakávané výsledky (zvyčajne do tabuľky) a používa všeobecnejší testovací skript, ktorý dokáže načítať vstupné dáta a vykonať rovnaký testovací skript s rôznymi dátami.
- **Prístup k testovaniu riadený kľúčovými slovami**: Pri tomto prístupe sú definované kľúčové slová popisujúce akcie testovaného systému (napr. prihlásenie užívateľa, vyplnenie formuláru). Testovací skript je zložený z týchto kľúčových slov a odpovedajúcich testovacích dát.

Vyššie uvedené prístupy obvykle vyžadujú odborné znalosti skriptovacieho jazyka. To platí pre testerov, vývojárov alebo špecialistov na automatizáciu testov. Pri použití prístupu k testovaniu riadenehého dátami alebo kľúčovými slovami, testeri aj keď niesú oboznámení so skriptovacím jazykom, môžu prispievať vytváraním testovacích dát a/alebo kľúčových slov k vopred definovaným skriptom. Bez ohľadu na použitú skriptovaciu techniku je potrebné pre každý test porovnať očakávané výsledky so skutočnými, a to buď dynamicky (zatiaľ čo test beží) alebo ex-post na základe uložených výsledkov.

Ďalšie podrobnosti a príklady testovania riadeného dátami a testovania riadeného kľúčovými slovami sú uvedené v učebných osnovách *ISTQB-CTAL-TAE* a ďalej v *Fewster 1999* a *Buwalda 2001*.

Nástroje na testovanie založenom na modeli (MBT) umožňujú zachytiť funkcionálnu špecifikáciu vo forme modelu, ako je napríklad diagram činností. Prípravu modelu zvyčajne vykonáva návrhár systému. MBT nástroj interpretuje model s cieľom vytvoriť špecifikáciu testovacích prípadov, ktoré môžu byť následne uložené v nástroji na manažment testovania a/alebo vykonané nástrojom na vykonávanie testov (viď učebná osnova *ISTQB-CTFL-MBT*).

Nástroje na manažment testovania

Nástroje na manažment testovania často obsahujú rozhrania na komunikáciu s inými nástrojmi alebo funkcionalitu pre prácu s tabuľkami. A to z rôznych dôvodov, vrátane:

- vytvorenia užitočných informácií vo formáte, ktorý vyhovuje potrebám organizácie
- zachovanie konzistentnej trasovateľnosti požiadaviek oproti nástroju na správu požiadaviek
- prepojenie informácií o verziách testovaných objektov s nástrojom na konfiguračný manažment

Dostupnosť komunikačných rozhraní je nutné zvážiť hlavne pri zavádzaní integrovaného nástroja, napr. komplexného nástroja na manažment životného cyklu softvéru (ALM – Application Lifecycle Management). Takéto nástroje obsahujú modul na manažment testovania, ale aj iné moduly (napr. na správu harmonogramu projektu a informácií o rozpočte), s ktorými pracujú rôzne skupiny užívateľov v rámci organizácie.

6.2 Efektívne využívanie nástrojov

6.2.1 Hlavné zásady pri výbere nástrojov

Medzi hlavné aspekty pri výbere nástroja pre organizáciu patria:

- posúdenie vyspelosti danej organizácie, jej silných a slabých stránok
- identifikácia príležitostí, kde by mohlo použitie nástroja zlepšiť proces testovania
- pochopenie technológií, ktoré používa testovací objekt a zvolenia nástroja kompatibilného s týmito technológiami
- pochopenie existujúcich nástrojov na zostavovanie a priebežnú integráciu, ktoré sa v rámci organizácie používajú, aby bola zaistená vzájomná kompatibilita a integrácia
- vyhodnotenie nástroja vzhľadom k jasným požiadavkám a objektívnym kritériám
- zistenie, či nástroj nie je k dispozícii zdarma na skúšobnú dobu (trial period) vrátane jej dĺžky

- vyhodnotenie kvality dodávateľa (z pohľadu poskytovania školení, podpory a obchodných aspektov) alebo úrovne podpory pre nekomerčné nástroje (napr. u open source licenciách)
- identifikácia interných požiadaviek na zaškolenie a priebežná podpora užívateľom pri používaní nástroja
- vyhodnotenie potrieb školenia na obecné zručnosti v oblasti testovania (a automatizácie testov) pre tých, ktorí budú s nástrojom priamo pracovať
- zváženie výhod a nevýhod rôznych licenčných modelov (napr. komerčných alebo open source)
- odhad pomeru nákladov a prínosov na základe konkrétneho biznis prípadu (ak je vyžadovaný)

Záverečným krokom výberového konania by malo byť vyhodnotenie overenia konceptu (proof-of concept). Overenie konceptu prebieha obvykle v rámci existujúcej infraštruktúry organizácie a vrátane softvéru, ktorý bude testovaný po prípadnej implementácií nástroja. Vďaka overeniu konceptu dokážeme posúdiť, či bude nástroj pracovať s testovaným softvérom efektívne a prípadne identifikovať potrebné zmeny v infraštruktúre organizácie.

6.2.2 Pilotné projekty pri zavádzaní nástrojov do organizácie

Po úspešnom overení konceptu a finálnom výbere nástroja, začína zavedenie vybraného nástroja do organizácie obvykle pilotným projektom a to s týmito cieľmi:

- získavanie hlbších znalostí o nástroji, pochopenie jeho silných a slabých stránok
- vyhodnotenie vhodnosti nástroja s ohľadom k existujúcim procesom a postupom vrátane určenia potrebných zmien
- rozhodnutie o štandardných spôsoboch používania, spravovania, uchovávania a údržby nástroja a testovania pracovných produktov (napr. rozhodnutie o konvenciách na pomenovanie súborov a testov, výber noriem kódovania, vytváranie knižníc a definovanie modularity testovacích sád)
- posúdenie toho, či sú prínosy dosiahnuteľné za rozumnú cenu
- pochopenie metrík, ktoré chcete, aby nástroj zhromažďoval a reportoval vrátane samotnej konfigurácie nástroja

6.2.3 Faktory úspechu pri zavádzaní nástrojov

Faktory úspechu hodnotenia, implementácie, nasadenia a podpory nástrojov v rámci organizácie zahŕňajú:

- postupnú implementáciu nástrojov do organizácie
- prispôsobenie a zlepšenie procesov tak, aby zohľadňovali používanie nástrojov
- zaistenie školení, koučingu a mentorovania pre užívateľov nástrojov
- definovanie pokynov pre používanie nástrojov (napr. interné normy pre automatizáciu)
- zaistenie spôsobu získavania informácií o skutočnom používaní nástroja
- monitorovanie používania a prínosov nástroja
- poskytovanie podpory užívateľom nástroja
- získavanie a vyhodnocovanie poznatkov získaných od všetkých používateľov

Pri zavádzaní nástroja do organizácie je tiež dôležité zabezpečiť, aby bol nástroj technicky a organizačne integrovaný do životného cyklu vývoja softvéru. Za prevádzku a podporu nástroja môžu byť zodpovedné samostatné divízie vnútri organizácie a/alebo externí dodávatelia.

Pre viac informácií o nástrojoch na vykonávanie testov viď *Graham 2012.*

7 Referencie

Normy

ISO/IEC/IEEE 29119-1 (2013) Software and systems engineering - Software testing - Part 1: Concepts and definitions

ISO/IEC/IEEE 29119-2 (2013) Software and systems engineering - Software testing - Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2013) Software and systems engineering - Software testing - Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2015) Software and systems engineering - Software testing - Part 4: Test techniques

ISO/IEC 25010, (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models

ISO/IEC 20246: (2017) Software and systems engineering — Work product reviews

UML 2.5, Unified Modeling Language Reference Manual, http://www.omg.org/spec/UML/2.5.1/, 2017

ISTQB® Dokumenty

ISTQB® Glossary

ISTQB® Slovník (SK)

ISTQB® Foundation Level Overview 2018

ISTQB-CTFL-MBT Foundation Level Model-Based Tester Extension Syllabus

ISTQB-CTFL-AT Foundation Level Agile Tester Extension Syllabus

ISTQB-CTAL-TA Advanced Level Test Analyst Syllabus

ISTQB-CTAL-TTA Advanced Level Technical Test Analyst Syllabus

ISTQB-CTAL-TM Advanced Level Test Manager Syllabus

ISTQB-CTAL-SEC Advanced Level Security Tester Syllabus

ISTQB-CTAL-TAE Advanced Level Test Automation Engineer Syllabus

ISTQB-CTEL-TM Expert Level Test Management Syllabus

ISTQB-CTEL-ITP Expert Level Improving the Test Process SyllabusCertified Tester

Knihy a články

Beizer, B. (1990) *Software Testing Techniques (2e)*, Van Nostrand Reinhold: Boston MA

Black, R. (2017) *Agile Testing Foundations*, BCS Learning & Development Ltd: Swindon UK

Black, R. (2009) *Managing the Testing Process (3e)*, John Wiley & Sons: New York NY

Buwalda, H. et al. (2001) *Integrated Test Design and Automation*, Addison Wesley: Reading MA

Copeland, L. (2004) *A Practitioner's Guide to Software Test Design*, Artech House: Norwood MA

Craig, R. and Jaskiel, S. (2002) *Systematic Software Testing*, Artech House: Norwood MA

Crispin, L. and Gregory, J. (2008) *Agile Testing*, Pearson Education: Boston MA

Fewster, M. and Graham, D. (1999) *Software Test Automation*, Addison Wesley: Harlow UK

Gilb, T. and Graham, D. (1993) *Software Inspection*, Addison Wesley: Reading MA

Graham, D. and Fewster, M. (2012) *Experiences of Test Automation*, Pearson Education: Boston MA

Gregory, J. and Crispin, L. (2015) *More Agile Testing*, Pearson Education: Boston MA

Jorgensen, P. (2014) *Software Testing, A Craftsman's Approach (4e)*, CRC Press: Boca Raton FL

Kaner, C., Bach, J. and Pettichord, B. (2002) *Lessons Learned in Software Testing*, John Wiley & Sons: New York NY

Kaner, C., Padmanabhan, S. and Hoffman, D. (2013) *The Domain Testing Workbook*, Context-Driven Press: New York NY

Kramer, A., Legeard, B. (2016) Model-Based Testing Essentials: *Guide to the ISTQB® Certified ModelBased Tester: Foundation Level*, John Wiley & Sons: New York NY

Myers, G. (2011) *The Art of Software Testing, (3e)*, John Wiley & Sons: New York NY

Sauer, C. (2000) "The Effectiveness of Software Development Technical Reviews: A Behaviorally

Motivated Program of Research," *IEEE Transactions on Software Engineering, Volume 26, Issue 1, pp 1-*

Shull, F., Rus, I., Basili, V. July 2000. "How Perspective-Based Reading can Improve Requirement Inspections." *IEEE Computer, Volume 33, Issue 7, pp 73-79*

van Veenendaal, E. (ed.) (2004) *The Testing Practitioner* (Chapters 8 - 10), UTN Publishers: The Netherlands

Wiegers, K. (2002) *Peer Reviews in Software*, Pearson Education: Boston MA

Weinberg, G. (2008) *Perfect Software and Other Illusions about Testing*, Dorset House: New York NYCertified Tester

Ostatné zdroje (bez priamych odkazov v učebnej osnove)

Black, R., van Veenendaal, E. and Graham, D. (2019) *Foundations of Software Testing: ISTQB® Certification (4e)*, Cengage Learning: London UK

Hetzel, W. (1993) *Complete Guide to Software Testing (2e)*, QED Information Sciences: Wellesley MA

8 Príloha A – Obecné informácie k učebnej osnove

História dokumentu

Tento dokument je učebná osnova pre certifikáciu ISTQB® Certifikovaný tester základnej úrovne (ISTQB Certified Tester Foundation Level), čo je prvá úroveň medzinárodnej kvalifikácie schválenej ISTQB® (www.istqb.org).

Tento dokument bol pripravený behom júna až augusta 2019 pracovnou skupinou z členov, ktorí boli menovaní organizáciou International Software Testing Qualifications Board (ISTQB®). Aktualizácie boli pridané po vstupoch od lokálnych členských výborov, ktoré používali učebné osnovy základnej úrovne - Foundation Syllabus 2018.

Predošlá verzia tohto dokumentu bola vypracovaná v rokoch 2014 až 2018 pracovnou skupinou pozostávajúcou z členov menovaných ISTQB®. Verzia 2018 bola spočiatku revidovaná zástupcami všetkých členských výborov ISTQB® a následne zástupcami vybranými z medzinárodnej komunity pre testovanie softvéru.

Ciele certifikácie základnej úrovne

- Získať uznanie v oblasti testovania ako nevyhnutnú a profesionálnu súčasť softvérového inžinierstva.
- Poskytnúť štandardný rámec pre rozvoj kariéry testerov.
- Umožniť profesionálne kvalifikovaným testerom získať uznanie u zamestnávateľov, zákazníkov a partnerov a zlepšiť postavenie testerov.
- Podporovať konzistentné a osvedčené postupy testovania v rámci všetkých disciplín softvérového inžinierstva.
- Identifikovať oblasti testovania, ktoré sú relevantné a prínosné pre dané odvetvie.
- Umožniť dodávateľom softvéru najať certifikovaných testerov a politikou náboru a rozvoja testerov získať obchodnú výhodu nad svojimi konkurentmi.
- Poskytnúť príležitosť testerom alebo osobám so záujmom o testovanie získať medzinárodne uznávanú kvalifikáciu v oblasti testovania.

Ciele medzinárodnej kvalifikácie

- Umožniť porovnávať znalosti v oblasti testovania naprieč krajinami.
- Uľahčiť uplatnenie testerov v zahraničí.
- Umožniť nadnárodným a medzinárodným projektom mať jednotné chápanie problematiky testovania.
- Celosvetovo zvýšiť počet kvalifikovaných testerov.
- Ako medzinárodná iniciatíva mať väčší vplyv a hodnotu v porovnaní s lokálnymi riešeniami v rámci krajiny alebo regiónu.
- Rozvíjať jednotný medzinárodný súbor znalostí a vedomostí o testovaní prostredníctvom učebných osnov a terminológie. Zvýšiť úroveň znalostí o testovaní pre všetkých zúčastnených.
- Podporovať testovanie ako profesiu v ďalších krajinách sveta.
- Umožniť testerom získať uznávanú kvalifikáciu v ich rodnom jazyku.
- Umožniť zdieľanie vedomostí a zdrojov naprieč krajinami sveta.

- Zabezpečiť medzinárodnú kvalifikáciu a uznanie testerov prostredníctvom zapojenia viacerých krajín.

Vstupné požiadavky pre túto kvalifikáciu

Vstupné kritérium pre absolvovanie skúšky z ISTQB® Certifikovaný tester základnej úrovne, je záujem kandidátov o testovanie softvéru. Dôrazne sa však odporúča, aby kandidáti tiež:

- Mali aspoň minimálne skúsenosti buď s vývojom, alebo testovaním softvéru v rozsahu šiestich mesiacov na pozícii testera systémových alebo akceptačných testov, prípadne ako vývojár softvéru.
- Absolvovali školenie akreditované podľa ISTQB® noriem jedným z členských výborov ISTQB.

História certifikácie základnej úrovne v oblasti testovania softvéru

Nezávislú certifikáciu testerov softvéru zahájil vo Veľkej Británii Skúšobný výbor informačných systémov (Information Systems Examination Board, ISEB) založením Výboru pre testovanie softvéru (Software Testing Board, www.bcs.org.uk/iseb) v roku 1998. V roku 2002 začala spoločnosť ASQF v Nemecku podporovať nemecký kvalifikačný program pre testerov (German Tester Qualification Scheme, www.asqf.de). Táto učebná osnova je založená na osnovách ISEB a ASQF, no zahŕňa reorganizovaný, aktualizovaný a rozšírený obsah. Hlavný dôraz je kladený na témy, ktoré najviac pomáhajú testerom po praktickej stránke.

Existujúci certifikát základnej úrovne v testovaní softvéru (napr. z ISEB, ASQF alebo z jedného z uznaných členských výborov ISTQB) pridelený pred vydaním tohto medzinárodného certifikátu sa považuje za jemu rovnocenný. Certifikát základnej úrovne má neobmedzenú platnosť a nie je nutné ho obnovovať. Dátum vydania je uvedený na certifikáte.

V rámci každej zo zúčastnených krajín sú lokálne záležitosti riadené národným alebo regionálnym výborom pre testovanie softvéru uznaným ISTQB®. Povinnosti členských výborov sú definované na úrovni ISTQB globálne, ale implementované sú v rámci každej krajiny samostatne. Medzi povinnosti lokálnych výborov patrí akreditovanie poskytovateľov školení a zabezpečenie skúšok.

9 Príloha B – Študijné ciele a kognitívna úroveň znalostí

Pre účely tejto učebnej osnovy sú definované nasledujúce študijné ciele. Každá téma v učebnej osnove bude preskúšaná podľa odpovedajúceho študijného cieľa.

Úroveň 1: Zapamätať si (K1)

Kandidát je schopný rozpoznať, zapamätať si a vybaviť si daný termín alebo pojem.

Kľúčové slová: identifikovať, zapamätať si, spomenúť si, vybaviť si, určiť, poznať

Príklady:

Dokážete určiť definíciu "zlyhania" ako:

- "Nedoručenie služby koncovému používateľovi alebo akejkoľvek inej zainteresovanej strane" alebo
- "Odchýlka komponentu alebo systému od jeho očakávanej dodávky, doručenia alebo výsledku"

Úroveň 2: Pochopiť (K2)

Kandidát dokáže vybrať príčiny alebo vysvetlenia k výrokom vzťahujúcim sa k téme, dokáže zhrnúť, porovnať, klasifikovať, roztriediť a uviesť príklady pre koncept testovania.

Kľúčové slová: zhrnúť, zovšeobecniť, abstrahovať, klasifikovať, porovnať, namapovať, odlíšiť, ilustrovať, interpretovať, previesť, znázorniť, odvodiť, vyvodiť, roztriediť, vytvoriť modely.

Príklady:

Dokážete vysvetliť dôvod, prečo by sa testovacia analýza a návrh mali vykonať čo najskôr:

- Nájsť defekty v bode, kde je ich odstránenie lacnejšie
- Nájsť najvýznamnejšie defekty čo najskôr

Dokážete vysvetliť podobnosti a rozdiely medzi integračným testovaním a systémovým testovaním:

- Podobnosti: testovacie objekty pre integračné a systémové testovanie tvorí viac ako jeden komponent, a ako integračné a systémové testovanie môže zahŕňať nefunkcionálne typy testov.
- Rozdiely: integračné testovanie sa sústreďuje na rozhrania a interakcie, zatiaľ čo systémové testovanie sa sústreďuje na aspekty celého systému, ako je spracovanie od začiatku do konca (end-to-end)

Úroveň 3: Aplikovať (K3)

Kandidát dokáže zvoliť správny koncept či techniku a použiť ich v danom kontexte

Kľúčové slová: implementovať, vykonať, použiť, sledovať postup, použiť postup

Príklady:

- Dokáže identifikovať hraničné hodnoty pre platné a neplatné triedy ekvivalencie
- Dokáže zvoliť testovacie prípady z diagramu prechodu stavov s cieľom pokryť všetky prechody.

Referencie

(pre kognitívne úrovne vzdelávacích cieľov)

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 Príloha C – poznámky k vydaniu

Učebná osnova ISTQB® Foundation Syllabus 2018 V3.1 je malou aktualizáciou verzie 2018. Samostatné poznámky k vydaniu (release notes) 2018 V3.1 obsahujú sumár zmien každej kapitoly. Okrem toho bola vydaná verzia so sledovaním zmien.

ISTQB® učebná osnova základnej úrovne z roku 2018 je výraznou aktualizáciou učebnej osnovy vydanej z roku 2011. Z tohto dôvodu nie sú k dispozícii žiadne podrobné poznámky k vydaniu na úrovni jednotlivých kapitol a sekcií. Avšak prehľad hlavných zmien je uvedený tu. Okrem toho (v samostatnom dokumente s poznámkami k vydaniu) poskytuje ISTQB® trasovateľnosť medzi študijnými cieľmi učebnej osnovy základnej úrovne z roku 2011 a študijnými cieľmi učebnej osnovy základnej úrovne z roku 2018, kde je popísané, ktoré študijné ciele boli pridané, aktualizované alebo odstránené.

Na začiatku roku 2017 bolo evidovaných viac ako 550 000 osôb vo viac než 100 krajinách, ktorí absolvovali skúšku základnej úrovne, z toho bolo viac ako 500 000 certifikovaných testerov.
Za predpokladu, že za účelom zloženia skúšky si všetci z nich prečítali učebné osnovy základnej úrovne, jedná sa pravdepodobne o najčítanejší dokumentu v oblasti testovania vôbec.

Táto významná aktualizácia bola vykonaná s ohľadom k tomuto dedičstvu s cieľom zvýšiť hodnotu poskytovanú zo strany ISTQB® ďalším 500 000 ľuďom v globálnej komunite testerov.

V tejto verzii boli upravené všetky študijné ciele tak, aby boli atomické a poskytovali jasnú trasovateľnosť medzi každým študijným cieľom a konkrétnou sekciou / konkrétnym sekciám obsahu (a skúškovým otázkam), ktoré sa vzťahujú k tomuto študijnému cieľu. Rovnako tak je zaistená spätná trasovateľnosť od sekcií obsahu (a skúškových otázok) k pridruženému študijnému cieľu. Okrem toho časové intervaly pridelené jednotlivým kapitolám sú teraz realistickejšie než tie, ktoré obsahovali učebné osnovy z roku 2011, pričom boli použité osvedčené heuristiky a vzorce použité v iných učebných osnovách ISTQB®. Tieto postupy sú založené na analýze študijných cieľov, ktoré majú byť zahrnuté v každej kapitole.

Napriek tomu, že sa jedná o učebné osnovy základnej úrovne vyjadrujúce osvedčené postupy a techniky, ktoré obstáli v skúške času, prináša táto učebná osnova zmeny vzhľadom na modernizáciu prezentácie materiálu, najmä pokiaľ ide o metódy vývoja softvéru (napr. Scrum a princíp priebežného nasadzovanie) a technológie (napr. internet vecí). Zároveň došlo k aktualizácii použitých noriem tak, aby boli viac relevantné:

1. Norma ISO/IEC/IEEE 29119 nahrádza normu IEEE 829
2. Norma ISO/IEC 25010 nahrádza normu ISO 9126
3. Norma ISO/IEC 20246 nahrádza normu IEEE 1028

Z dôvodu dramatického rozšírenie portfólia ISTQB® za posledných desať rokov boli do osnov pridané, v prípade potreby, rozširujúce krížové odkazy na súvisiace materiály obsiahnuté v iných učebných osnovách ISTQB®. Zároveň bola vykonaná revízia za účelom zosúladenia s ostatnými učebnými osnovami ISTQB® a slovníkom pojmov. Cieľom je uľahčiť čítanie, porozumenie, učenie a preklad so zameraním na zvýšenie praktickej užitočnosti a rovnováhy medzi znalosťami a zručnosťami.

Podrobnú analýzu zmien vykonaných v tejto verzii nájdete v dokumente *ISTQB® Certified Tester Foundation Level Release Notes 2018*